



UNIVERSIDAD ESAN  
FACULTAD DE INGENIERÍA  
INGENIERÍA DE TECNOLOGÍAS DE INFORMACIÓN Y SISTEMAS

Diseño de un modelo de Deep Learning para el pre-diagnóstico de nódulos tiroideos a través  
de imágenes de ultrasonido

Tesis de investigación para el curso Trabajo de Tesis II

Antonny Fernando Corcino Castillo  
Asesor: Marks Calderón

Lima, 10 de junio de 2024

Esta tesis denominada:

**DISEÑO DE UN MODELO DE DEEP LEARNING PARA EL PRE-DIAGNÓSTICO DE  
NÓDULOS TIROIDEOS A TRAVÉS DE IMÁGENES DE ULTRASONIDO**

ha sido aprobada.

.....

Jurado 1

.....

Jurado 2

.....

Jurado 3

Universidad ESAN

2024

DISEÑO DE UN MODELO DE DEEP LEARNING PARA EL PRE-DIAGNÓSTICO DE  
NÓDULOS TIROIDEOS A TRAVÉS DE IMÁGENES DE ULTRASONIDO

## **Dedicatoria**

A mi madre.

Gracias por tu apoyo todos estos años.

## Índice general

<b>Resumen</b>	<b>1</b>
<b>Introducción</b>	<b>3</b>
<b>Capítulo I: Planteamiento del Problema</b>	<b>5</b>
1.1 Descripción de la Realidad Problemática	5
1.2 Formulación del Problema	9
1.2.1 Problema General	9
1.2.2 Problemas Específicos	9
1.3 Objetivos de la Investigación	10
1.3.1 Objetivo General	10
1.3.2 Objetivos Específicos	10
1.4 Justificación de la Investigación	10
1.4.1 Teórica	10
1.4.2 Práctica	11
1.4.3 Metodológica	11
1.5 Delimitación del Estudio	11
1.5.1 Espacial	11
1.5.2 Temporal	12
1.5.3 Conceptual	12
<b>Capítulo II: Marco Teórico</b>	<b>13</b>
2.1 Antecedentes de la investigación	13
2.2 Bases Teóricas	50
2.2.1 Deep Learning	50
2.2.2 Transfer Learning	58
2.2.3 Data Augmentation	59
2.2.4 Nódulos Tiroideos	60
2.2.5 Ecografía y las imágenes de ultrasonido	62
2.3 Marco Conceptual	63
2.3.1 Inteligencia Artificial	63
2.3.2 Perceptrón	63
2.3.3 Perceptrón Multicapa	64
2.3.4 Machine Learning	65
2.4 Hipótesis	66
2.4.1 Hipótesis General	66

2.4.2	Hipótesis Específicas . . . . .	66
<b>Capítulo III:</b>	<b>Metodología de la Investigación . . . . .</b>	<b>67</b>
3.1	Diseño de la investigación . . . . .	67
3.1.1	Tipo de la investigación . . . . .	67
3.1.2	Enfoque de la investigación . . . . .	67
3.1.3	Población . . . . .	67
3.1.4	Muestra . . . . .	68
3.2	Metodología de Implementación de la Solución . . . . .	68
3.3	Metodología para la Medición de Resultados de la Implementación . . . . .	69
3.4	Cronograma de actividades y presupuesto . . . . .	72
<b>Capítulo IV:</b>	<b>Desarrollo de la Solución . . . . .</b>	<b>74</b>
4.1	Determinación y evaluación de alternativas de solución . . . . .	74
4.2	Propuesta solución . . . . .	77
4.2.1	Planeamiento y descripción de Actividades . . . . .	77
4.2.2	Desarrollo de actividades . . . . .	80
4.3	Medición de la solución . . . . .	124
4.3.1	Análisis de Indicadores cuantitativo . . . . .	124
4.3.2	Simulación de solución . . . . .	125
<b>Referencias . . . . .</b>		<b>128</b>
<b>Anexos . . . . .</b>		<b>132</b>
A	Árbol de Problemas . . . . .	133
B	Árbol de Objetivos . . . . .	134
C	Matriz de Consistencia . . . . .	135

## Índice de Figuras

Figura 1.	Tasa bruta de incidencia de cáncer de tiroides en mujeres por 100 000 personas . . . . .	5
Figura 2.	Tasa bruta de mortalidad de cáncer de tiroides en mujeres por 100 000 personas . . . . .	6
Figura 3.	Tasa bruta de incidencia de cáncer de tiroides en varones por 100 000 personas . . . . .	6
Figura 4.	Tasa bruta de mortalidad de cáncer de tiroides en varones por 100 000 personas . . . . .	7
Figura 5.	Tasa bruta de incidencia y mortalidad de cáncer de tiroides por género y región . . . . .	8
Figura 6.	Representación del método desarrollado . . . . .	17
Figura 7.	Arquitectura de modelo TC-ViT . . . . .	18
Figura 8.	Arquitectura de la red desarrollada . . . . .	20
Figura 9.	Balanceo de clases usando Aumento de Datos . . . . .	26
Figura 10.	Estructura de ViT implementado . . . . .	27
Figura 11.	Imágenes de ultrasonido de la tiroides generadas . . . . .	47
Figura 12.	Convolución de imagen de Fashion MNIST . . . . .	50
Figura 13.	Convolución de imagen con filtro de líneas verticales . . . . .	51
Figura 14.	Convolución de imagen con filtro de líneas horizontales . . . . .	51
Figura 15.	Ejemplo de max-pooling con un pool de 2x2 . . . . .	52
Figura 16.	DenseNet con 5 capas . . . . .	54
Figura 17.	Arquitectura del modelo Transformer . . . . .	56
Figura 18.	Arquitectura del modelo ViT . . . . .	58
Figura 19.	Ejemplo de Transfer Learning . . . . .	59
Figura 20.	Ejemplo de Data Augmentation . . . . .	60
Figura 21.	Imagen tomográfica de glándula de tiroides . . . . .	61
Figura 22.	Imágenes de ultrasonido de nódulos tiroideos . . . . .	63
Figura 23.	Perceptrón . . . . .	64
Figura 24.	Perceptrón Multicapa . . . . .	65
Figura 25.	Enfoque del Machine Learning . . . . .	66
Figura 26.	Metodología de implementación . . . . .	68
Figura 27.	Matriz de Confusión . . . . .	70
Figura 28.	Cronograma de actividades . . . . .	72
Figura 29.	Imágenes de ultrasonido de la tiroides generadas . . . . .	76
Figura 30.	Ejemplo de imágenes del conjunto de datos de la Universidad de Colombia	81

Figura 31.	Ejemplo de archivo XML del conjunto de datos de la Universidad de Colombia . . . . .	81
Figura 32.	Ejemplo de imágenes del conjunto de datos de Zhuijiang Hospital of Southem Medical University . . . . .	82
Figura 33.	Ejemplo de archivo CSV del conjunto de datos de Zhuijiang Hospital of Southem Medical University . . . . .	83
Figura 34.	Gráfica circular de conjunto de datos de entrenamiento . . . . .	84
Figura 35.	Gráfica circular de conjunto de datos de prueba . . . . .	85
Figura 36.	Algoritmo de organización de imágenes . . . . .	86
Figura 37.	Nueva distribución de ficheros para los datos . . . . .	86
Figura 38.	Algoritmo para lectura de los datos en DataLoader . . . . .	87
Figura 39.	Arquitectura de Red Generadora . . . . .	89
Figura 40.	Arquitectura de Red Discriminadora . . . . .	89
Figura 41.	Parte final del entrenamietno del DCGAN . . . . .	90
Figura 42.	Plot de pérdidas de las red Generadora y Discriminadora . . . . .	91
Figura 43.	Imágenes de ultrasonido de nódulos tiroideos generados y reales . . . . .	91
Figura 44.	Parte final del entrenamietno del DCGAN con imágenes de nódulos malignos	92
Figura 45.	Plot de pérdidas de las red Generadora y Discriminadora de imágenes de nódulos malignos . . . . .	93
Figura 46.	Imágenes de ultrasonido de nódulos tiroideos generados y reales (DCGAN de imágenes de nódulos malignos) . . . . .	93
Figura 47.	Gráfica circula de data de entrenamiento balanceada . . . . .	94
Figura 48.	Proceso de entrenamiento del modelo n°1 . . . . .	96
Figura 49.	Recall y Precision del modelo n°1 . . . . .	96
Figura 50.	Proceso de entrenamiento del modelo n°2 . . . . .	97
Figura 51.	Recall y Precision del modelo n°2 . . . . .	98
Figura 52.	Proceso de entrenamiento del modelo n°3 . . . . .	98
Figura 53.	Recall y Precision del modelo n°3 . . . . .	99
Figura 54.	Proceso de entrenamiento del modelo n°4 . . . . .	100
Figura 55.	Recall y Precision del modelo n°4 . . . . .	100
Figura 56.	Proceso de entrenamiento del modelo n°5 . . . . .	101
Figura 57.	Recall y Precision del modelo n°5 . . . . .	101
Figura 58.	Proceso de entrenamiento del modelo n°6 . . . . .	102
Figura 59.	Recall y Precision del modelo n°6 . . . . .	103
Figura 60.	Proceso de entrenamiento del modelo n°7 . . . . .	103
Figura 61.	Recall y Precision del modelo n°7 . . . . .	104
Figura 62.	Proceso de entrenamiento del modelo n°8 . . . . .	104

Figura 63.	Recall y Precision del modelo n°8 . . . . .	105
Figura 64.	Proceso de entrenamiento del modelo n°9 . . . . .	105
Figura 65.	Recall y Precision del modelo n°9 . . . . .	106
Figura 66.	Proceso de entrenamiento del modelo n°10 . . . . .	106
Figura 67.	Recall y Precision del modelo n°10 . . . . .	107
Figura 68.	Proceso de entrenamiento del modelo n°11 . . . . .	107
Figura 69.	Recall y Precision del modelo n°11 . . . . .	108
Figura 70.	Proceso de entrenamiento del modelo n°12 . . . . .	108
Figura 71.	Recall y Precision del modelo n°12 . . . . .	109
Figura 72.	Proceso de entrenamiento del modelo n°13 . . . . .	110
Figura 73.	Recall y Precision del modelo n°13 . . . . .	110
Figura 74.	Proceso de entrenamiento del modelo n°14 . . . . .	111
Figura 75.	Recall y Precision del modelo n°14 . . . . .	111
Figura 76.	Proceso de entrenamiento del modelo n°15 . . . . .	112
Figura 77.	Recall y Precision del modelo n°15 . . . . .	112
Figura 78.	Proceso de entrenamiento del modelo n°16 . . . . .	113
Figura 79.	Recall y Precision del modelo n°16 . . . . .	113
Figura 80.	Proceso de entrenamiento del modelo n°17 . . . . .	114
Figura 81.	Recall y Precision del modelo n°17 . . . . .	114
Figura 82.	Proceso de entrenamiento del modelo n°18 . . . . .	115
Figura 83.	Recall y Precision del modelo n°18 . . . . .	115
Figura 84.	Proceso de entrenamiento del modelo n°19 . . . . .	116
Figura 85.	Recall y Precision del modelo n°19 . . . . .	116
Figura 86.	Proceso de entrenamiento del modelo n°20 . . . . .	117
Figura 87.	Recall y Precision del modelo n°20 . . . . .	117
Figura 88.	Proceso de entrenamiento del modelo n°21 . . . . .	118
Figura 89.	Recall y Precision del modelo n°21 . . . . .	118
Figura 90.	Proceso de entrenamiento del modelo n°22 . . . . .	119
Figura 91.	Recall y Precision del modelo n°22 . . . . .	119
Figura 92.	Proceso de entrenamiento del modelo n°23 . . . . .	120
Figura 93.	Recall y Precision del modelo n°23 . . . . .	120
Figura 94.	Proceso de entrenamiento del modelo n°24 . . . . .	121
Figura 95.	Recall y Precision del modelo n°24 . . . . .	121
Figura 96.	Proceso de entrenamiento del modelo híbrido . . . . .	123
Figura 97.	Recall y Precision del modelo híbrido . . . . .	123
Figura 98.	Prueba de modelo con imagen de ultrasonido de nódulo maligno . . . . .	126
Figura 99.	Prueba de modelo con imagen de ultrasonido de nódulo benigno . . . . .	127

## Índice de Tablas

Tabla 1.	Comparación de resultados de los modelos (precision) . . . . .	22
Tabla 2.	Comparación de resultados de los modelos (sensitivity) . . . . .	22
Tabla 3.	Comparación de resultados de los modelos (f1-score) . . . . .	23
Tabla 4.	Comparación de resultados de los modelos (specificity) . . . . .	23
Tabla 5.	Comparación de resultados según accuracy con el conjunto de datos HAM10000	24
Tabla 6.	Comparación de resultados según accuracy con el conjunto de datos DER-MOFIT . . . . .	24
Tabla 7.	Resultado de los modelo ViT basados en Transfer Learning en el conjunto de datos DDSM (accuracy, AUC, f1-score) . . . . .	29
Tabla 8.	Resultado de los modelo ViT basados en Transfer Learning en el conjunto de datos DDSM (precision, recall, MCC) . . . . .	29
Tabla 9.	Resultado de los modelo ViT basados en Transfer Learning en el conjunto de datos DDSM (kappa) . . . . .	29
Tabla 10.	Resultado de los modelo ViT entrenados desde cero en el conjunto de datos DDSM (accuracy, AUC, f1-score) . . . . .	30
Tabla 11.	Resultado de los modelo ViT entrenados desde cero en el conjunto de datos DDSM (precision, recall, MCC) . . . . .	30
Tabla 12.	Resultado de los modelo ViT entrenados desde cero en el conjunto de datos DDSM (kappa) . . . . .	30
Tabla 13.	Resultado de los modelos CNN basados en Transfer Learning en el conjunto de datos DDSM (accuracy, AUC, f1-score) . . . . .	31
Tabla 14.	Resultado de los modelos CNN basados en Transfer Learning en el conjunto de datos DDSM (precision, recall, MCC) . . . . .	31
Tabla 15.	Resultado de los modelos CNN basados en Transfer Learning en el conjunto de datos DDSM (kappa) . . . . .	31
Tabla 16.	Comparación de resultados de los modelos MedViT con los CNN y AutoML (PathMNIST, ChestMNIST y DermaMNIST) . . . . .	34
Tabla 17.	Comparación de resultados de los modelos MedViT con los CNN y AutoML (OCTMNIST, PneumoniaMNIST y RetinaMNIST) . . . . .	35
Tabla 18.	Comparación de resultados de los modelos MedViT con los CNN y AutoML (BreastMNIST, BloodMNIST y TissueMNIST) . . . . .	35
Tabla 19.	Comparación de resultados de los modelos MedViT con los CNN y AutoML (OrganMNIST, OrganCMNIST y OrganSMNIST) . . . . .	36
Tabla 20.	Resumen de características de los conjuntos de datos . . . . .	38
Tabla 21.	Resultados con el conjunto de datos Chest X-ray . . . . .	39

Tabla 22.	Resultados con el conjunto de datos Kvasir . . . . .	40
Tabla 23.	Resultados con el conjunto de datos Kvasir-Capsule . . . . .	41
Tabla 24.	Resultados de los modelos entrenados con el conjunto de datos 2D (MCC, accuracy y f1-score) . . . . .	44
Tabla 25.	Resultados de los modelos entrenados con el conjunto de datos 2D (AUC, precision y recall) . . . . .	45
Tabla 26.	Resultados de los modelos entrenados con el conjunto de datos 3D (MCC, accuracy y f1-score) . . . . .	45
Tabla 27.	Resultados de los modelos entrenados con el conjunto de datos 3D (AUC, precision y recall) . . . . .	45
Tabla 28.	Comparación de los modelos entrenados . . . . .	49
Tabla 29.	Presupuesto . . . . .	73
Tabla 30.	Resultados de los modelos entrenados con los dos conjuntos de datos, técnicas de entrenamiento y pesos . . . . .	124

## Índice de Ecuaciones

Ecuación 1 Fórmula para calcular el accuracy . . . . .	70
Ecuación 2 Fórmula para calcular el recall . . . . .	71
Ecuación 3 Fórmula para calcular el precision . . . . .	71

## Resumen

Con los casos de cáncer de tiroides incrementando cada año en todo el mundo, los especialistas en el área y diversos investigadores se han visto en la necesidad de encontrar un método probado o herramienta eficaz que ayude a los médicos y radiólogos, con distintos niveles de experiencia, a realizar detecciones y diagnósticos de nódulos tiroideos de forma más rápida y con menos errores, incluso con la capacidad de llegar a ser una opción factible en el contexto de las clínicas u hospitales de bajos recursos y poco personal especializado para realizar un diagnóstico temprano de este mal y así evitar futuras complicaciones de cáncer avanzado, ya que existe un porcentaje nada despreciable que un nódulo llegue a convertirse con el paso del tiempo, y sin tratamiento, en un problema mucho más grave difícil de sobrellevar.

Por estos motivos, y tomando en cuenta el grado de desarrollo actual del extenso campo de la Inteligencia Artificial y junto con el fácil acceso al gran poder computacional, en la presente investigación se plantea desarrollar un modelo de Deep Learning que sirva como herramienta de soporte en el diagnóstico de nódulos tiroideos para mejorar la eficacia y eficiencia del diagnóstico a través de imágenes de ultrasonido.

**Palabras claves:** Aprendizaje profundo, Redes Neuronales Convolucionales, Vision Transfomers, imágenes, ultrasonido, tiroides, clasificación.

## Abstract

With thyroid cancer cases increasing every year around the world, specialists and researchers have found it necessary to find a proven method or effective tool to help physicians and radiologists, with different levels of experience, to detect and diagnose thyroid nodules more quickly and with fewer errors. Even with the ability to become a feasible option in the context of clinics or hospitals with low resources and few specialized personnel to make an early diagnosis of this disease and thus avoid future complications of advanced cancer, since there is a not insignificant percentage that a nodule becomes malignant with the passage of time, and without treatment, a much more serious problem difficult to cope with.

For these reasons, and taking into account the current degree of development of the extensive field of Artificial Intelligence and the easy access to great computational power, this research proposes to develop a Deep Learning model that serves as a support tool in the diagnosis of thyroid nodules to improve the effectiveness and efficiency of diagnosis through ultrasound images.

**Keywords:** Deep Learning, Convolutional Neural Networks, Vision Transformers, images, ultrasound, thyroid, classification.

## Introducción

La presente investigación tiene como principal objetivo el desarrollo y propuesta de un modelo de inteligencia para pre-diagnosticar de manera correcta y rápida la naturaleza de un nódulo en la glándula de la tiroides, esto a través de la clasificación de este en benigno o maligno. Se plantea utilizar recursos y algoritmos que brinda el extenso campo de estudio de la Inteligencia Artificial. En particular, se usarán los extensamente desarrollados algoritmos de Deep Learning como las Redes Neuronales Convolucionales y Vision Transfomers para lograr encontrar patrones en imágenes de ultrasonido de la glándula de tiroides, y así lograr correctos pre-diagnósticos en nuevos casos que se presenten al modelo. Las imágenes de ultrasonido de la glándula de tiroides que se usará para el entrenamiento y prueba de los distintos modelos a entrenar procederán de un repositorio en línea debidamente validado.

El proceso que se seguirá para desarrollar un modelo eficaz, capaz de reducir el tiempo que le toma a un médico o radiólogo en detectar y diagnosticar un nódulo tiroideo consiste en una parte inicial de análisis y filtrado de las imágenes, esto con el objetivo de obtener un buen conjunto de datos para el siguiente paso que es el modelado. Se usarán distintos modelos y técnicas dentro del área del Deep Learning. Finalmente, se usarán métricas para evaluar cada uno de estos modelos, con el objetivo final de determinar aquel de mejor desempeño. Esto se mostrará a más detalle en la descripción de la metodología de la implementación.

Esta herramienta de Inteligencia Artificial no pretende ser un sustituto al diagnóstico de un profesional especializado en este tipo de casos. Como se verá más adelante, en las investigaciones presentadas como antecedentes donde se desarrollan modelos capaces de clasificar nódulos en la glándula de tiroides a través de imágenes, el objetivo final es brindar una herramienta eficaz y eficiente que permita a los profesionales del sector a realizar diagnósticos más rápidos. Esto es importante debido a la naturaleza de desarrollo de esta enfermedad, pues con una detección temprana y correcta del tipo de nódulo con el que el especialista se enfrenta, se puede acelerar tratamientos eficaces y evitar aquellos que son totalmente innecesarios para el paciente.

El problema a abordar, y principal incentivo de la investigación, radica en el proceso prolongado y de no tan lata precisión que, de manera general, los médicos o radiólogos han ido desarrollando en el diagnóstico de nódulos tiroideos, además de las escasas herramientas basadas en Inteligencia Artificial en este campo. También observar que los casos de cáncer en esta glándula van en aumento en el Perú y el mundo, genera una preocupación y alta necesidad de desarrollar nuevos y mejores métodos o herramientas junto con los grandes avances tecnológicos como la Inteligencia Artificial que ha desmotrado ser capaz de reducir los índices de

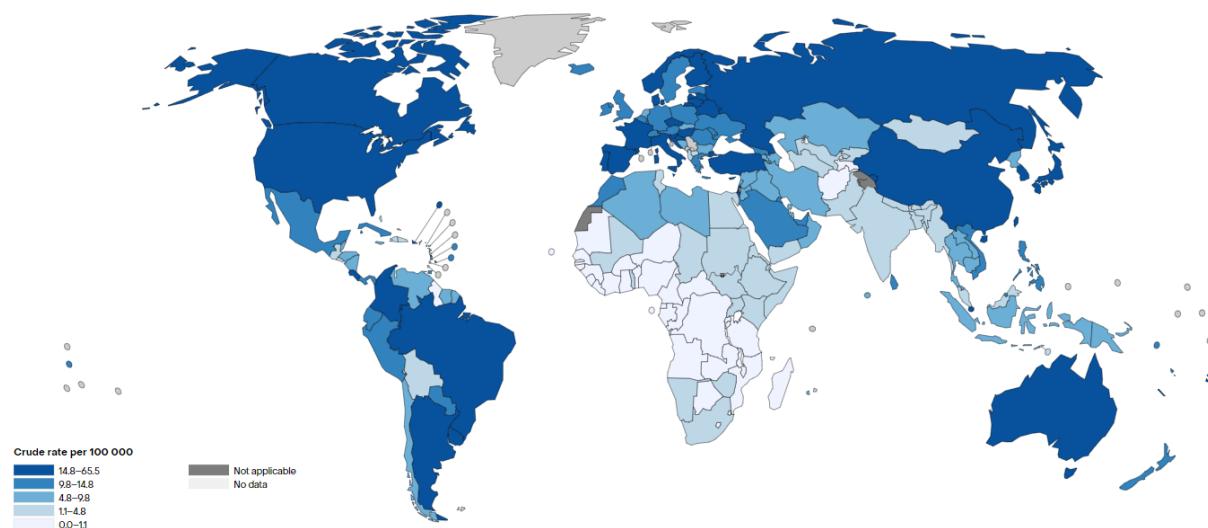
errores y mejorar los tiempos en el diagnóstico o clasificación de un nódulo benigno o maligno.

## Capítulo I: Planteamiento del Problema

### 1.1 Descripción de la Realidad Problemática

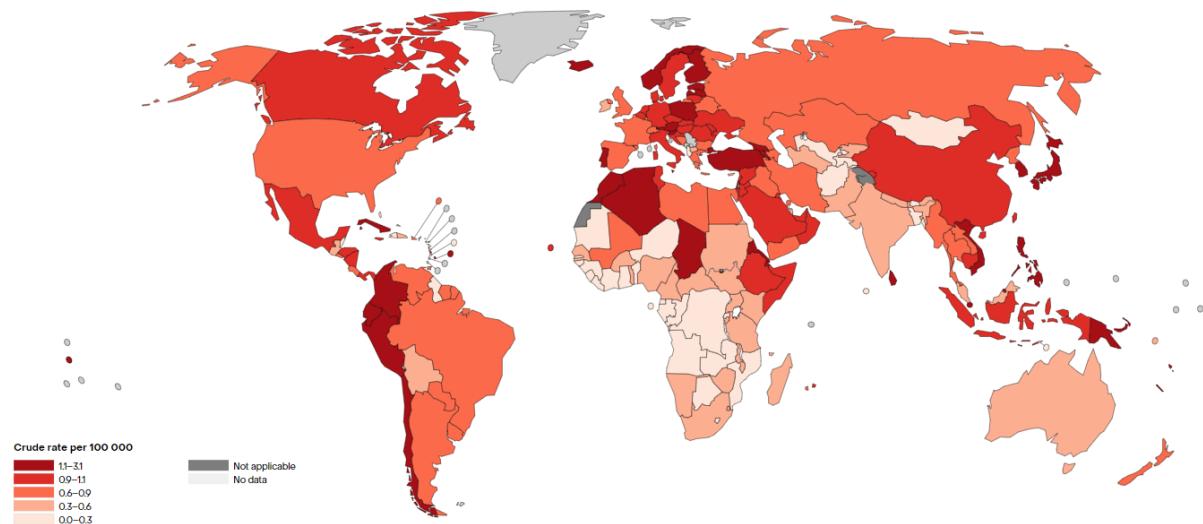
A nivel mundial, las tasas de presencia de cáncer de tiroides en personas varían de acuerdo con la edad, al género y al país. En el caso de Perú se tiene que por cada 100 000 mujeres existen 10.9 de casos de incidencia en este tipo de cáncer, y por cada 100 000 varones, se tiene un 3.6 de casos. Además de los casos de incidencia, la mortalidad también está presente y varía en varias partes del mundo. En Perú la mortalidad de pacientes mujeres con cáncer de tiroides es 1.7 por cada 100 000 personas, mientras que en los varones es de 0.77 casos cada 100 000 personas. (Organización Mundial de la Salud, 2022)

A continuación, se presentan la Figura 1 y la Figura 2 que muestran los distintos índices de incidencia y mortalidad de cáncer de tiroides en mujeres, donde se puede resaltar la presencia de Perú en los rangos más altos de cada uno de estos.



**Figura 1.** Tasa bruta de incidencia de cáncer de tiroides en mujeres por 100 000 personas.

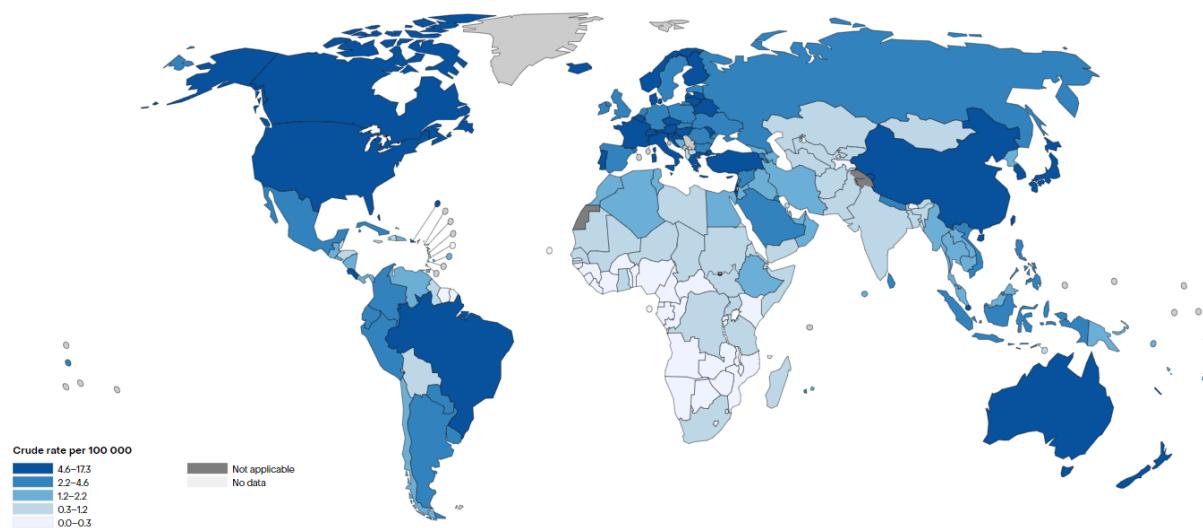
Fuente: Organización Mundial de la Salud (2022). *Cancer Today*.



**Figura 2.** Tasa bruta de mortalidad de cáncer de tiroides en mujeres por 100 000 personas.

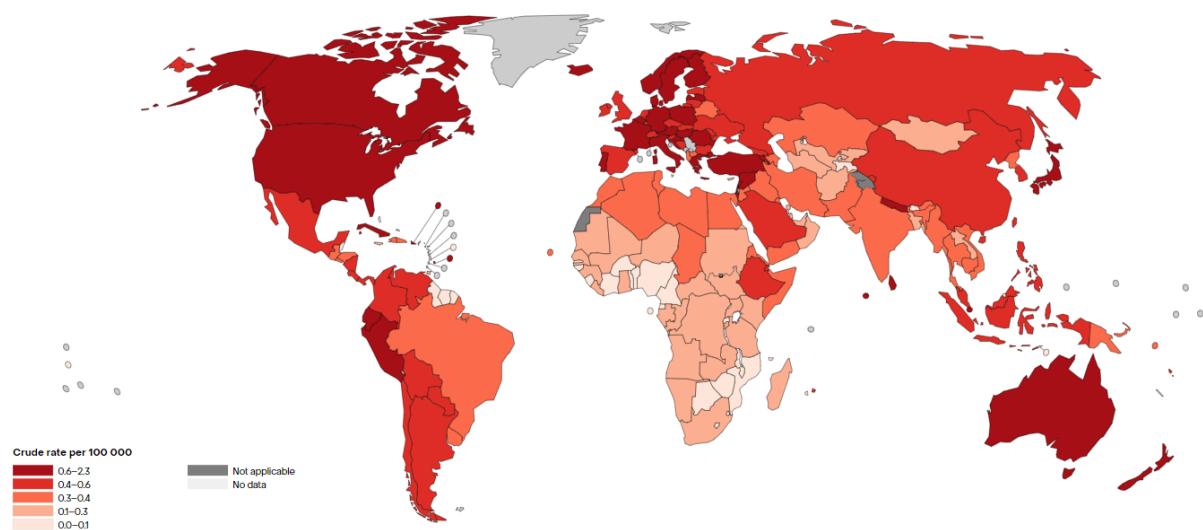
Fuente: Organización Mundial de la Salud (2022). *Cancer Today*.

De igual forma, en el caso de los varones, el Perú también se encuentra entre los rangos más altos de incidencia y mortalidad. A continuación, se muestran estos datos en la Figura 3 y Figura 4.



**Figura 3.** Tasa bruta de incidencia de cáncer de tiroides en varones por 100 000 personas.

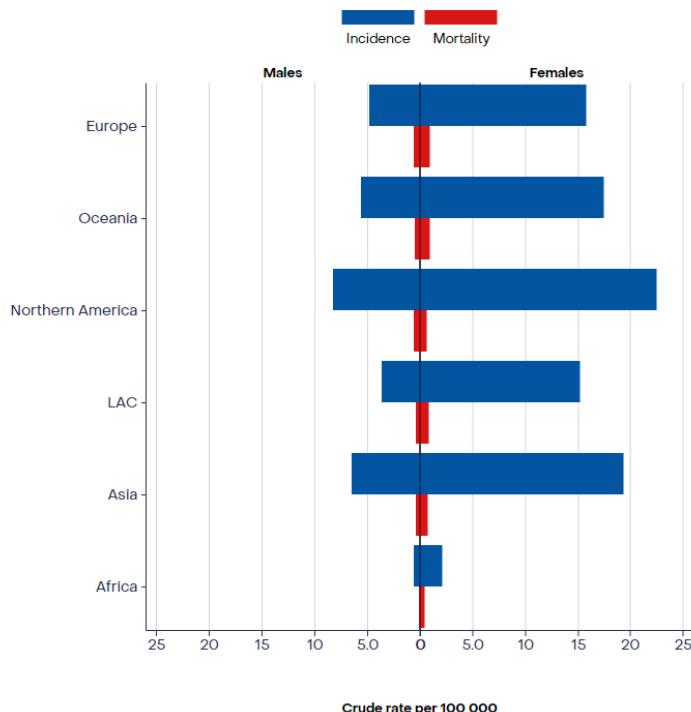
Fuente: Organización Mundial de la Salud (2022). *Cancer Today*.



**Figura 4.** Tasa bruta de mortalidad de cáncer de tiroides en varones por 100 000 personas.

Fuente: Organización Mundial de la Salud (2022). *Cancer Today*.

Con la Figura 5 que muestra los mismos índices distribuidos por género y regiones del mundo, es fácil notar la alta incidencia de este tipo de cáncer en las mujeres, siendo la región con mayor incidencia América del Norte, mientras que la de mayor mortalidad es Oceanía. La región de Latino América y el Caribe supera a Norte América en mortalidad, aunque se encuentra por debajo de Oceanía y Europa.



**Figura 5.** Tasa bruta de incidencia y mortalidad de cáncer de tiroides por género y región.

Fuente: Organización Mundial de la Salud (2022). *Cancer Today*.

Además, es importante mencionar que este aumento de incidencia a nivel global de cáncer de tiroides se debe a diversos factores relacionados a problemas de salud como la obesidad y a factores medioambientales como por ejemplo exposición al yodo (Kim et al., 2020). Sin embargo, otro factor es la poca disposición y capacidad de realizar un diagnóstico a tiempo de los nódulos tiroideos que, de forma general, cerca del 7 % al 15 % de los casos de llegan a ser cancerígenos (Haugen et al., 2016).

Para la detección temprana de este tipo de cáncer o desarrollo de tumores, depende en gran medida de la experiencia y la capacidad cognitiva de un experto en radiología, y muchos de estos se ven en la necesidad de utilizar no muy avanzados sistemas de pre-diagnóstico por computadora, mejor conocido como CAD por sus siglas en inglés (Zhu et al., 2021). Ante grandes limitaciones de sistemas CAD básicos, y aprovechando el extenso uso de la Inteligencia Artificial, el Deep Learning y sus algoritmos son capaces de incorporar mayor eficacia a dichos sistemas.

El Deep Learning ha sido usado ampliamente como herramienta para el procesamiento de imágenes médicas, no solo para detectar diferentes tipos de cáncer o nódulos como los relacionados a los pulmones, sino también para la retinopatía diabética y localización de feto en ecografías, e inclusive en la detección del COVID-19 (Bhattacharya et al., 2021). Además,

el aumento de la calidad de imágenes de ultrasonido que se fue desarrollando en los últimos 30 años, aumentando cada vez más resolución de las imágenes y reduciendo el tiempo de adquisición, ha permitido una mejora en términos de detección de enfermedad; sin embargo, aún se requiere de un médico especializado y debidamente entrenado para lograr un realizar un correcto análisis de las imágenes, por ello se vio en la necesidad de encontrar métodos de clasificación automatizada, pero dichos métodos antiguos consumían bastante tiempo, gran poder computacional y poca capacidad de generalización de resultados, es así que en este contexto, apareció una novedosa arquitectura de Deep Learning que actualmente es usado en diferentes áreas el día de hoy: las Redes Neuronales Convolucionales o CNN, quitando en gran medida los problemas de las antigua técnicas (Singh et al., 2020). Sin embargo, es importante mencionar que estos últimos años han ido apareciendo nuevas arquitecturas capaces incluso de superar a los ya altamente difundidos CNN. Y es que a pesar del dominio actual de estas redes, las arquitecturas basadas en Transformer han ido tomando cada vez más mayor importancia dentro del mundo de la Inteligencia Artificial, y esto incluye también el área de procesamiento de imágenes médicas gracias a los Vision Transformers o ViT, una versión de los Transformers originales orientados al procesamiento de imágenes.

El desarrollo de estas nuevas tecnologías basadas en distintos algoritmos de Inteligencia Artificial, ya sea en CNN, ViT o incluso ambos, podrían mejorar aún más los tiempos de pre-diagnóstico de nódulos tiroideos y, además, aumentar los porcentajes de aciertos de los especialistas en estas tareas de diagnóstico a través de un trabajo conjunto con estas tecnologías de alto potencial.

## 1.2 Formulación del Problema

Con el objetivo de formular los objetivos de esta investigación, se propusieron las siguientes preguntas.

### 1.2.1 Problema General

PG: ¿Es posible implementar un modelo de Deep Learning para el pre-diagnóstico de nódulos tiroideos a través de imágenes de ultrasonido?

### 1.2.2 Problemas Específicos

- PE1: ¿Qué características del conjunto de datos a usar para entrenar y evaluar los modelos son ideales para obtener buenos resultados?
- PE2: ¿Qué técnicas de preprocessamiento se deberían aplicar a las imágenes de ultrasonido del conjunto de datos?

- PE3: ¿De qué forma se puede evaluar el rendimiento de los modelos de Deep Learning en la clasificación de imágenes de ultrasonido?
- PE4: ¿Qué arquitecturas de Deep Learning tienen el más alto desempeño en la clasificación de imágenes de ultrasonido de la tiroides?

## 1.3 Objetivos de la Investigación

A continuación, se presentan el objetivo general y los objetivos específicos.

### 1.3.1 Objetivo General

OG: Diseñar un modelo de Deep Learning para el pre-diagnóstico de nódulos tiroideos a través de imágenes de ultrasonido.

### 1.3.2 Objetivos Específicos

- OE1: Determinar las características ideales del conjunto de datos a usar para entrenar y evaluar los modelos.
- OE2: Determinar las técnicas de preprocessamiento que se deben aplicar a las imágenes de ultrasonido del conjunto de datos.
- OE3: Identificar las métricas de evaluación de rendimiento de los modelos de Deep Learning en la clasificación de imágenes de ultrasonido.
- OE4: Determinar las arquitecturas de Deep Learning que tienen el más alto desempeño en la clasificación de imágenes de ultrasonido de la tiroides.

## 1.4 Justificación de la Investigación

### 1.4.1 Teórica

Esta investigación plantea el desarrollo un modelo de Deep Learning capaz de pre-diagnosticar nódulos de la tiroides mediante el uso de imágenes de ultrasonido. A través de este, se aborda el problema común de procesos de diagnóstico largos y, en gran porcentaje, imprecisos realizados por el personal especialista, ya que actualmente requieren de una extensa experiencia para lograr mejores resultados en este tipo de análisis de imágenes médicas. Además, la gran cantidad de investigaciones publicadas los últimos años sobre la Inteligencia Artificial han demostrado la alta capacidad de este tipo de tecnologías para desempeñarse en tareas como la detección, segmentación y clasificación enfocado en diversos campos. Específicamente el Deep Learning ha demostrado ser de las arquitecturas más preferidas para el

procesamiento de imágenes, especialmente aquellas basadas en Redes Neuronales Convolucionales, los cuales, actualmente, están dominando las tareas relacionadas a imágenes por su alto desempeño. Sin embargo, estos últimos años, gracias al gran avance y divulgación de los modelos basados en Transformers, se ha visto el surgimiento de una nueva clase de arquitecturas que prometen mejorar el desempeño de los actuales modelos dominantes: los Vision Transformer (ViT). Siguiendo esta nueva tendencia, y observando el poco desarrollo actual de modelos ViT para el pre-diagnóstico de nódulos tiroideos, la investigación desarrollará modelos basados en ViT, además de los ya conocidos CNN.

### 1.4.2 Práctica

La investigación concluirá con una arquitectura de Inteligencia Artificial basada en Deep Learning de alto desempeño capaz de realizar el pre-diagnóstico de nódulos en la glándula de la tiroides, es decir, podrá determinar si un nódulo es benigno o maligno. Esto permitirá agilizar el proceso de diagnóstico realizado por las personas especialistas. Los beneficiarios finales serán todos los pacientes que pasen por este proceso de diagnóstico. Además, esta investigación evidenciará la capacidad de los modelos basados en arquitecturas de Deep Learning para realizar tareas de predicción (benigno o maligno) en base a imágenes de ultrasonido de la glándula de la tiroides.

### 1.4.3 Metodológica

El desarrollo de un modelo de Deep Learning para el pre-diagnóstico de nódulos tiroideos permitirá acelerar el proceso de diagnóstico a través de imágenes de ultrasonido, lo cual permitirá que los pacientes obtengan sus resultados de forma más rápida y eficiente, posteriormente, esto favorecerá a que el personal especialista determine los tratamientos adecuados lo más antes posible, llegando así a mejorar el bienestar de los pacientes, puesto que con un tratamiento a tiempo de los nódulos se puede evitar un posible cáncer en la glándula de la tiroides.

## 1.5 Delimitación del Estudio

A continuación, se presentará la delimitación espacial, temporal y conceptual.

### 1.5.1 Espacial

Debido a que la problemática y necesidad de diagnosticar a tiempo los nódulos tiroideos no es propio de una región o país, la data recopilada para realizar un correcto análisis de Inteligencia Artificial son basados en países extranjeros y de gran alcance como China, del cual se obtuvo el conjunto de datos de imágenes de ultrasonido. Los proyectos previos más afines al

tema presentados en la investigación son desarrollados en diversos países extranjeros.

### **1.5.2 Temporal**

Los datos presentados en esta investigación sobre casos de incidencia y mortalidad de la tiroides son del año 2022. De igual forma, las imágenes de ultrasonido de glándulas de tiroides con presencia de nódulos que se usarán para entrenar y validar el modelo de clasificación son recopiladas del conjunto de datos de acceso libre TNCD publicada en el año 2022. Los antecedentes relacionados a esta investigación fueron publicados entre el 2019 y 2023.

### **1.5.3 Conceptual**

La presente investigación se centrará en el desarrollo de un modelo de Deep Learning para realizar la clasificación de nódulos de la glándula de tiroides y así lograr determinar si es de carácter benigno o maligno. Esto se logrará a través del uso de diferentes arquitecturas como los basados en Redes Neuronales Convolucionales y los Vision Transformers. Estos modelos se desempeñarán en la tarea de análisis de imágenes médicas, específicamente, en imágenes de ultrasonido de la tiroides. Además, se usarán técnicas propias de la Inteligencia Artificial como la Transferencia de Aprendizaje y el Aumento de Datos para obtener mejores resultados.

## Capítulo II: Marco Teórico

### 2.1 Antecedentes de la investigación

En esta parte de la investigación se presentan algunos antecedentes relacionados a la detección y pre-diagnóstico de nódulos en distintos órganos y a través de diversas metodologías. Estos ayudarán a entender el enfoque y obtener bases para un correcto desarrollo del proyecto en cuestión.

Moreira Aresta (2021) presenta una investigación relacionada a la elaboración de un sistema CAD (Computer-aided Detection) para la detección y segmentación de nódulos presentes en los pulmones.

Para realizar esta investigación, se tuvo en consideración la realidad que se atraviesa a nivel mundial sobre el cáncer de pulmón, y como este afecta a las personas sin importar su género, convirtiéndolo uno de los tipos de cáncer más mortales según la investigación mencionada.

La presencia de nuevas formas de detectarlo como el uso de tomografías computarizadas tuvo un gran impacto para la detección temprana de los nódulos en estos órganos, ya que estos pueden significar un futuro desarrollo de cáncer. Sin embargo, realizar un análisis de este tipo de imágenes médicas no era nada trivial debido al complejo proceso de análisis con estas tecnologías. Ante esto, el Deep Learning apareció con una herramienta eficaz para ayudar a los médicos en el análisis de estas imágenes; sin embargo, pese a sus grandes ventajas, su aplicabilidad es muy limitada. Por tal motivo, esta tesis incluye modernos enfoques y técnicas para una detección automática de nódulos pulmonares.

De forma general, el algoritmo presentado en esta investigación consiste en dos bloques. El primero de estos se encarga de la detección de nódulo a través de técnicas de detección de objetos, mientras que el segundo se encarga de la segmentación de este. Los resultados obtenidos fueron del 79 % en recall; sin embargo, esto no fue suficiente para poder considerar al modelo como robusto, por ello, posteriormente, se utilizaron técnicas innovadoras en colaboración expertos en el área. Las anotaciones de los especialistas, junto con los resultados del sistema, mejoraron el desempeño general de la detección.

Para aumentar aún más las capacidades del sistema, se desarrolló en la parte final de la investigación un modelo de Deep Learning para la detección automática y la segmentación de nódulos pulmonares a través de imágenes tomográficas. Este también consistió en dos bloques, uno encargado de la segmentación automática y otro que corrige el anterior en base a dos puntos en los límites del nódulo. El modelo consiguió demostrar su capacidad para corregir la

segmentación de nódulos pequeños, además de segmentar los nódulos no sólidos, los cuales presentaban un reto para el sistema.

Felgueiras Carvalho (2019) reafirman la importancia de construir este tipo de sistemas mencionando al cáncer de pulmón como el principal tipo de cáncer en causar muertes en todo el mundo.

El sistema desarrollado en esta investigación fue basado en el diagnóstico a través de imágenes tomográficas que comúnmente es realizado por un médico experto; sin embargo, se menciona que este análisis siempre está sujeto a errores ya que existe mucha subjetividad en este tipo de diagnóstico, lo que conlleva muchas veces errores de detección. Además de la existencia de la tendencia al rechazo hacia esto tipo de sistemas CADx (Computer-aided Diagnosis) por parte de los médicos, esto debido a la falta de comprensión del cómo se realiza este tipo de pre-diagnóstico.

Para afrontar esta desconfianza a este tipo de sistemas, se menciona la existencia del Deep Hierarchical Semantic Convolutional Neural Network (HSCNN) que añadía a la capacidad de predecir si un nódulo era maligno o benigno, la función de otorgar evidencia visual del pre-diagnóstico a través de la predicción de las características del nódulo. Sin embargo, el conjunto de datos usado en esta investigación difería de las evaluaciones de los propios médicos. Por tal motivo, en la presente investigación se puso como un objetivo probar si la disminución de esta varianza en los datos podría mejorar los resultados del modelo HSCNN.

A través de un análisis de los datos, se logró mejorar la descripción de características. Este nuevo conjunto de datos fue revisado por especialista para comprobar su validez antes de ser ingresado al modelo HSCNN para predecir si un nódulo era maligno o benigno. Este proceso se hizo a través de un k-folds igual a 4 junto con cross-validation.

Los resultados fueron comparados con el modelo inicial, y se obtuvo que el nuevo HSCNN era mejor solo cuando se trataba de predecir si un nódulo era maligno. Las métricas del nuevo modelo fueron de 0.78 en accuracy, el AUC de la curva ROC es de 0.74, una sensibilidad 0.83 y una especificidad de 0.89, frente a las métricas de modelo original que obtuvo un accuracy de 0.84, un AUC de la curva ROC de 0.86, sensibilidad de 0.71 y una especificidad de 0.89. Esto significó que el modelo se equivocaba más cuando los nódulos a analizar eran pequeños. En general, los resultados distaron de los propios del modelo inicial, esto es debido a la reducción considerable de imágenes del conjunto de datos original.

Supanta Zapata (2021) menciona la incidencia de cáncer de pulmón en el Perú en el periodo 2010-2012 con un gran número de 3 121 casos diagnosticados, convirtiéndolo en el tercer tipo de cáncer más común en el país, y ocupa el segundo lugar en mortalidad con un

número de 2 591 de muertes en el mismo periodo, esto debido al tardío diagnóstico de este mal. Por este motivo, es obvia la necesidad de realizar un examen de detección temprana donde es posible un tratamiento y, posteriormente, una posible cura a esta enfermedad.

La investigación presenta una forma de realizar detección de nódulos pulmonares a través de imágenes radiográficas digitales que comúnmente presentan problemas como baja claridad y resaltado de las características, por ende, generan dificultades para realizar un correcto análisis de este tipo de imágenes. Las técnicas usadas son corrección gamma, análisis de proyecciones, erosión, filtros geométricos, dilatación, filtro de convergencia y la umbralización por el método Otsu, esto aplicado a un conjunto de datos de 50 radiografías de tórax.

Los resultados muestran una sensibilidad del 0.91, especificidad de 0.96 y precisión de 0.94.

Otra investigación relacionada a detectar alguna enfermedad a través de técnicas de Deep Learning y visión por computadora es la presentada por Monroy Malca (2021) donde toma a la enfermedad de Parkinson como mal a detectar, esto a través del análisis de la escritura de una persona. Para tal objetivo, en la investigación se desarrolló un modelo de visión computacional para el pre-diagnóstico de esta enfermedad.

La metodología empieza con la adquisición y preprocessamiento de los datos, para posteriormente usar técnicas de extracción de características como SIFT, SURF, ORB y HOG. Estos serán ingresados a un modelo de Machine Learning (SVM, RF, KNN) que finalmente realizará una clasificación. Además, se usaron diferentes arquitecturas de Redes Neuronales Convolucionales como Inception, VGG16, VGG19, LeNet y ResNet50.

Los resultados finales fueron un 0.99 en accuracy, 0.99 en precisión, 0.99 en recall, 0.98 en F1-score y AUC.

Kang et al. (2022) muestran la frecuencia de nódulos tiroideos en los diagnósticos, estando presente en el rango de 19 % y 68 % de todos los casos clínicos recopilados de personas adultas. El cáncer de tiroides es el número 9 en incidencia, mientras que es el número 6 en mortalidad, esto a nivel mundial según datos del 2018.

Se menciona que es importante tener métodos menos invasivos para determinar el cáncer de tiroides, teniendo en cuenta además que es necesario la detección a tiempo, pues esto aumenta la probabilidad de ser curado. El análisis de imágenes de ultrasonido es una buena opción a el análisis de, por ejemplo, cirugías. Sin embargo, esta forma de diagnóstico depende mucho de la experiencia del médico especialista, lo cual vuelve a este tipo de análisis muy subjetivo.

La segmentación y clasificación son herramientas claves dentro de un sistema de diagnóstico asistido, siendo ambos muy relacionados entre sí, pues comparten ciertas características de las imágenes (bordes de las imágenes pueden ser usados para la clasificación y segmentación). Afrontar ambas tareas al mismo tiempo en un solo modelo podría ser más robusto.

La red MTL propuesta en uno de los antecedentes es similar a una red de segmentación multiclases que genera como salida mapa de segmentación y predicciones categóricas.

La investigación presentada en este antecedente se centra también en una red MTL que también realizará procesos de segmentación y clasificación.

La metodología empieza con la data. Esta fue recolectada del West China Hospital. Se tuvieron 4 493 imágenes de ultrasonido, cada uno de un paciente. En específico, 2 576 imágenes pertenecen a nódulos benignos, mientras que 1 917 son malignos. Las anotaciones fueron verificadas por especialistas.

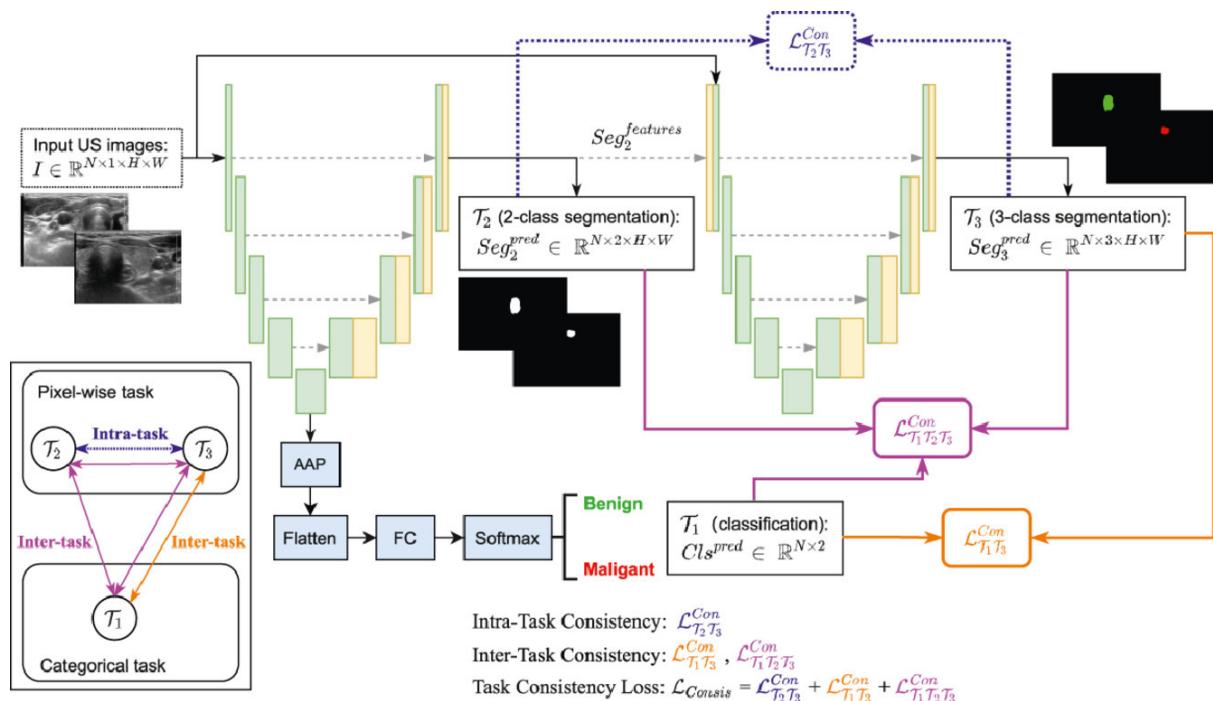
Para la evaluación de la clasificación se usó el accuracy, f1-score, ROC, área bajo la curva ROC (AUC). En el caso de la segmentación, se usó Dice coefficient y Intersection of Union (IoU).

Además, se definieron 3 medidas para cuantificar la inconsistencia al nivel de las tareas. El primero de estos evalúa a la clasificación, el segundo por la tarea de segmentación de segunda clase, y el tercero es para la segmentación de tercera clase.

El modelo construido en esta investigación se basó en MSL (Multi Stage Learning) y MTL (Multi Task Learning). Se diseño una red MS-MTL. Además, se determinó diferentes tipos de consistencia de tareas: intra e inter task consistency.

Los resultados fueron extraídos de 3 modelos CIsNet, MS-MTL y MS-MTL con intra e inter task consistency. La medida AUC obtenida para cada modelo es 0.9277, 0.9523 y 0.9608, respectivamente. Además, se determinó que el intra task consistency y el MSL aumenta el desempeño de la segmentación. El caso de inter task consistency con MTL mejora el desempeño de las tareas de clasificación y segmentación. Con la combinación de ambas tareas de consistencias también se demuestra su efectividad.

El método usado en esta investigación se muestra en la Figura 6.

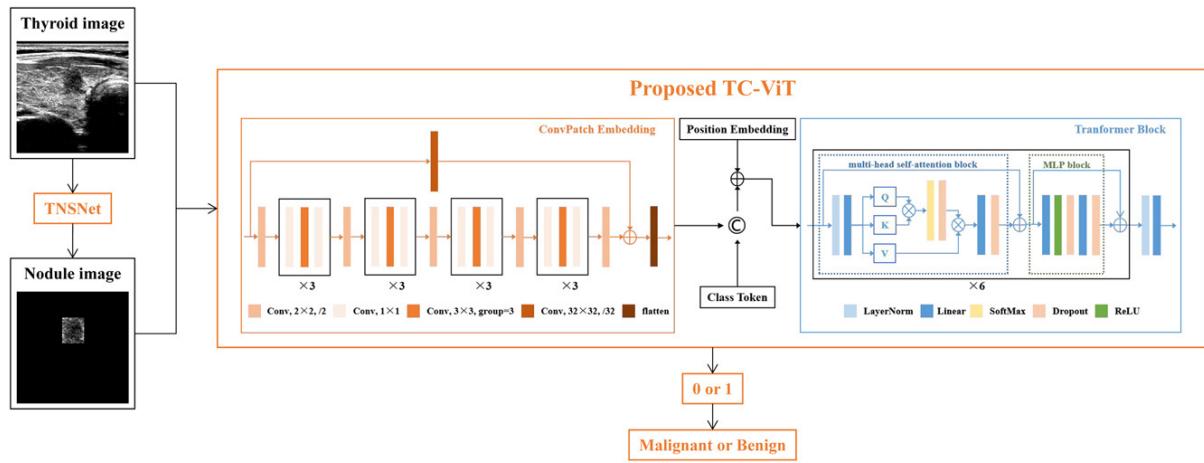


**Figura 6.** Representación del método desarrollado.

Fuente: Kang et al. (2022). *Thyroid nodule segmentation and classification in ultrasound images through intra- and inter-task consistent learning.*

Sun et al. (2023) presenta el problema de clasificar nódulos tiroideos nivel 3 debido a pocas características representativas que diferencien a los nódulos benignos de nivel 3 con los nódulos malignos, esto conlleva a obtener bajas precisiones en el pre-diagnóstico. Ante esto, en el artículo se presenta un modelo clasificador de nódulos tiroideos con ViT (Vision-Transformer-based) y el contrast learning. Estas técnicas ayudaron a minimizar la distancia de características en nódulos de una misma clase, lo cual mejora la capacidad predictiva del modelo. Finalmente, en la fase de testeo del modelo se logró un accuracy de 0.869, mientras que las demás métricas indican la superioridad frente a otros modelos clásicos de Deep Learning que también son usados para clasificar.

En la Figura 7 se presenta la arquitectura de Vison Transformer desarrollada.



**Figura 7.** Arquitectura de modelo TC-ViT.

Fuente: Sun et al. (2023). *Classification for thyroid nodule using ViT with contrastive learning in ultrasound images.*

Zhang et al. (2023) presenta el nivel de importancia de poseer una buena base de datos etiquetada correctamente para lograr entrenar un buen modelo de Machine Learning. En este artículo se desarrolla y presenta a la herramienta Multistep Automated Data Labelling Procedure (MADLaP) que facilita y automatiza el proceso de etiquetado de datos relacionados a nódulos tiroideos. Este incluye el procesamiento de lenguaje natural basado en reglas, el Deep Learning para segmentación de imágenes y el reconocimiento óptico de caracteres. MADLaP fue desarrollado en la fase de entrenamiento con datos de 378 pacientes, mientras que en la fase de prueba se usaron datos de 93. Este obtuvo finalmente un accuracy de 0.83.

Deng et al. (2022) ponen al cáncer de tiroides como el que más ha prevalecido en las últimas 3 décadas. Existen diversos sistemas que ayudan a la detección de los nódulos en esta glándula; sin embargo, muchos de estos solo se limitan a determinar si un nódulo es maligno o benigno, y no muestran el porqué de la toma de esa decisión por parte del sistema, lo cual genera desconfianza entre los especialistas al momento de usarlos. Para afrontar esto, se desarrolla primeramente una estratificación de riesgo basada en el léxico estandarizado ACR TI-RADS. Posteriormente, se realiza la clasificación entre benigno y maligno. De forma general, el método realizará una caracterización del nódulo basado en ACR TI-RADS para detectar su nivel de riesgo y la clase al que pertenece (benigno o maligno). Los resultados muestran en la evaluación un accuracy de 0.9355, un sensitivity de 0.9386 y una specificity de 0.9314.

Se realizó la notación de las imágenes de nódulos con los indicadores del diccionario de ACR TI-RADS. Para afrontar el desbalanceo de la data, se realizó un proceso de mejora de la data creando imágenes a través de giros, recortes y mezclas. Se extrajeron las áreas de interés

de las imágenes a través de una red en cascada. Se quitaron las anotaciones manuales en las imágenes (limpieza de imagen). Una vez concluido con el procesamiento de las imágenes, se realizó la construcción y ejecución del modelo de Deep Learning Multi-Task Learning (MLT). Finalmente, el modelo obtenido fue comparado con otros a través de los siguientes indicadores: accuracy, sensitivity, specificity y el área bajo la curva.

Wang et al. (2020) mencionan que los nódulos tiroideos son uno de los primeros síntomas que podrían conllevar a un cáncer en la tiroides. Este tipo de cáncer es uno de los que tiene mayor incidencia y que esta tendencia ha ido en crecimiento durando los últimos 30 años.

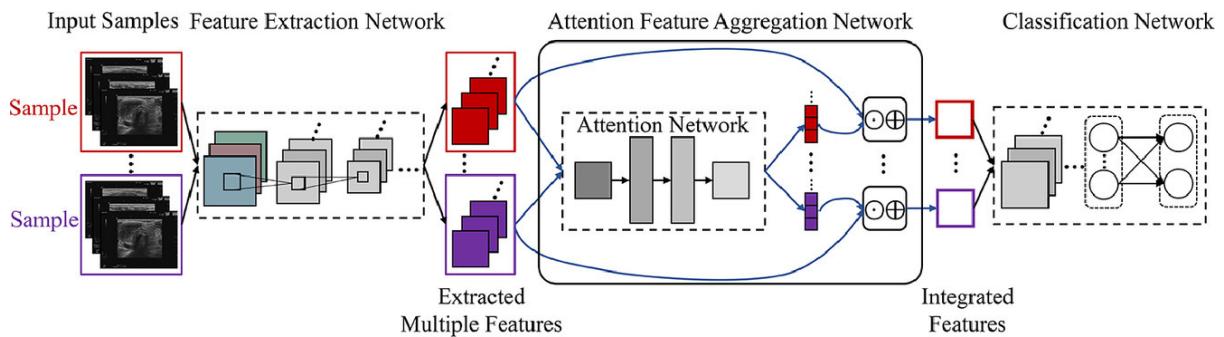
Además, se menciona que, para ayudar a esta detección, se han propuesto anteriormente varios sistemas de diagnóstico asistido; sin embargo, estos solo realizan dicho proceso a través de solo una imagen de ultrasonido en vez de usar todas aquellas que se obtienen de un examen. Por esto, en este artículo, se desarrolla un modelo de Deep Learning para el pre-diagnóstico de tiroides a través de varias imágenes de ultrasonido. Esto a través de una integración de todas las características de las imágenes realizadas en un examen.

La base de datos usada fue construida, y se obtuvieron resultados perfectamente comparables con los resultados de los antecedentes revisados en este artículo.

La metodología consistía en la construcción de tres redes distintas: feature extraction network, attention-based feature aggregation network, classification network. Estas tres redes juntas forman el modelo objetivo desarrollado en este artículo.

En general, la metodología consistía en, primero, la construcción del conjunto de datos que se conforma de imágenes de ultrasonido procedentes de un hospital. Se recolectaron cerca de 7 800 imágenes de 1 046 exámenes de entre los años 2015 y 2018. En segundo lugar, se realizó el etiquetado de los datos. Estas anotaciones se realizaron de acuerdo con los exámenes en general, y no a cada una de las imágenes que la conforman. Luego, se realizó la separación de la data en entrenamiento, prueba y validación. En la parte final de experimentación, se realizó un preprocesamiento con Data Augmentation y se definió 100 épocas para la fase de entrenamiento. El modelo ganador de la fase de validación fue probado con la data de prueba. Las métricas usadas para la evaluación fueron el accuracy, sensitivity (true positive rate) y el área bajo la curva (AUC ROC).

La Figura 8 muestra la arquitectura propuesta en esta investigación.



**Figura 8.** Arquitectura de la red desarrollada.

Fuente: Sun et al. (2023). *Automatic diagnosis for thyroid nodules in ultrasound images by deep neural networks.*

Sun et al. (2022) mencionan que, en el sistema endocrino, un problema muy común son los nódulos tiroideos. Normalmente estos pueden ser sólidos o de otras variadas texturas. La incidencia de esta enfermedad se ha ido incrementando a través de los años. Comúnmente se usan las imágenes de ultrasonido para detectar estos nódulos, esto lo hacen en tiempo real, tiene bajo costo y no es invasivo, por lo cual es una buena opción. Estas imágenes pueden otorgar información valiosa de los nódulos como los márgenes, ecogenicidad, calcificación, composición, etc. Además, existe el Thyroid Image Reporting y el sistema de datos TI-RADS que pueden cuantificar las características otorgadas por las imágenes de ultrasonido y evaluar si es benigno o maligno. Sin embargo, estas imágenes pueden traer problemas como bajo contraste y ruido de moteado, lo cual genera retos para extraer sus características. Este problema puede superarse a través de una correcta segmentación de los nódulos tiroideos. Así, la segmentación es un paso importante para realizar un pre-diagnóstico correcto. Esto puede ser aplicado a la evaluación automática en TI-RADS para facilitar la clasificación de los nódulos. Se añadió un shape-supervised path para mejorar la identificación de la forma, y así lograr una mejor segmentación.

La data que usaron se obtuvo de diversos escáneres disponibles comercialmente, juntando 3 786 imágenes de ultrasonido de nódulos tiroideos. Posteriormente, se realiza un pre-procesamiento para estandarizar la data. Se realizó un proceso de limpieza de las imágenes para quitar los datos privados de los pacientes, y borrar imágenes erróneas. Se propuso TNSNet para asegurar una mejor detección y segmentación. Este modelo es un dual-path network que contiene dos partes region path y shape path. Ambos caminos se enfocan en las características de bordes y texturas (cada uno de estos para cada camino).

Se diseñó un dual-path loss function para el entrenamiento de ambos caminos del modelo.

Para el camino de región, se usó el binary cross-entropy loss y el generalized Dice loss, ambos para entrenar la región de segmentación de los nódulos tiroideos. En el caso del segundo camino (camino de la forma) se combinó el Hausdoff distance loss y un modificado active contour loss. Esto fue construido para el entrenamiento del contorno de segmentación.

Las métricas usadas para evaluar la segmentación del modelo son Dice simialrity coefficient (DSC), sensitivity, specificity y el accuracy.

Los resultados muestran que en el caso de los nódulos benignos todos los modelos (el construido en el artículo y los de referencia) logran buenos resultados, esto debido a la buena calidad de las imágenes. En el caso de los nódulos malignos, el modelo TNSNet superó por completo a los otros modelos de referencia. Hubo otros experimentos, en los cuales, el modelo presentado en este artículo, de manera general, obtuvo mejores resultados.

La investigación realizada por Yang et al. (2023) se centra en la problemática del cáncer de piel, específicamente en el grave problema de los melanomas y la alta necesidad de una detección temprana para mejorar las posibilidades de supervivencia de los pacientes con este mal.

Además, se menciona que aunque la técnica más usada para la detección de melanomas es la dermatoscopia, su precisión es muy variada dependiendo el tipo de cáncer de piel con el que se está tratando. En este contexto, las técnicas de Deep Learning han demostrado ser un herramienta capaz de mejorar la precisión en las tareas relacionadas a clasificar el cáncer de piel, incluso superando al desempeño de los propios dermatólogos.

A pesar de estos logros del desarrollo de Deep Learning en esta área, se menciona que aún hay dificultad para la clasificación de algunos tipos de cáncer de piel como el melanoma. Por este motivo se propone una nueva arquitectura de red basada en los transformadores con el objetivo de mejorar la capacidad de los algoritmos de clasificación de cáncer de piel.

La metodología seguida por los autores inicia con la preparación de los datos. En este caso se usó el conjunto de datos HAM10000 que consta de 7 clases distintas de cáncer de piel. Las imágenes tuvieron que ser redimensionadas para los 4 distintos modelos a entrenar: Inception ResNet con Soft Attention (IRv2 + SA), ResNet50 con Soft Attention (ResNet50 + SA), ViTfSCD-Large y ViTfSCD-Base. Los parámetros de estos dos últimos modelos se muestran en la siguiente figura.

Además, se aplicó a las imágenes una limpieza, eliminando los duplicados. También pasó por un proceso de balanceo de las clases donde se intentó igualar la cantidad de imágenes en cada una de estas. Se realizó el proceso de Aumento de Datos a través de modificaciones en las imágenes como alterar la saturación de forma aleatoria, cambiar el contraste y el brillo.

Finalmente, se definió los porcentajes de división para el conjunto de datos, siendo 85 % para el entrenamiento y 15 % para la prueba.

Para la evaluación de los modelos se utilizaron las métricas de accuracy, precision, recall (sensitivity), specificity y F1 score. Los resultados de evaluar su capacidad de clasificación de los 4 modelos se muestran en la Tabla 1, Tabla 2, Tabla 3 y Tabla 4.

Tabla 1

*Comparación de resultados de los modelos (precision).*

Tipo de cáncer de piel	Precision			
	IRv2+SA	ResNet50+SA	ViTfSCD-B	ViTfSCD-L
AKIEC	0.81	0.67	0.82	0.66
BCC	0.95	0.90	1.00	0.89
BKL	0.73	0.72	0.81	0.86
DF	0.62	0.62	0.67	0.60
MEL	0.70	0.62	0.60	0.79
NV	0.95	0.96	0.94	0.97
VASC	0.91	0.83	0.83	1.00

Fuente: Yang et al. (2023). *A Novel Vision Transformer Model for Skin Cancer Classification.*

Tabla 2

*Comparación de resultados de los modelos (sensitivity).*

Tipo de cáncer de piel	Sensitivity			
	IRv2+SA	ResNet50+SA	ViTfSCD-B	ViTfSCD-L
AKIEC	0.57	0.52	0.61	0.83
BCC	0.73	0.69	0.54	0.65
BKL	0.79	0.70	0.65	0.76
DF	0.83	0.83	1.00	1.00
MEL	0.47	0.59	0.53	0.68
NV	0.97	0.98	0.98	0.99
VASC	1.00	1.00	1.00	1.00

Fuente: Yang et al. (2023). *A Novel Vision Transformer Model for Skin Cancer Classification.*

Tabla 3  
Comparación de resultados de los modelos (*f1-score*).

Tipo de cáncer de piel	F1-score			
	IRv2+SA	ResNet50+SA	ViTfSCD-B	ViTfSCD-L
AKIEC	0.67	0.59	0.70	0.73
BCC	0.83	0.78	0.70	0.76
BKL	0.76	0.71	0.72	0.81
DF	0.71	0.71	0.80	0.75
MEL	0.56	0.61	0.56	0.73
NV	0.96	0.97	0.96	0.98
VASC	0.95	0.91	0.91	1.00

Fuente: Yang et al. (2023). *A Novel Vision Transformer Model for Skin Cancer Classification*.

Tabla 4  
Comparación de resultados de los modelos (*specificity*).

Tipo de cáncer de piel	Specificity			
	IRv2+SA	ResNet50+SA	ViTfSCD-B	ViTfSCD-L
AKIEC	1.00	0.99	1.00	0.99
BCC	1.00	1.00	1.00	1.00
BKL	0.98	0.98	0.99	0.99
DF	1.00	1.00	1.00	1.00
MEL	0.99	0.99	0.99	0.99
NV	0.80	0.84	0.76	0.88
VASC	1.00	1.00	1.00	1.00

Fuente: Yang et al. (2023). *A Novel Vision Transformer Model for Skin Cancer Classification*.

De los resultados, se observó que el modelo ViTfSCD-Large obtuvo el más alto precision, recall, y F1 scores. Además, el mismo modelo obtuvo un accuracy de 94.1 %, siendo el valor más alto, mientras que el modelo ViTfSCD-Base obtuvo 91.4 %.

Finalmente, se realizó una comparación de los modelos a través de su accuracy. En esta última sección, se comparó con otros modelos del estado de arte desarrollados con el conjunto de datos HAM10000. El primero de estos, implementado en 2020 y denominado EfficientNet logró un accuracy de 92.6 %. También, en el año 2021, se desarrolló un método semi supervisado para la clasificación de imágenes médicas que obtuvo un accuracy de 92.5 %.

Para esta comparación final, también se consideraron los resultados originales de los modelos IRv2, IRv2 + SA, ResNet50 y ResNet50 + SA junto con los resultados obtenidos en esta investigación. Además, también se realizaron experimentos con los modelos ViT originales (ViT-Base 16 y ViT-Large 16). Los resultados finales presentados en la Tabla 5 muestran que los modelos ViTfSCD-Base y ViTfSCD-Large son superiores a sus versiones originales.

Tabla 5

*Comparación de resultados según accuracy con el conjunto de datos HAM10000.*

Métodos	Año	Accuracy general en sus paper	Accuracy general en propios experimentos
Loss balancing y ensemble	2020	92.6 %	-
Semi-supervised	2021	92.54 %	-
ResNet50	2021	90.5 %	-
IRv2	2021	91.2 %	-
ResNet50 con Soft Attention	2021	91.5 %	91.5 %
Inception ResNet con Soft Attention	2021	93.4 %	91.9 %
Vision Transformer (ViT)-Base16	2021	-	91.1 %
ViT-Large16	2021	-	93.7 %
ViT para detección de cáncer de piel-Base	2022	-	91.4 %
ViT para detección de cáncer de piel-Largo	2022	-	94.1 %

Fuente: Yang et al. (2023). *A Novel Vision Transformer Model for Skin Cancer Classification.*

También se realizaron experimentos con el conjunto de datos DERMOFIT que consta de imágenes de lesiones clínicas de la piel divididas en 10 clases distintas. Los resultados originales con este conjunto de datos muestran que el modelo ResNet50 tuvo mejor desempeño que el modelo de árbol de decisión y KNN.

Los nuevos experimentos con este conjunto de datos se llevaron a cabo con los modelos ResNet50, IRv2 y ViTfSCD-Base. Los resultados presentados en la Tabla 6 muestran el desempeño superior del modelo ViTfSCD frente a los demás.

Tabla 6

*Comparación de resultados según accuracy con el conjunto de datos DERMOFIT.*

Métodos	Año	Accuracy general en sus paper	Accuracy general en propios experimentos
Decision Tree	2020	78.1 %	-
Flat ResNet50	2020	78.7 %	-
ResNet50	2022	-	71.2 %
Inception ResNet (IRv2)	2022	-	71.9 %
IRv2 + Soft Attention	2022	-	75.0 %
Vision Transformer para la detección de cáncer de piel-Base	2021	-	80.5 %

Fuente: Yang et al. (2023). *A Novel Vision Transformer Model for Skin Cancer Classification.*

La investigación presentada por Ayana et al. (2023) se enfoca en el cáncer más común de las mujeres en los Estados Unidos: el cáncer de mama. Se dice que la tasa de incidencia de este mal ha ido en aumento en 0.5 % cada año; sin embargo, la tasa de mortalidad ha caído a un

45 % considerando datos desde el año 1989 hasta el 2020, esto debido a las mejoras constantes de los tratamientos y a las detecciones a tiempo, siendo la mamografía una técnica crucial en este último.

A pesar del importante papel de las mamografías en la detección a tiempo del cáncer de mama, existen casos de diagnósticos erróneos que muchas veces conllevan a procedimientos y gastos innecesarios para un paciente.

Ante este problema, se dice que se han desarrollado diversos sistema de detección asistida por computadora o CAD por sus siglas en inglés que tiene el principal objetivo de reducir los errores en la detección de este cáncer.

Existen varios tipos de sistema CAD. Algunos no usan la Inteligencia Artificial, mientras otros utilizan las técnicas altamente difundidas del Deep Learning. Específicamente los modelos basados en las Redes Neuronales Convolucionales (CNN) han sido consideradas como prometedoras para las tareas detección de tumores de mama, llevándolos incluso a ser capaces de reducir la probabilidad de error humano. Sin embargo, esta clase de modelos también tienen sus dificultades.

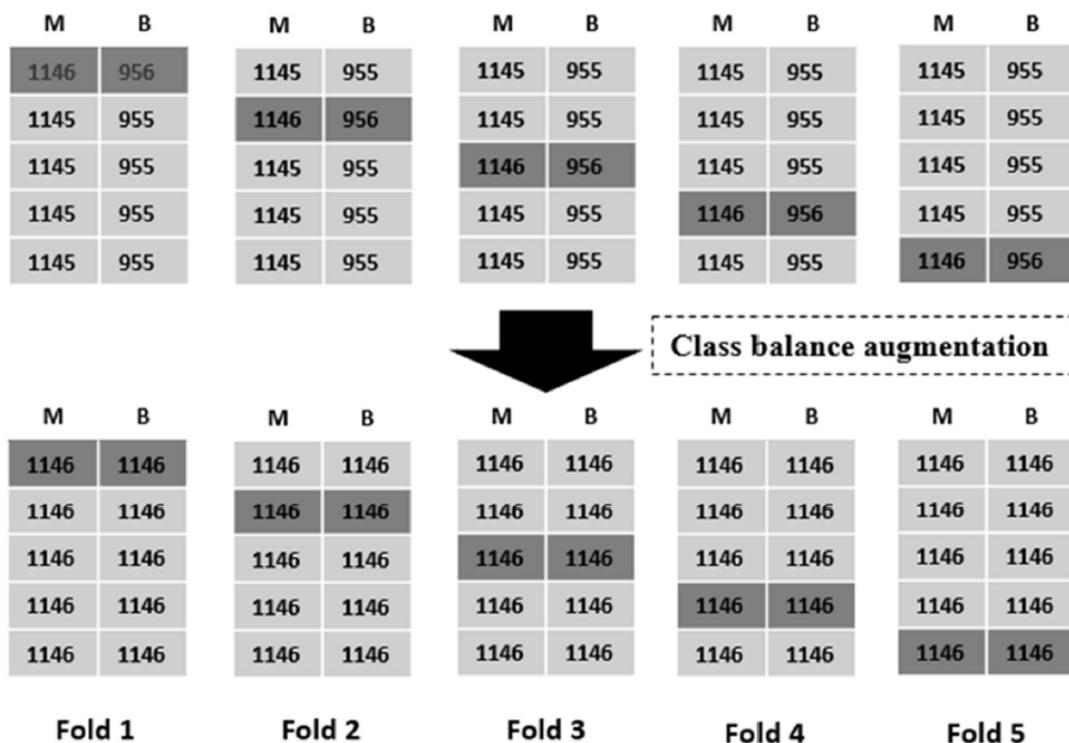
Los modelos CNN son computacionalmente caros debido a la complejidad y cantidad de convoluciones que se realizan. Además, en el contexto del cáncer de mama, estos modelos tienen dificultades en la localización de tumores, y una alta necesidad en el pre procesamiento, pues es necesario mejorar la calidad y reducir el ruido de las imágenes.

Otro problema a considerar es la falta de conjuntos de datos para el análisis de imágenes médicas. Para abordar esto, es necesario aplicar Transfer Learning y el Aumento de Datos.

En esta investigación se desarrolló un modelo de Deep Learning basado en los Vision Transformers y el Transfer Learning para la detección de cáncer de mama a través de mamografías. Para lograr esto, primero se abordó el problema de desbalanceo de las dos clases (benigno y maligno) en el conjunto de datos de mamografías, posteriormente, se desarrolló un método de Transfer Learning basado en los Vision Transformer con el objetivo de mejorar las deficiencias de los métodos de Transfer Learning basados en CNN.

El conjunto de datos usado para entrenar y probar el modelo fue el Digital Database for Screening Mammography (DDSM). Este consiste (hasta el último acceso del 12 de septiembre del 2022) de 13 128 imágenes mamográficas de las cuales 5 970 son benignas y 7 158 son malignas. Esto presentaba un claro desbalance con un ratio de 0.65:0.35. Esta distribución entre las clases podría generar errores en la etapa de aprendizaje del modelo, por este motivo, se desarrolló un nuevo método para el balanceo de datos a través de la técnica de Aumento de Datos, específicamente, se aplicó variaciones en la fluctuaciones de color de las imágenes,

corrección gamma, giro horizontal, sal y pimientas, y afilado o sharpening. El objetivo fue balancear el conjunto de datos para realizar validación cruzada de 5 pliegues, para lograr esto, se dividió al conjunto de datos en 5 partes o pliegues. Para las primeras 4 partes, se colocaron 1 145 imágenes de tumores malignos en cada una, mientras que para el quinto pliegue se tenían 1 146. Además, en los primeros 4 pliegues se tuvieron 955 imágenes de tumores benignos, a la vez que en el quinto pliegue se tuvieron 956. Para el lograr el balanceo de clases, se sometió a imágenes de tumores benignos al proceso de aumento de imágenes 5 veces, mientras que para al grupo de imágenes de tumores malignos solo se aplicó una vez. Así, finalmente se obtuvo 1 146 imágenes para ambas clases presentes en cada pliegue. Este proceso se muestra en la Figura 9.



**Figura 9.** Balanceo de clases usando Aumento de Datos.

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification*.

Una vez obtenida la data ya balanceada, se realizó el preprocesamiento a las imágenes, específicamente, se redimensionó a las imágenes a 244 x 244 pixeles, esto debido a que el tamaño de las imágenes del conjunto de datos era variable.

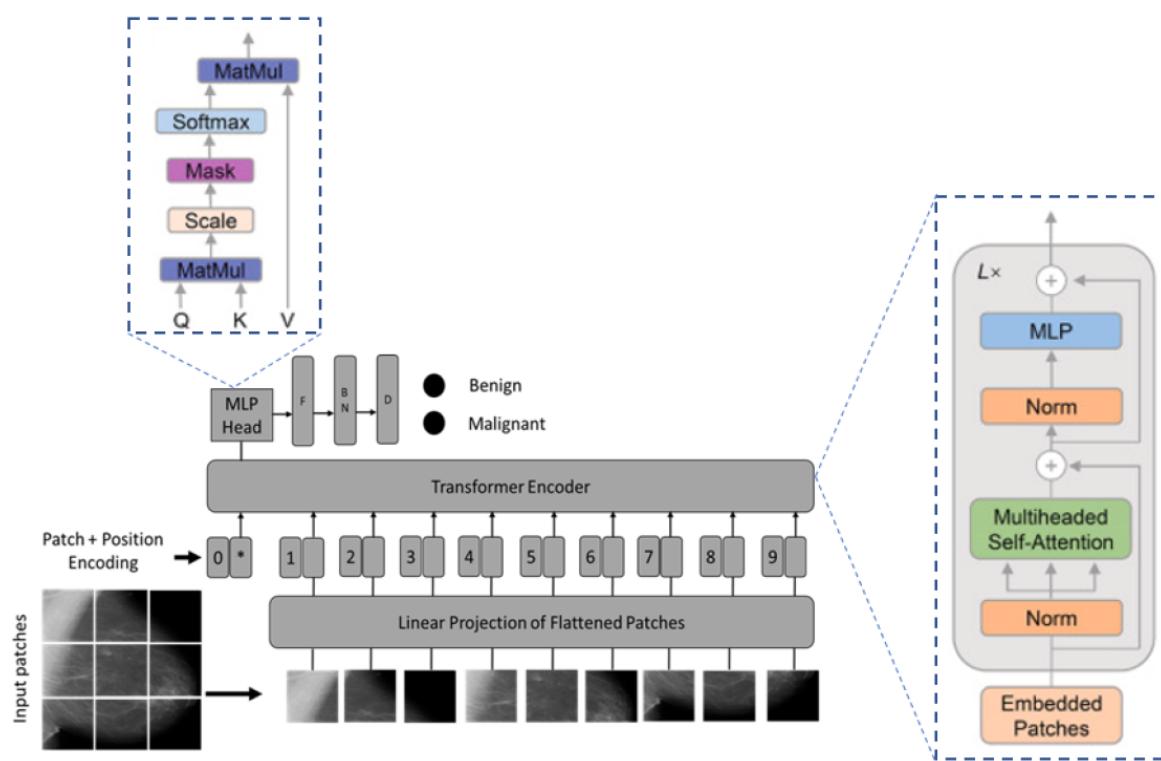
El diseño del modelo consiste inicialmente de la arquitectura de Vision Transformer. Se explica que siendo un modelo derivado de los convencionales transformer usados para el

proceso del lenguaje natural o NLP donde las entradas son de una sola dimensión, los Vision Transformer se adaptaron para recibir entradas de dos dimensiones (imágenes).

Específicamente, el modelo se encarga de dividir las entradas en pequeñas partes, también de dos dimensiones, conocidas como patches que finalmente servirán como entrada y serán pasada como word tokens, de igual forma que en un modelo transformer NLP. Aunque antes de proveer estos patches, se realizan los procesos de flattening, sequence imbedding, learnable embedding y patch embedding.

Además del bloque transformer, se tiene una parte de MLP encargada de la tarea de clasificación. Este posee una sola capa oculta y utiliza la función de activación GELU.

En el Figura 10 se muestra la estructura del modelo implementado en la investigación.



**Figura 10.** Estructura de ViT implementado.

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification.*

El Transfer Learning fue aplicado en la investigación a través del uso de modelos de Vision Transformer pre-entrenados con el extenso conjunto de datos ImageNet, y posteriormente utilizados para entrenar el conjunto de datos de imágenes mamográficas. Es decir que se aprovechó el conocimiento de dichos modelos para la tarea final de clasificación de mamografías en

benigno y maligno. Los modelos usados para este proceso fueron el Vision Transformer (ViT), Swin Transformer (Swin-T) y el Pyramid Vision Transformer (PVT).

El modelo Swin-T posee cuatro distintas versiones: Swin-base, Swin-tiny, Swin-small y Swin-large. Las variantes usadas en esta investigación fueron el Swin-small y Swin-base. Mientras que de las 4 versiones de PVT (PVT-tiny, PVT-small, PVT-medium y PVT-large), se usaron solo el PVT-medium y PVT-large.

Los modelos fueron entrenados con 50 épocas, una tasa de aprendizaje de 0.0001, optimizador Adam y un tamaño de lote de 64. Además, se dividió el conjunto de datos en 80 % para entrenamiento y 20 % para prueba.

Los Vision Transformer usaron GELU como función de activación, mientras que los CNN usaron ReLu. Ambos usaron el regularizador L2.

También se aplicó validación cruzada de 5 para lograr una mejor comparación en el rendimiento de los modelos.

Las métricas usadas para el proceso de evaluación de los modelos fueron el accuracy, AUC, F1-score, precision, recall, el coeficiente de correlación de Matthew y el kappa score.

Se evaluó el método propuesto a través de 5 formas distintas: comparación de desempeño de los 3 modelos con Transfer Learning, comparación de los modelos Vision Transformer entrenados desde cero con el conjunto de datos de mamografías, comparación de Transfer Learning de Vision Transformer con los CNN, comparación de costo computacional de cada modelo Vision Transformer y, finalmente, se comparó el desempeño de los métodos desarrollados en esta investigación con otros que usaron el mismo conjunto de datos.

Como se muestra en la Tabla 7, Tabla 8 y Tabla 9, los 6 modelos Transfer Learning basados en arquitectura de Vision Transformer y entrenados con el conjunto de datos DDSM tuvieron un desempeño uniforme a través de todas la métricas.

Tabla 7

*Resultado de los modelo ViT basados en Transfer Learning en el conjunto de datos DDSM (accuracy, AUC, f1-score).*

Arquitectura	Modelo	Accuracy (95 %)	AUC (95 %)	F1 Score (95 %)
Vision	ViT-Base	1 ± 0	1 ± 0	1 ± 0
Transformer	ViT-Large	1 ± 0	1 ± 0	1 ± 0
Swin Transformer	Swin-Small	1 ± 0	1 ± 0	1 ± 0
	Swim-Base	1 ± 0	1 ± 0	1 ± 0
Pyramid Vision	PVT-Medium	1 ± 0	1 ± 0	1 ± 0
Transformer	PVT-Large	1 ± 0	1 ± 0	1 ± 0

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification.*

Tabla 8

*Resultado de los modelo ViT basados en Transfer Learning en el conjunto de datos DDSM (precision, recall, MCC).*

Arquitectura	Modelo	Precision (95 %)	Recall (95 %)	MCC (95 %)
Vision	ViT-Base	1 ± 0	1 ± 0	1 ± 0
Transformer	ViT-Large	1 ± 0	1 ± 0	1 ± 0
Swin Transformer	Swin-Small	1 ± 0	1 ± 0	1 ± 0
	Swim-Base	1 ± 0	1 ± 0	1 ± 0
Pyramid Vision	PVT-Medium	1 ± 0	1 ± 0	1 ± 0
Transformer	PVT-Large	1 ± 0	1 ± 0	1 ± 0

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification.*

Tabla 9

*Resultado de los modelo ViT basados en Transfer Learning en el conjunto de datos DDSM (kappa).*

Arquitectura	Modelo	Kappa (95 %)
Vision	ViT-Base	1 ± 0
Transformer	ViT-Large	1 ± 0
Swin Transformer	Swin-Small	1 ± 0
	Swim-Base	1 ± 0
Pyramid Vision	PVT-Medium	1 ± 0
Transformer	PVT-Large	1 ± 0

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification.*

La Tabla 10, Tabla 11 y Tabla 12 muestran los resultados de los modelos Vision Transformer implementados desde cero y entrenados con el conjunto de datos DDSM.

Tabla 10

*Resultado de los modelo ViT entrenados desde cero en el conjunto de datos DDSM (accuracy, AUC, f1-score).*

Arquitectura	Modelo	Accuracy (95 %)	AUC (95 %)	F1 Score (95 %)
Vision	ViT-Base	0.74 ± 0.02	0.73 ± 0.03	0.74 ± 0.01
Transformer	ViT-Large	0.72 ± 0.04	0.72 ± 0.02	0.72 ± 0.03
Swin Transformer	Swin-Small	0.75 ± 0.02	0.75 ± 0.03	0.75 ± 0.01
	Swim-Base	0.76 ± 0.01	0.75 ± 0.02	0.75 ± 0.02
Pyramid Vision	PVT-Medium	0.78 ± 0.02	0.77 ± 0.02	0.78 ± 0.01
Transformer	PVT-Large	0.77 ± 0.03	0.77 ± 0.01	0.77 ± 0.02

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification.*

Tabla 11

*Resultado de los modelo ViT entrenados desde cero en el conjunto de datos DDSM (precision, recall, MCC).*

Arquitectura	Modelo	Precision (95 %)	Recall (95 %)	MCC (95 %)
Vision	ViT-Base	0.74 ± 0.01	0.74 ± 0.01	0.73 ± 0.03
Transformer	ViT-Large	0.72 ± 0.04	0.72 ± 0.03	0.71 ± 0.02
Swin Transformer	Swin-Small	0.75 ± 0.02	0.75 ± 0.02	0.74 ± 0.03
	Swim-Base	0.75 ± 0.01	0.76 ± 0.01	0.75 ± 0.01
Pyramid Vision	PVT-Medium	0.78 ± 0.02	0.78 ± 0.02	0.77 ± 0.01
Transformer	PVT-Large	0.77 ± 0.02	0.77 ± 0.02	0.77 ± 0.01

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification.*

Tabla 12

*Resultado de los modelo ViT entrenados desde cero en el conjunto de datos DDSM (kappa).*

Arquitectura	Modelo	Kappa (95 %)
Vision	ViT-Base	0.73 ± 0.02
Transformer	ViT-Large	0.72 ± 0.01
Swin Transformer	Swin-Small	0.74 ± 0.02
	Swim-Base	0.75 ± 0.02
Pyramid Vision	PVT-Medium	0.77 ± 0.02
Transformer	PVT-Large	0.77 ± 0.01

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification.*

Se puede observar que el modelo PVT-medium obtuvo un mejor desempeño comparado a los demás modelos. Sin embargo, estos resultados se mantienen muy por debajo de los presentados anteriormente.

Finalmente, también se obtuvieron resultados de los experimentos de Transfer Learning con modelos CNN y el conjunto de datos DDSM. Estos se muestran en la Tabla 13, Tabla 14 y Tabla 15.

Tabla 13

*Resultado de los modelos CNN basados en Transfer Learning en el conjunto de datos DDSM (accuracy, AUC, f1-score).*

Arquitectura	Modelo	Accuracy (95 %)	AUC (95 %)	F1 Score (95 %)
ResNet	ResNet50	0.95 ± 0.01	0.96 ± 0.01	0.95 ± 0.01
	ResNet101	0.95 ± 0.01	0.95 ± 0.02	0.95 ± 0.01
EfficientNet	EfficientNetB0	0.94 ± 0.02	0.94 ± 0.01	0.94 ± 0.01
	EfficientNetB2	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01
InceptionNet	InceptionNetV2	0.93 ± 0.02	0.93 ± 0.01	0.93 ± 0.02
	InceptionNetV3	0.94 ± 0.01	0.94 ± 0.01	0.94 ± 0.01

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification.*

Tabla 14

*Resultado de los modelos CNN basados en Transfer Learning en el conjunto de datos DDSM (precision, recall, MCC).*

Arquitectura	Modelo	Precision (95 %)	Recall (95 %)	MCC (95 %)
ResNet	ResNet50	0.95 ± 0.01	0.95 ± 0.01	0.94 ± 0.01
	ResNet101	0.95 ± 0.01	0.95 ± 0.01	0.94 ± 0.02
EfficientNet	EfficientNetB0	0.94 ± 0.01	0.94 ± 0.01	0.93 ± 0.03
	EfficientNetB2	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01
InceptionNet	InceptionNetV2	0.93 ± 0.02	0.93 ± 0.02	0.93 ± 0.03
	InceptionNetV3	0.94 ± 0.01	0.94 ± 0.01	0.93 ± 0.02

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification.*

Tabla 15

*Resultado de los modelos CNN basados en Transfer Learning en el conjunto de datos DDSM (kappa).*

Arquitectura	Modelo	Kappa (95 %)
ResNet	ResNet50	0.94 ± 0.02
	ResNet101	0.94 ± 0.02
EfficientNet	EfficientNetB0	0.93 ± 0.02
	EfficientNetB2	0.95 ± 0.01
InceptionNet	InceptionNetV2	0.92 ± 0.02
	InceptionNetV3	0.93 ± 0.02

Fuente: Ayana et al. (2023). *Vision-Transformer-Based Transfer Learning for Mammogram Classification.*

El desempeño de los modelos CNN también se encuentran por debajo de los Vision Transformer presentados inicialmente.

Manzari et al. (2023) nos menciona que dentro del análisis de imágenes médicas, la tarea de clasificación es crucial. Por ello, este se ha visto beneficiado constantemente de novedosos sistemas de diagnóstico asistido, siendo los de mayor desempeño aquellos basados en Redes Neuronales Convolucionales (CNN), pues ofrecen predicciones cada vez más precisas incluso comparándose con los médicos. Sin embargo, estos también tienen algunas desventajas o dificultades, principalmente lo relacionado a su capacidad de aprendizaje de dependencias a largo alcance en conjuntos de datos visuales, esto debido al conocido sesgo local de las CNN.

En este contexto, las arquitecturas basadas en transformers y su mecanismo de auto-atención, han demostrado ser capaces de solucionar estas desventajas de las CNN. Esto debido a su capacidad de modelar las dependencias a largo alcance de los datos, convirtiéndolos así en modelos superiores a las arquitectura de CNN, aunque con mayor necesidad en cantidad de datos de entrenamiento, recursos computacionales y tiempo.

A pesar de los beneficios que pueden traer las arquitecturas transformer frente a los CNN, estos están muy limitadas en su uso en las tareas de análisis de imágenes médicas. Esto es debido principalmente a sus desventajas ya mencionadas (alta necesidad de recursos computacionales y tiempo), pues en el entorno clínico, la eficiencia y la velocidad es de suma importancia para realizar pre-diagnósticos satisfactorios.

Los altos requerimientos de los transformer traen consigo grandes dificultades, impiadiendo que estos puedan ser usados en situaciones clínicas reales. Además, también existe el problema de la suposición de una distribución similar en los datos que se van a ingresar al modelo; sin embargo, se sabe que esto no siempre se cumple en el entorno real, debido a que las imágenes médicas son capturadas a través de distintos dispositivos, diversos protocolos y diferentes lugares. Esta discrepancia en los datos puede generar que el rendimiento del modelo caiga considerablemente.

Ante este problema, la investigación propone un nueva arquitectura basada en transformer con capacidad de generalización de distintos tipos de imágenes médicas como las tomografías computarizadas, rayos X y ultrasonido. Su estructura se caracteriza por ser híbrida jerárquica, poseer patch embedding y poseer convoluciones y bloques de transformer. Además, se incluyó en la arquitectura del transformer un bloque de atención convolucional multi-cabezal con el fin de mejorar la capacidad de aprendizaje de representación local, incluso este permitió reducir la complejidad computacional comparado con un modelo transformer convencional.

Comparándolo con los CNN, el modelo desarrollado también demostró una capacidad

superior en la tarea de predicción, al mismo tiempo que poseía menor complejidad.

De forma generar el modelo MedViT, propuesto en esta investigación, tiene como objetivo lograr una arquitectura híbrida capaz de desempeñarse en la tarea de clasificación de imágenes médicas, a través de la combinación de bloques de convolución y transformer, además de estar también compuesto por capas de path embedding, siguiendo así una tradicional arquitectura piramidal jerárquica.

Los bloques principales de modelo son el Efficient Convolution Block (ECB) y el Local Transformer Block (LTB). Estos se encargan de capturar las dependencias de los datos de entrada a largo y corto plazo.

También se propuso una nueva técnica de Aumento de Datos denominada Patch Momentum Charge (PMC) con el objetivo de mejorar el rendimiento del modelo.

Los experimentos se realizaron con 12 distintos conjuntos de datos de la categoría imágenes médicas. Todos estos se encontraron en MedMNIST, y conforman imágenes de tomografías computarizadas, rayos X, ultrasonido y tomografías de coherencia óptica (OCT). Esta variedad de datos permiten el entrenamiento de modelos en tareas de clasificación.

De forma específica, los conjuntos de datos usados fueron el PathMNIST (posee 100 000 image patches categorizadas en 9 clases distintas relacionadas a patologías en el colon), ChestMNIST (posee 112 120 imágenes frontales de rayos X de 32 717 pacientes dividida en 14 clases distintas de enfermedades de tórax), DermaMNIST (posee 10 015 imágenes dermatoscópicas recopiladas de distintas fuentes y divididas en 7 clases relacionadas a enfermedades de la piel), OCTMNIST (consta de 109 309 imágenes de OCT recopiladas de pacientes con enfermedades de la retina y dividida en 4 clases distintas), PneumoniaMNIST (recopila 5 856 imágenes radiográficas pediátricas de tórax categorizadas en 2 clases que indican presencia o no de neumonía), RetinaMNIST (provee 1 600 imágenes de fondo de retina de 628 pacientes y se divide en 5 clases que indican la gravedad de la enfermedad de retinopatía diabética), TissueMNIST (dividida en 8 clases, este conjunto de datos está compuesto de 236 386 imágenes segmentadas de tejidos de corteza renal), BloodMNIST (consta de 17 092 imágenes de células sanguíneas categorizadas en 8 clases), BreastMNIST (categorizada en las clases de benigno, maligno y normal, este conjunto de datos posee 780 ecografías mamarias) y OrganMNIST (consta de imágenes de tomografías computarizadas abdominales divididas en 11 clases en sus 3 versiones: Axial, Coronal y Sagital).

En la etapa de entrenamiento del modelo con los 12 conjuntos de datos descritos anteriormente, se siguieron los mismos ajustes de entrenamiento de MedMNISTv2. Se usaron 100 épocas con un tamaño de lote de 128 y se aplicó el redimensionado a las imágenes para

obtener la forma de 224 x 224 pixeles. Además, se usó el optimizar AdamW con una tasa de aprendizaje de 0.0001.

MedViT consiste en 3 distintas arquitecturas definidas por su tamaño: MedViT-T, MedViT-S y MedViT-L. Estas fueron entrenadas de forma separada con cada uno de los conjuntos de datos. También se incluyó el Aumento de Datos a través de PMC.en la etapa de entrenamiento.

Las métricas usadas para evaluar las 3 versiones del modelo MedViT fueron el accuracy y la curva ROC. Esta elección fue debido a que el experimento se llevó a cabo en varios conjuntos de datos distintos entre sí.

De los resultados mostrados en la Tabla 16, Tabla 17, Tabla 18 y Tabla 19, los modelos MedViT superan, de forma general, en gran medida a los modelos basados en CNN en la misma tarea. Esto lleva a concluir que el modelo demuestra una alta capacidad de generalización en imágenes médicas.

Tabla 16

*Comparación de resultados de los modelos MedViT con los CNN y AutoML (PathMNIST, ChestMNIST y DermaMNIST).*

Métodos	PathMNIST		ChestMNIST		DermaMNIST	
	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18 (28)	0.983	0.907	0.768	0.947	0.917	0.735
RestNet-18 (224)	0.989	0.909	0.773	0.947	0.920	0.754
ResNet-50 (28)	0.990	0.911	0.769	0.947	0.913	0.735
RestNet-50 (224)	0.989	0.892	0.773	0.948	0.912	0.731
auto-sklearn	0.934	0.716	0.649	0.779	0.902	0.719
AutoKeras	0.959	0.834	0.742	0.937	0.915	0.749
Google AutoML	0.944	0.728	0.778	0.948	0.914	0.768
MedViT-T (224)	0.994	0.938	0.786	0.956	0.914	0.768
MedViT-S (224)	0.993	0.942	0.791	0.954	0.937	0.780
MedViT-L (224)	0.984	0.933	0.805	0.959	0.920	0.773

Fuente: Manzari et al. (2023). *MedViT: A robust vision transformer for generalized medical image classification.*

Tabla 17

*Comparación de resultados de los modelos MedViT con los CNN y AutoML (OCTMNIST, PneumoniaMNIST y RetinaMNIST).*

Métodos	OCTMNIST		PneumoniaMNIST		RetinaMNIST	
	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18 (28)	0.943	0.743	0.944	0.854	0.717	0.524
RestNet-18 (224)	0.958	0.763	0.956	0.864	0.710	0.493
ResNet-50 (28)	0.952	0.762	0.948	0.954	0.726	0.528
RestNet-50 (224)	0.958	0.776	0.962	0.884	0.716	0.511
auto-sklearn	0.887	0.601	0.942	0.855	0.690	0.515
AutoKeras	0.955	0.763	0.947	0.878	0.719	0.503
Google AutoML	0.963	0.771	0.991	0.946	0.750	0.531
MedViT-T (224)	0.961	0.767	0.993	0.949	0.752	0.534
MedViT-S (224)	0.960	0.982	0.995	0.961	0.773	0.561
MedViT-L (224)	0.945	0.761	0.991	0.921	0.754	0.552

Fuente: Manzari et al. (2023). *MedViT: A robust vision transformer for generalized medical image classification.*

Tabla 18

*Comparación de resultados de los modelos MedViT con los CNN y AutoML (BreastMNIST, BloodMNIST y TissueMNIST).*

Métodos	BreastMNIST		BloodMNIST		TissueMNIST	
	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18 (28)	0.901	0.863	0.998	0.958	0.930	0.676
RestNet-18 (224)	0.891	0.833	0.998	0.963	0.933	0.681
ResNet-50 (28)	0.857	0.812	0.997	0.956	0.931	0.680
RestNet-50 (224)	0.866	0.842	0.997	0.950	0.932	0.680
auto-sklearn	0.836	0.803	0.984	0.878	0.828	0.532
AutoKeras	0.871	0.831	0.998	0.961	0.941	0.703
Google AutoML	0.919	0.861	0.998	0.966	0.924	0.673
MedViT-T (224)	0.934	0.896	0.996	0.950	0.943	0.703
MedViT-S (224)	0.938	0.897	0.997	0.951	0.952	0.731
MedViT-L (224)	0.929	0.883	0.996	0.954	0.935	0.699

Fuente: Manzari et al. (2023). *MedViT: A robust vision transformer for generalized medical image classification.*

Tabla 19

*Comparación de resultados de los modelos MedViT con los CNN y AutoML (OrganMNIST, OrganCMNIST y OrganSMNIST).*

Métodos	OrganMNIST		OrganCMNIST		OrganSMNIST	
	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18 (28)	0.997	0.935	0.992	0.900	0.972	0.782
RestNet-18 (224)	0.998	0.951	0.994	0.920	0.974	0.778
ResNet-50 (28)	0.997	0.935	0.992	0.905	0.972	0.770
RestNet-50 (224)	0.998	0.947	0.993	0.911	0.975	0.785
auto-sklearn	0.963	0.762	0.976	0.829	0.845	0.672
AutoKeras	0.994	0.905	0.990	0.879	0.974	0.813
Google AutoML	0.990	0.886	0.988	0.877	0.964	0.749
MedViT-T (224)	0.995	0.931	0.991	0.901	0.972	0.789
MedViT-S (224)	0.996	0.928	0.993	0.916	0.987	0.805
MedViT-L (224)	0.997	0.943	0.994	0.922	0.973	0.806

Fuente: Manzari et al. (2023). *MedViT: A robust vision transformer for generalized medical image classification.*

Regmi et al. (2023) nos menciona la gran importancia que tiene el análisis de imágenes médicas para la detección a tiempo de enfermedades potencialmente mortales. Los radiólogos y personal clínico realizan esta importante tarea en la gran mayoría de los casos; sin embargo, sus interpretaciones de las imágenes no siempre están en lo correcto debido a que su análisis muchas veces está condicionado por el mismo observador, llevando a que los niveles de precisión para interpretar este tipo de datos no sean tan altos.

Esta tarea de interpretación de imágenes médicas es aun más importante en los casos de una alta necesidad de detectar a tiempo alguna enfermedad; por ejemplo, las relacionadas a los pulmones o las enfermedades gastrointestinales que pueden no causar daño como el reflujo ácido u otras de grave impacto como el cáncer de colon. Una detección a tiempo de estos permite aplicar los tratamientos debidos y así, consecuentemente, mejorar la calidad de vida de los pacientes. En este contexto, los sistemas de diagnóstico asistido por computadora (CAD por sus siglas en inglés) han demostrado ser de gran ayuda para los médicos y expertos clínicos en la tarea de realizar diagnósticos tempranos.

La capacidad de las Redes Neuronales Convolucionales (CNN) de poder adaptarse en detectar distintas características de un conjunto de datos han permitido su desarrollo en el campo médico, específicamente en las tareas de segmentación, clasificación, registro y reconstrucción de imágenes médicas.

Otra técnica popular que va tomando cada vez más importancia son los Vision Transformer (ViT) capaces también de desempeñarse satisfactoriamente en las tareas de clasificación de imágenes, a pesar de ser basados en los originales transformers especializados en

tareas relacionadas a texto.

Otro tipo de arquitectura basada en transformer son los Data-Efficient Image Transformer (DeiT), una versión de nuevos de los ViT caracterizados por su baja dependencia a los datos a través de su enfoque maestro-estudiante.

En esta investigación se presentan distintas arquitecturas basadas en CNN y Transformer para la clasificación de radiografías de tórax en distintos tipos de enfermedades relacionadas. Específicamente, se usaron los modelos DeiT, ViT y Ensemble, este último basado en otros modelos CNN. También se aplicó Transfer Learning junto a modelos CNN previamente entrenados. Finalmente todos los modelos fueron comparados en la misma tarea de clasificación de imágenes.

Se usaron distintas versiones de ViT (ViT-B/16, ViT-L/16, ViT-L/32) pre-entrenados con el conjunto de datos ImageNet-21k.

La diferencia principal entre ViT-L/16 y ViT-L/32 radica en el tamaño de los patch de entrada que permiten, siendo para el primer caso un patch de tamaño 16x16, mientras que el segundo un patch de 32 x 32. Todos estos modelos pasaron por un proceso de fine-tuning con 3 conjuntos de datos.

El primer conjunto de datos usado fue el Chest X-ray dataset que consistía en 7 135 imágenes, divididas en 4 clases (COVID-19, neumonía, tuberculosis y normal).

El conjunto de datos Kvasir consiste de 1 000 imágenes en cada una de las 8 clases existentes.

El último conjunto de datos, Kvasir-Capsule dataset consiste de 44 228 imágenes de endoscopía divididas en 13 clases. La cantidad de imágenes varía en cada clase del conjunto de datos, lo cual lo vuelve altamente imbalanceado.

En la Tabla 20 se resumen las características de los 3 conjuntos de datos.

Tabla 20  
*Resumen de características de los conjuntos de datos.*

Dataset	Nº de Imágenes	Tamaño de entrada	Entren.	Val.	Prue.	Aplicación
Chest X-ray dataset	7135	variable	5693	671	771	Enfermedad pulmonar
Kvasir dataset	8000	720 x 556	6400	800	800	Endoscopía
Kvasir-Capsule	44152	336 x 336	19280	4820	23061	Enfermedad intestinal

Fuente: Regmi et al. (2023). *Vision transformer for efficient chest X-ray and gastrointestinal image classification.*

El modelo ViT base (ViT-B) está compuesto por 12 bloques transformer con módulos de autoatención de 12 cabezas, permitiendo un total de 86 millones de parámetros entrenables.

El modelo ViT large (ViT-L) consiste de 24 bloques transformer con módulos de autoatención de 16 cabezas, formando así 307 millones de parámetros entrenables.

Las imágenes del conjunto de datos Chest X-ray fueron redimensionadas a 224 x 224 pixeles. Esto también se aplicó al conjunto de datos Kvasir.

En el caso de las imágenes de Kvasir-Capsule, estas tuvieron que ser redimensionadas a 64 x 64 pixeles.

Se usaron dos modelos de DeiT: DeiT-Ti y DeiT-B 384. El primero de estos permitía imágenes de 224 x 224 de resolución; mientras que el segundo, 384 x 384.

Todos los conjuntos de datos fueron divididos en data de entrenamiento, validación y prueba.

Además, se aplicaron distintas técnicas de Aumento de Datos para cada uno de los conjuntos de datos. En el caso de Chest X-ray se aplicó el reescalado, cambio de brillo, desplazamiento vertical y horizontal, y zoom de forma aleatoria. Para el conjunto de datos Kvasir, se alteró el brillo de las imágenes, se rotaron y se dieron vuelta de manera vertical. Finalmente, para Kvasir-Capsule, se aplicó rotación, se dio vuelta a las imágenes de manera horizontal y vertical.

Los modelos CNN pre-entrenados que se usaron fueron el DenseNet201, DenseNet121, InceptionRestNetV2, Xception, MovileNetV2 y el Ensemble que combinaba DenseNet201 y DenseNet121.

Se probaron varios hiper-parámetros a través de prueba y error. Además, en el caso del conjunto de datos de Chest X-ray y Kvasir, se usó la función de pérdida categorical cross-

entropy. Con Kvasir-Capsule, se usó el Focal Loss debido al desbalance de los datos en cada clase.

Las métricas usadas para evaluar la capacidad de clasificación de los modelos fueron el Matthews Correlation Coefficient (MCC), Frames Per Second (FPS), precision, F1-score, recall y accuracy.

El precision utilizado fue la media ponderada, esto pues es un método que asigna pesos de acuerdo a la cantidad de observaciones en cada clase, volviéndolo ideal para conjuntos de datos desbalanceados. Además, también se obtiene la desviación estándar del Recall, F1-score y Precision con el fin de verificar la variabilidad de los resultados.

Se realizó la prueba t por pares para comparar los resultados con la métrica MCC del modelo con mejor rendimiento.

Los resultados de los experimentos se muestran en la Tabla 21, Tabla 22 y Tabla 23.

Tabla 21

*Resultados con el conjunto de datos Chest X-ray.*

Método	Precision	Recall	F1-score	Acc.	MCC	P-val.	FPS
DenseNet201 +	0.9345 ± 0.0324	0.9326 ± 0.0584	0.9319 ± 0.0236	0.9326	0.8931	1.24e-03	16.08
DenseNet121	0.9169 ± 0.0358	0.9157 ± 0.0538	0.9150 ± 0.0195	0.9157	0.8658	8.41e-05	22.47
DenseNet201	0.9314 ± 0.0358	0.9287 ± 0.0678	0.9277 ± 0.0370	0.9287	0.8878	3.37e-04	26.15
DenseNet121	0.9415 ± 0.0358	0.9403 ± 0.0678	0.9400 ± 0.0370	0.9287	0.9052	1.18e-04	20.48
Incep.RestNetV2	0.9380 ± 0.0324	0.9377 ± 0.0461	0.9375 ± 0.0358	0.9377	0.9010	4.80e-05	21.34
Xception	0.9133 ± 0.0625	0.9092 ± 0.0508	0.9099 ± 0.0401	0.9092	0.8567	8.96e-04	21.85
MobileNetV2	0.9266 ± 0.0453	0.9261 ± 0.0326	0.9262 ± 0.0368	0.9261	0.8821	3.37e-04	25.84
DeiT-Ti	0.9332 ± 0.0592	0.9300 ± 0.0728	0.9291 ± 0.0272	0.9300	0.8899	1.10e-03	16.75
DeiT-B 384	0.9527 ± 0.0362	0.9520 ± 0.0301	0.9521 ± 0.0277	0.9520	0.9236	3.83e-02	11.49
ViT-L/32	0.9533 ± 0.0202	0.9533 ± 0.0301	0.9532 ± 0.0225	0.9533	0.9259	-	11.20
ViT-L/16	0.9291 ± 0.0592	0.9248 ± 0.0728	0.9248 ± 0.0272	0.9248	0.8813	3.77e-04	20.74
ViT-B/16							

Fuente: Regmi et al. (2023). *Vision transformer for efficient chest X-ray and gastrointestinal image classification.*

Tabla 22

*Resultados con el conjunto de datos Kvasir.*

Método	Precision	Recall	F1-score	Acc.	MCC	P-val.	FPS
DenseNet201 + DenseNet121	0.9284 ± 0.0559	0.9263 ± 0.0711	0.9258 ± 0.0538	0.9263	0.9161	2.60e-01	12.32
DenseNet201	0.9262 ± 0.0467	0.9250 ± 0.0628	0.9246 ± 0.0472	0.9250	0.9146	1.52e-01	21.59
DenseNet121	0.9136 ± 0.0523	0.9112 ± 0.0749	0.9105 ± 0.0515	0.9113	0.8991	6.63e-02	24.35
Incep.RestNetV2	0.8877 ± 0.0619	0.8875 ± 0.0753	0.8870 ± 0.0648	0.8875	0.8716	1.90e-02	21.18
Xception	0.9032 ± 0.0611	0.9025 ± 0.0761	0.9020 ± 0.0639	0.9025	0.8888	4.64e-02	43.12
MobileNetV2	0.8789 ± 0.0689	0.8775 ± 0.0826	0.8769 ± 0.0691	0.8775	0.8603	1.42e-02	43.97
DeiT-Ti	0.9353 ± 0.0337	0.9350 ± 0.0394	0.9349 ± 0.0338	0.9350	0.9258	5.82e-01	25.80
DeiT-B 384	0.9408 ± 0.0401	0.9401 ± 0.0466	0.9399 ± 0.0370	0.9401	0.9319	3.73e-01	15.81
ViT-L/32	0.9375 ± 0.0532	0.9337 ± 0.0698	0.9333 ± 0.0458	0.9337	0.9249	4.55e-01	13.35
ViT-L/16	0.9454 ± 0.0400	0.9437 ± 0.0487	0.9436 ± 0.0345	0.9437	0.9360	-	11.34
ViT-B/16	0.9433 ± 0.0427	0.9400 ± 0.0532	0.9398 ± 0.0260	0.9400	0.9316	5.53e-01	21.50

Fuente: Regmi et al. (2023). *Vision transformer for efficient chest X-ray and gastrointestinal image classification.*

Tabla 23  
*Resultados con el conjunto de datos Kvasir-Capsule.*

Método	Precision	Recall	F1-score	Acc.	MCC	P-val.	FPS
DenseNet201 + DenseNet121	0.6633 ± 0.2485	0.7230 ± 0.2859	0.6737 ± 0.2594	0.7230	0.3560	9.63e-03	578.15
DenseNet201	0.6606 ± 0.2522	0.7198 ± 0.2820	0.6737 ± 0.2583	0.7198	0.3585	8.88e-03	1142.90
DenseNet121	0.6564 ± 0.2660	0.7187 ± 0.2848	0.6720 ± 0.2621	0.7187	0.3500	4.28e-03	1143.35
Incep.RestNetV2	0.6059 ± 0.2647	0.6900 ± 0.2742	0.6548 ± 0.0336	0.6203	0.2277	3.06e-04	544.81
Xception	0.6159 ± 0.2484	0.6895 ± 0.2710	0.6310 ± 0.2431	0.6895	0.2544	3.34e-04	1223.44
MobileNetV2	0.5903 ± 0.2645	0.6800 ± 0.2732	0.5925 ± 0.2342	0.6800	0.1637	7.041e-04	3555.12
DeiT-Ti	0.6100 ± 0.2603	0.6839 ± 0.2669	0.6203 ± 0.2431	0.6839	0.2212	5.06e-05	281.24
DeiT-B 384	0.6496 ± 0.3020	0.7180 ± 0.2770	0.6657 ± 0.3010	0.7185	0.3422	6.10e-04	520.21
ViT-L/32	0.6483 ± 0.2922	0.7182 ± 0.2985	0.6631 ± 0.2760	0.7182	0.3377	3.11e-02	343.60
ViT-L/16	0.6425 ± 0.2806	0.6751 ± 0.2725	0.6405 ± 0.2581	0.6751	0.2637	1.69e-03	262.09
ViT-B/16	0.6841 ± 0.2985	0.7156 ± 0.2899	0.7156 ± 0.2779	0.7156	0.3705	-	570.53

Fuente: Regmi et al. (2023). *Vision transformer for efficient chest X-ray and gastrointestinal image classification.*

Tampu et al. (2023) nos menciona que las tomografías de coherencia óptica (OCT) son un tipo de técnica de imágenes médicas que permiten la visualización de la microestructura de tejidos. Este ha sido altamente utilizado en distintas situaciones de análisis de imágenes médicas; por ejemplo, para las imágenes de arterias coronarias, el diagnóstico de cáncer, la gastroenterología y la odontología.

Debido a la capacidad de proporcionar imágenes de alta calidad, esta técnica es usada para la toma de decisiones en situaciones quirúrgicas a través de su implementación con dispositivos. Por ejemplo, esto es de gran ayuda en los casos de cirugías en la tiroides donde se necesita distinguir entre distintos tipos de tejidos y no dañar partes críticas del organismo, reduciendo así la morbilidad de los pacientes.

A pesar de estas grandes ventajas y aplicaciones de las imágenes OCT, su uso e interpretabilidad está limitado a un pequeño número de profesionales clínicos. Por este motivo, se han ido desarrollando distintos métodos de análisis de imágenes médicas con el fin de facilitar su interpretación; sin embargo, estos están comúnmente basados en las técnicas tradicionales

de procesamiento de imágenes que poseen dos grandes limitaciones: procesamiento lento y necesidad de intervención manual.

En este contexto, se han desarrollado nuevos métodos para sobrelevar las desventajas de los OCT. Los más conocidos son los basados en las Redes Neuronales Convolucionales (CNN) que han ido demostrando los últimos años su capacidad de obtener un alto rendimiento en tareas de clasificación, segmentación y detección de objetivos en imágenes.

Existe una gran variedad de arquitecturas de Deep Learning basadas en CNN desarrolladas para la clasificación de imágenes. Las más conocidas son ResNet, AlexNet e Inception. Todas entrenadas con el conjunto de datos ImageNet que consta de imágenes naturales. Sin embargo, las imágenes médicas tienen un comportamiento distinto, y es que estas tienen toda la información dispersa en la imagen y no solo enfocada en una región. Además, se ha ido experimentando con redes poco profundas con este tipo de imágenes y los resultados muestran un desempeño similar a las grandes pre-entrenadas arquitecturas.

Otro tipo de redes, aparte de los CNN, desarrolladas en el campo del análisis de imágenes médicas son los Capsule Networks y Transformer.

Las redes de Deep Learning se han desarrollado en distintos campos de la medicina para la clasificación de tejidos enfermos a través de imágenes dadas por la técnica OCT. Esto ha llevado a que se tengan buenos desempeños en esta tarea, superando en la mayoría de los casos el 85 % en precisión.

El objetivo de esta investigación fue el desarrollo, evaluación y explicación de distintos modelos de Deep Learning en la tarea de clasificación de tejidos en las categorías de normal y enfermo en dos tipos de conjuntos de datos (2D y 3D OCT). Para lograr esto, se evaluaron los modelos actualmente disponibles para tareas con OCT, se implementó un Vision Transformer y se desarrollaron 2 modelos basados en CNN.

Para esta investigación se tuvieron dos tipos de conjuntos de datos.

El primer tipo, OCT 2D, consistía inicialmente de 66 988 2D b-scan (imágenes obtenidas a través de la técnica OCT) de la tiroides. Sin embargo, para una mejor investigación del impacto de resolución anisotrópica de los pixeles, se crearon dos conjuntos de datos. El primero estaba compuesto de imágenes anisotrópicas, mientras que el segundo se construyó a base de aplicar un remuestreo de las imágenes a una resolución isotrópica. Ambos conjuntos de datos tuvieron que ser recortadas a 200 x 200 píxeles. Finalmente, se seleccionó el modelo LightOCT y la tarea de clasificación de densidad folicular (alta o baja) para la evaluación del impacto de este cambio en las imágenes.

Para la creación de conjunto de datos 3D, se tuvo el reto de superar las limitaciones de la capacidad de memoria del hardware gráfico, y es que el volumen original de los datos era demasiado grande para que los modelos pueden entrenar. De este modo, se realizó un proceso de extracción de 15 b-scan no consecutivos. Además, se pasó por un proceso de remuestreo para que tuvieran dimensiones isotrópicas y se aplicó un recorte para obtener un tamaño de 200 x 200 píxeles.

En la etapa de división de datos en entrenamiento y prueba, se empleó la técnica de división por volume/subject con el fin de evitar la evaluación errónea de los modelos debido al tipo de imágenes de OCT que comúnmente generan filtración de datos entre los conjuntos de entrenamiento y prueba.

Todas las tareas de clasificación tuvieron un conjunto de datos de prueba compuesta por 1000 imágenes 2D en cada una de las clases.

El conjunto de datos 3D fue dividido en 250 muestras para cada clase en la tarea de clasificación por enfermedad.

Los experimentos se desarrollaron con 5 distintos tipos de clasificación. El primer tipo de clasificación fue la distinción entre tejidos normales y anormales. El segundo tuvo el objetivo de categorizar según la estructura folicular (normal, agrandada o encogida). El tercero consistía en clasificar según la densidad folicular (alta o baja densidad). El cuarto tipo de clasificación determinaba si una estructura folicular era normal o agrandada. El último tuvo el objetivo de clasificar las muestras por enfermedad (6 clases).

Además de los mencionados anteriormente, también se evaluó el rendimiento de los modelos con dos conjuntos de datos de OCT de acceso abierto: Kermany versión 2 de oftalmología y AIIMS de tejidos mamarios cancerígenos. Esto con el objetivo de demostrar la capacidad de generalización de los modelos desarrollados.

Los modelos entrenados desde cero con los conjuntos de datos 2D fueron RestNet50, LightOCT, Custom Mk-CNN (multikernel CNN), Custom sResNet4 (shallow-ResNet) y Vision Transformer. En el caso del conjunto de datos 3D, se usaron dos modelos: 3D-LightOCT y 3D-ViT.

Para el entrenamiento de los modelos se tuvieron distintas configuraciones. El modelo LightOCT se optimizó a través del algoritmo de descenso por gradiente estocástico con el objetivo de reducir la pérdida definida por el categorical cross-entropy. Los demás modelos usaron la función de pérdida weighted categorical cross-entropy con el optimizador Lookahead y el optimizador interno ADAM. Además, también se calcularon los pesos para el balanceo de clases del conjunto de datos de entrenamiento.

El entrenamiento de los modelos 2D fue con la técnica de validación cruzada 5-folds con 250 épocas. Además, para el conjunto de datos de estos modelos, se aplicó Aumento de Datos sobre la marcha (giro horizontal y vertical aleatorios, rotación aleatoria de 30 grados y zoom aleatorio de 10 %).

Para evaluar el rendimiento de los modelos se usó la matriz de confusión multiclas. También se calculó el Matthew's Correlation Coefficient (MCC), accuracy, precision, recall, F1-score y AUC para cada modelo.

En la Tabla 24, Tabla 25, Tabla 26 y Tabla 27 se muestran los resultados finales.

Tabla 24

*Resultados de los modelos entrenados con el conjunto de datos 2D (MCC, accuracy y f1-score).*

Data 2D	Modelo	MCC	Accuracy	F1-scpre
Task 1 normal vs abnormal (2 clases)	LightOCT	0.23 ± 0.07	0.58 ± 0.03	0.49 ± 0.05
	ResNet50	0.31 ± 0.09	0.61 ± 0.06	0.54 ± 0.09
	Mk-CNN	0.04 ± 0.09	0.51 ± 0.02	0.40 ± 0.04
	sResNet4	0.45 ± 0.07	0.68 ± 0.03	0.66 ± 0.04
Task 2 Basado en estructura (3 clases)	ViT	0.37 ± 0.06	0.68 ± 0.03	0.68 ± 0.03
	LightOCT	0.36 ± 0.10	0.57 ± 0.07	0.55 ± 0.07
	ResNet50	0.48 ± 0.11	0.64 ± 0.08	0.64 ± 0.08
	Mk-CNN	0.47 ± 0.13	0.62 ± 0.10	0.55 ± 0.14
Task 5 Por enfermedad (6 clases)	sResNet4	0.47 ± 0.06	0.63 ± 0.04	0.62 ± 0.04
	ViT	0.41 ± 0.05	0.58 ± 0.03	0.54 ± 0.04
	LightOCT	0.33 ± 0.08	0.43 ± 0.07	0.42 ± 0.06
	ResNet50	0.31 ± 0.67	0.41 ± 0.05	0.39 ± 0.07
	Mk-CNN	0.40 ± 0.03	0.47 ± 0.02	0.45 ± 0.04
	sResNet4	0.34 ± 0.08	0.43 ± 0.07	0.37 ± 0.08
	ViT	0.38 ± 0.08	0.47 ± 0.07	0.45 ± 0.06

Fuente: Tampu et al. (2023). *Diseased thyroid tissue classification in OCT images using deep learning: Towards surgical decision support.*

Tabla 25

*Resultados de los modelos entrenados con el conjunto de datos 2D (AUC, precision y recall).*

Data 2D	Modelo	AUC	Precision	Recall
Task 1 normal vs abnormal (2 clases)	LightOCT	0.74±0.09	0.72±0.09	0.58±0.03
	ResNet50	0.75±0.10	0.74±0.10	0.61±0.06
	Mk-CNN	0.70±0.15	0.70±0.12	0.51±0.02
	sResNet4	0.80±0.04	0.77±0.04	0.68±0.03
	ViT	0.74±0.02	0.73±0.02	0.68±0.03
	LightOCT	0.76±0.05	0.61±0.08	0.57±0.07
Task 2 Basado en estructura (3 clases)	ResNet50	0.84±0.06	0.74±0.08	0.64±0.08
	Mk-CNN	0.83±0.04	0.68±0.06	0.62±0.10
	sResNet4	0.82±0.04	0.70±0.06	0.63±0.04
	ViT	0.78±0.02	0.59±0.03	0.58±0.03
	LightOCT	0.81±0.05	0.50±0.08	0.43±0.07
	ResNet50	0.79±0.06	0.46±0.09	0.41±0.05
Task 5 Por enfermedad (6 clases)	Mk-CNN	0.86±0.03	0.59±0.05	0.47±0.02
	sResNet4	0.83±0.07	0.51±0.10	0.43±0.07
	ViT	0.82±0.06	0.53±0.08	0.47±0.07

Fuente: Tampu et al. (2023). *Diseased thyroid tissue classification in OCT images using deep learning: Towards surgical decision support.*

Tabla 26

*Resultados de los modelos entrenados con el conjunto de datos 3D (MCC, accuracy y f1-score).*

Data 3D	Modelo	MCC	Accuracy	F1-score
Normal vs abnormal (2 clases)	LightOCT	0.26±0.07	0.58±0.03	0.49±0.05
	ViT	0.37±0.06	0.68±0.03	0.68±0.03
Basado en estructura (3 clases)	LightOCT	0.36±0.10	0.57±0.07	0.55±0.07
	ViT	0.41±0.05	0.58±0.03	0.54±0.04
Por enfermedad (6 clases)	LightOCT	0.33±0.08	0.43±0.07	0.42±0.06
	ViT	0.38±0.08	0.47±0.07	0.45±0.06

Fuente: Tampu et al. (2023). *Diseased thyroid tissue classification in OCT images using deep learning: Towards surgical decision support.*

Tabla 27

*Resultados de los modelos entrenados con el conjunto de datos 3D (AUC, precision y recall).*

Data 3D	Modelo	AUC	Precision	Recall
Normal vs abnormal (2 clases)	LightOCT	0.74±0.09	0.72±0.09	0.58±0.03
	ViT	0.74±0.02	0.73±0.02	0.68±0.03
Basado en estructura (3 clases)	LightOCT	0.76±0.05	0.61±0.08	0.57±0.07
	ViT	0.78±0.02	0.59±0.03	0.58±0.03
Por enfermedad (6 clases)	LightOCT	0.81±0.05	0.50±0.08	0.43±0.07
	ViT	0.82±0.06	0.53±0.08	0.47±0.07

Fuente: Tampu et al. (2023). *Diseased thyroid tissue classification in OCT images using deep learning: Towards surgical decision support.*

Según JERBI et al. (2023) el análisis de imágenes médicas ha tomado gran importancia las últimas décadas en los procesos de diagnóstico de enfermedades. Sin embargo, la complejidad y variedad de este tipo de imágenes conlleva a buscar nuevos métodos de análisis.

Específicamente las tareas de clasificación de imágenes médicas de tiroides conlleva grandes retos que requiere nuevas formas de análisis.

Existen distintos tipos de imágenes médicas para la tiroides. Se tiene a las imágenes de resonancia magnética (MRI) que usa las ondas de radio para crear las imágenes, las tomografías computarizadas (CT) que usan rayos X, tomografías por emisión de positrones (PET) que emplea una serie de fármacos radioactivo o trazadores además de una máquina de escaneo, finalmente, se tiene a las imágenes de ultrasonido, los cuales usan ondas sonoras de alta frecuencia. Este último es considerado como el proveedor de mayor información (estructura, forma y cantidad) de los nódulos en la glándula de la tiroides.

La decisión final en los diagnósticos de este tipo de casos es subjetiva. Esto debido a que se depende de la propia experiencia de los médicos y su mismo entorno.

Es por esto que se halla la necesidad de usar tecnologías, como las basadas en Inteligencia Artificial, que ayuden a los médicos a realizar diagnósticos más acertados. Por esto, se menciona que existen varios tipos de software aprobados y usados en varias situaciones médicas. Uno de los de mayor desempeño son los sistema de diagnósticos asistido por computadora (CAD), unos de los software con Inteligencia Artificial más usados. Este tipo de programas de computadora son capaces de analizar imágenes médicas y otorgar un pre-diagnóstico capaz de ayudar en la toma de decisiones.

A pesar de la capacidad de los actuales sistemas CAD capaces de realizar tareas de clasificación de imágenes médicas, hay una necesidad de mejores métodos y técnicas capaces de procesar grandes cantidades de datos.

En ese contexto, los últimos años se han ido desarrollando modelos basados en Redes Neuronales Convolucionales, como VGG16, EfficientNet y ResNet50, capaces de manejar grandes cantidades de datos.

El rendimiento de este tipo de modelos también se ve beneficiado por técnicas de Aumento de Datos donde se generan nuevas imágenes a través de; por ejemplo, su rotación, cambio de saturación o realce. Sin embargo, también existen técnicas más avanzadas de Aumento de Datos como el uso de Deep Convolutional Generative Adversarial Networks (DCGAN) que tiene la capacidad de generar imágenes a través de tomar como entrada algunas imágenes reales. Este tipo de procesos tienen el objetivo de solucionar el desbalance o baja cantidad de datos.

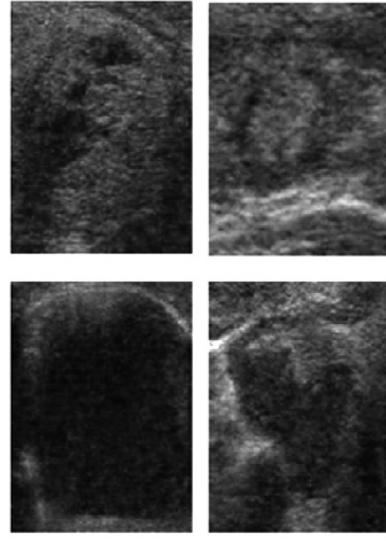
Se menciona que otra técnica que ha ido tomando gran importancia estos años son los Vision Transformers (ViT), basadas en los modelos transformer originales diseñadas para tareas de Procesamiento de Lenguaje Natural (NLP).

En esta investigación se desarrolló un modelo híbrido entre ViT y CNN con el objetivo de obtener una arquitectura capaz de otorgar predicciones más acertadas que los convencionales modelo de Deep Learning en la tareas de clasificación de nódulos tiroideos en la glándula de tiroides a través de imágenes de ultrasonido.

El conjunto de datos usado fue el de CIM@LAB de la Universidad Nacional de Colombia y el Instituto de Diagnóstico Médico. Este consiste en 472 imágenes de ultrasonido de la glándula tiroideas donde 397 eran de carácter maligno, y 75 de carácter benigno. Las imágenes poseen distintos tamaños, por ello se procedió con su redimensionamiento a 128 x 128 píxeles.

Debido a la baja cantidad de muestras del conjunto de datos, se realizó un Aumento de Datos a través del modelo DCGAN, el cual generó 10 000 imágenes, repartidas homogéneamente en cada clase.

Algunas imágenes generadas se muestran en la Figura 11.



**Generated Images:  
DCGAN**

**Figura 11.** Imágenes de ultrasonido de la tiroides generadas.

Fuente: JERBI et al. (2023). *Automatic classification of ultrasound thyroids images using vision transformers and generative adversarial networks.*

La arquitectura del modelo generador consta de una entrada que es ruido generado aleatoriamente a través de una distribución normal estándar, luego se pasa a través de seis convoluciones 2D transpuestas (fractionally-strided convolution) con tamaño de kernel 4 x 4 con el fin de hacer up-sampling hasta llegar al tamaño de 128 x 128. Todas las capas convolucionales excepto la primera tienen un stride de 2. La primera tiene el stride definido en 1 pixel. Finalmente, se usa ReLU en todas las capas convolucionales a excepción del a última.

La parte de la arquitectura encargada de discriminar tiene una entrada de tamaño 3 x 128 x 128. Este parte del modelo también incluye 6 capas de convolución 2D con un kernel de 4 x 4 y un stride de 2 para todas la capas a excepción de la capa final. Se usó Leaky ReLU con pendiente negativa de 0.2 como función de activación en todas la capas sin contar la capa final que se definió la función sigmoide.

Para la división del conjunto de datos, se usaron dos métodos distintos. El primero consistía en dividir el toda la data en 80 % para el entrenamiento (4 000 imágenes de nódulos benignos y 4 000 imágenes de nódulos malignos), 10% (500 benignos y 500 malignos) para la validación y 10% (500 benignos y 500 malignos) para la prueba. El segundo método fue el cross-validation con k-fold de tamaño de 10; es decir, se tomaron 9 000 imágenes para el entrenamiento y 1 000 para la prueba.

Las Redes Neuronales Convolucionales VGG16, EfficientNetB0 y ResNet50 se probaron con el conjunto de datos (dividido por los dos métodos mencionados) utilizando Transfer Learning. En la parte del clasificador se usó Softmax y SVM con un kernel igual a l2. Esta parte sirvió para definir cuál modelo CNN será usado para la implementación de la arquitectura híbrida.

La arquitectura de ViT usada fue el ViTB16. Este modelo recibe patches de 16 x 16. Es decir, las imágenes de ultrasonido se dividirán, de acuerdo a su tamaño, en patches de tamaño 16 x 16. Luego, cada uno de estos es aplanado y mapeado a través de una proyección lineal.

A la secuencia de embedded patches se le añade un nuevo tipo de embedding a la posición 0. Además, se añade otros embeddings 1D a cada embedded patch que indiquen sus respectivas posiciones. Esto finalmente será usado en el encoder del transformer.

El Transformer Encoder consta de un Multi-Head Self Attention (MSA) y un Perceptrón Multicapa (MLP) (dos capas con función GELU), donde la salida del MSA es alimentada al MLP.

La investigación propone que en lugar de alimentar con patches al modelo transformer, se use las salidas de un modelo CNN (ResNet50) como entrada. Esto se logra a través de pasar por un proceso de patching a los tensores resultantes del modelo CNN.

Los modelos CNN y ViT fueron entrenados con los mismos hiper-parámetros (optimizador SGD, tasa de aprendizaje de 0.0001, tamaño de lote de 32, 20 épocas e imágenes de 128 x 128).

En el caso del modelo híbrido de mayor desempeño, se usó la función de activación Softmax y la función de pérdida binary-cross entropy. El clasificador fue un modelo SVM con kernel l2, función de activación lineal y función de pérdida hinge.

Las métricas usadas para evaluar los modelos entrenados fueron el accuracy, precision, recall y f1-score.

Los resultados se muestran en la Tabla 28.

Tabla 28

*Comparación de los modelos entrenados.*

	Modelo	División de datos	F1-score	Recall	Precision	Accuracy
Softmax Classifier	VGG16	División clásica	93.06 %	93.06 %	93.06 %	92.90 %
	EffecientNetB0	División clásica	86.42 %	86.42 %	86.42 %	86.40 %
	RestNet50	División clásica	96.67 %	96.67 %	96.67 %	96.60 %
	ViT-B16	División clásica	92.28 %	92.28 %	92.28 %	92.10 %
	Hybrid ViT	División clásica	97.87 %	97.87 %	97.87 %	97.50 %
	VGG16	10-Folds	93.24 %	93.24 %	93.24 %	93.42 %
	EffecientNetB0	10-Folds	87.09 %	87.09 %	87.09 %	87.16 %
	RestNet50	10-Folds	96.66 %	96.66 %	96.66 %	96.71 %
	ViT-B16	10-Folds	96.10 %	96.10 %	96.10 %	95.98 %
	Hybrid ViT	10-Folds	96.96 %	96.96 %	96.96 %	97.10 %
SVM Classifier	VGG16	División clásica	92.78 %	90.82 %	94.96 %	93.69 %
	EffecientNetB0	División clásica	85.38 %	76.85 %	96.42 %	90.79 %
	RestNet50	División clásica	95.98 %	94.14 %	98.03 %	97.20
	ViT-B16	División clásica	94.55 %	92.08 %	97.23 %	95.09 %
	Hybrid ViT	División clásica	96.42 %	94.82 %	98.24 %	96.80 %
	VGG16	10-Folds	92.29 %	90.86 %	93.98 %	93.44 %
	EffecientNetB0	10-Folds	83.95 %	75.38 %	95.46 %	90.51 %
	RestNet50	10-Folds	96.46 %	94.96 %	98.16 %	97.32 %
	ViT-B16	10-Folds	96.12 %	95.14 %	97.18 %	96.17 %
	Hybrid ViT	10-Folds	96.67 %	95.01 %	98.51 %	97.63 %

Fuente: JERBI et al. (2023). *Automatic classification of ultrasound thyroids images using vision transformers and generative adversarial networks.*

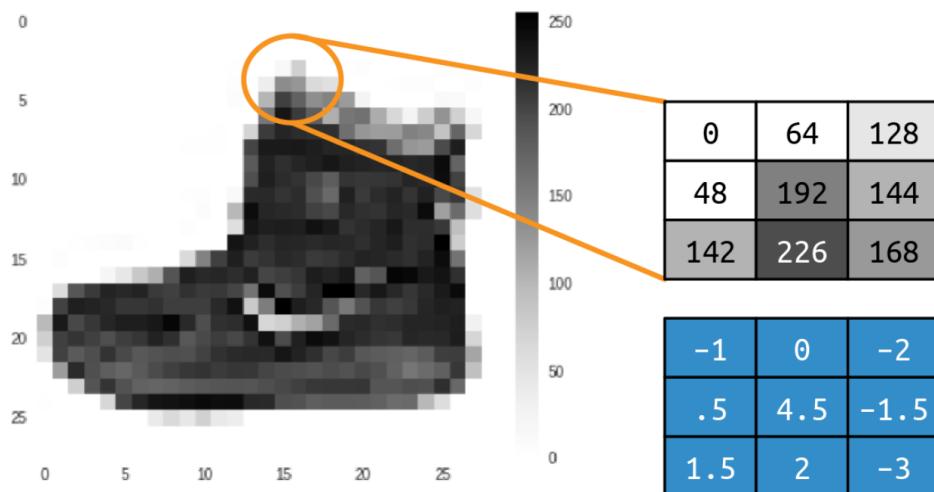
En relación a los modelos CNN, se pudo observar que el de mayor desempeño fue el RestNet50 con el clasificador SVM, alcanzando un accuracy de 97.63 %. Este hallazgo permitió determinar el CNN a usar en el modelo híbrido propuesto (ResNet50-ViT-B16) que obtuvo finalmente el mejor desempeño general con un accuracy de 97.63 %.

## 2.2 Bases Teóricas

### 2.2.1 Deep Learning

El Deep Learning o Aprendizaje Profundo es una rama del Machine Learning que conforma varios tipos de algoritmos y enfoques mucho más amplios. Este involucra tratar a los problemas que requiere mayor generalización como lo es la visión por computadora y el reconocimiento de voz. Es decir, estos problemas generales se refieren a problemas que los humanos pueden resolver con facilidad. El Deep Learning involucra al Aprendizaje Supervisado, No Supervisado y el Aprendizaje por Refuerzo. El concepto de “profundidad” se refiere a la característica de sus enfoques en usar muchas capas de redes neuronales artificiales, donde cada una de estas realiza una operación especial, para que en conjunto su complejidad y potencia de resolución de problemas sea mayor. (Hurbans, 2020)

En el caso de esta investigación, una de las técnicas del Deep Learning a usar son las Convoluciones. Esta técnica permite la extracción de características de imágenes para posteriormente realizar su clasificación con respecto a estos. En el proceso de convolución es importante mencionar a los filtros. Estos contienen pesos que son multiplicados por los valores de los pixeles de una imagen, para que finalmente se obtenga un nuevo valor de pixel (Moroney, 2020). En la Figura 12 se presenta un ejemplo gráfico de convolución 2D en una imagen del conjunto de datos Fashion MNIST.



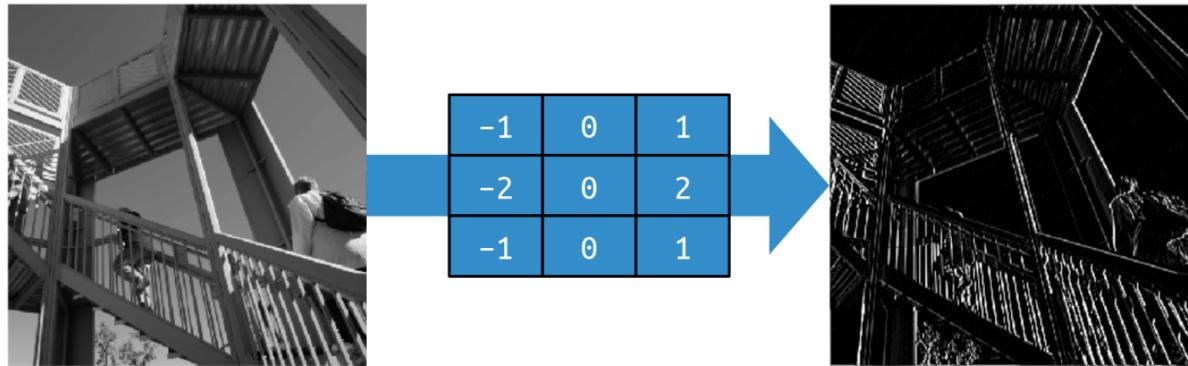
**Figura 12.** Convolución de imagen de Fashion MNIST.

Fuente: Moroney (2020). *AI and Machine Learning for Coders*.

Con un filtro de 3x3 como se muestra en la imagen, se puede modificar el valor del pixel. Este proceso se repite con cada pixel en imagen. Finalmente, se obtiene una imagen

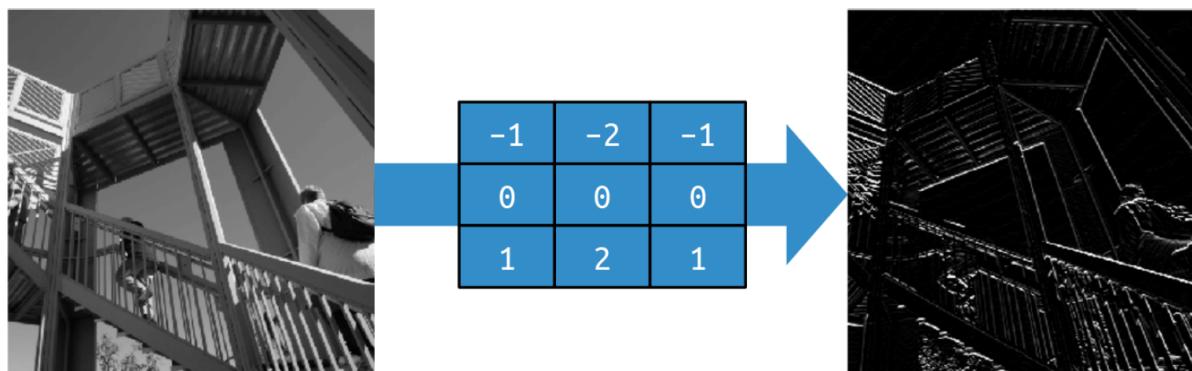
nueva conocida como una imagen filtrada. (Moroney, 2020) En el ejemplo anterior, el nuevo valor del pixel original 192 será 577, pues este es el resultado de multiplicar, y posteriormente sumar, los pesos del filtro con el valor del pixel y sus vecinos.

Existen distintos tipos de filtros que modifican y otorgan distintos tipos de resultados. Como ejemplo se presenta la Figura 13 y la Figura 14.



**Figura 13.** Convolución de imagen con filtro de líneas verticales.

Fuente: Moroney (2020). *AI and Machine Learning for Coders*.



**Figura 14.** Convolución de imagen con filtro de líneas horizontales.

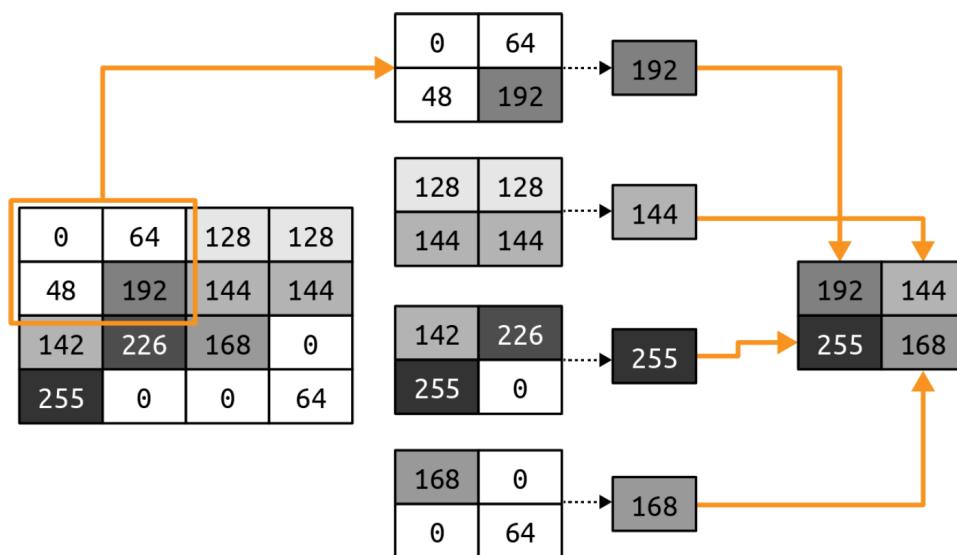
Fuente: Moroney (2020). *AI and Machine Learning for Coders*.

El primer filtro realiza modificaciones a una imagen original para finalmente obtener otra imagen distinta en donde se resaltan las líneas verticales. Caso contrario pasa con el segundo filtro, donde la imagen original es modificada para resaltar sus líneas horizontales. Así, existen distintos tipos de filtros con distintos pesos. Cada uno de estos permite resaltar en las imágenes, a través de modificaciones en sus píxeles, sus características más importantes que

serán de utilidad para diferenciar entre una clase u otra de un conjunto de imágenes. También se podría decir que esta técnica de aplicar convoluciones permite reducir la cantidad de información presente en las imágenes, entonces se podría aprender o encontrar un conjunto de filtros específicos capaces de reducir la gran cantidad de información de las imágenes en características que estén relacionadas a sus respectivas etiquetas (Moroney, 2020).

Para reducir la cantidad de información presente en las imágenes, manteniendo al mismo tiempo las características obtenidas por las convoluciones, es necesario aplicar otra técnica dentro del mundo del Deep Learning.

El proceso de pooling consiste en eliminar cierta cantidad de pixeles en una imagen, mientras se mantienen las partes resaltantes de la misma. Este proceso normalmente se realiza con agrupaciones de 2x2 en los pixeles de una imagen. Estas agrupaciones también son conocidas como pool. Dentro de cada uno de estas se selecciona el máximo valor de pixel. El proceso se repite con nuevos grupos de pixeles para finalmente obtener una nueva imagen de tamaño considerablemente reducido. (Moroney, 2020) En la Figura 15 se presenta un ejemplo gráfico.



**Figura 15.** Ejemplo de max-pooling con un pool de 2x2.

Fuente: Moroney (2020). *AI and Machine Learning for Coders*.

La imagen original de 4x4 pixeles, luego de ser aplicado el proceso de pooling, se obtiene una imagen reducida de 2x2 pixeles. De manera general, la imagen redujo a la cuarta parte de la cantidad original de pixeles.

Con estas dos técnicas del Deep Learning se han ido desarrollando distintas arquitecturas de redes neurales en la última década. Algunas más complejas y con más capas que otras.

Cada una de estas han logrado desempeñarse de forma satisfactoria con el conjunto de datos con el que se han entrenado, demostrando el gran potencial de la Redes Neuronales Convolucionales o CNN.

Algunas de las arquitecturas más conocidas y que han sido usadas en gran variedad de investigaciones son los VGG, ResNet, Inception y DenseNet.

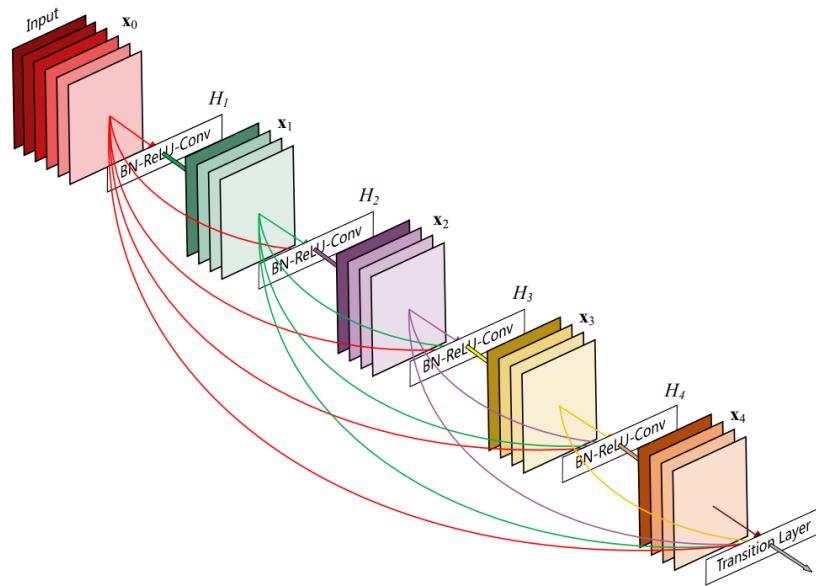
La arquitectura VGG o VGGNet es una red neuronal artificial con una profundidad de 16 o 19 capas (dependiendo de la versión que se analice) y usa pequeños filtros de convolución de tamaño 3x3 a través de estas. Este modelo participó en el ImageNet Challenge 2014, donde logró el primer puesto en las tareas de localización y segundo puesto en clasificación. Además, tiene la alta capacidad de generalización en otros conjuntos de datos, es decir, es capaz de obtener buenos resultados con imágenes distintas a las usadas para su entrenamiento. (Simonyan & Zisserman, 2015)

Otra arquitectura bastante difundida en el mundo del Deep Learning y las CNN es ResNet. Esta arquitectura nace de la premisa del aumento de la dificultad de entrenar las redes neuronales profundas. Con el objetivo de facilitar este proceso se añade a las ya conocidas CNN el concepto de residual, para obtener redes residuales que sean mucho más fáciles de optimizar y capaces de obtener mejor desempeño a más grande sea la profundidad de la red. El modelo tuvo 152 capas, ocho veces más grande que las arquitecturas VGG, y fue evaluada con el conjunto de datos ImageNet. (He et al., 2016)

Inception es una arquitectura que obtuvo un alto desempeño en las tareas de clasificación y detección en el ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). La principal distinción de este modelo es el uso de los recursos informáticos a través de la red. Esto significa que mientras más aumente la complejidad del modelo a través de su profundidad, los recursos computacionales no variarán a gran escala como se esperaría. Esto se logra a través del uso de módulos de Inception en algunas capas de la arquitectura en general. Los distintos módulos se encuentran apilados unos sobre otros y con algunas capas max-pooling. Dentro de esta arquitectura, se tiene a GoogLeNet, que no es más que una versión de Inception de 22 capas (este modelo fue el presentado a ILSVRC14). (Szegedy et al., 2015)

DenseNet nace con la premisa de que las CNN pueden obtener mejores desempeños si se tienen conexiones cortas entre las capas cerca de la entrada y salida de la red. La propuesta de esta red consiste en establecer una conexión de cada capa con todas las demás capas de la misma red. Esto trae grandes ventajas como la reducción del problema de gradiente de fuga, fomenta la reutilización de características extraídas en previas capas, además de disminuir la cantidad de parámetros. El modelo fue evaluado con los conjuntos de datos CIFAR-10, CIFAR-100, SVHN e ImageNet para la tarea de reconocimiento de objetos. (Huang et al., 2017) En la

Figura 16 se muestra una representación de DenseNet con 5 capas.



**Figura 16.** DenseNet con 5 capas.

Fuente: Huang et al. (2017). *Densely Connected Convolutional Networks*.

A pesar del gran desempeño del los modelos de basados den CNN, los últimos años han aparecido nuevas y prometedoras arquitecturas capaces de superarlos.

Según la investigación original presentada por Dosovitskiy et al. (2021), los modelos Transformer, dominantes actuales en tareas de procesamiento de lenguaje natural (NLP), se han convertido en el modelo predilecto cuando se trata de manejo de texto, esto debido a su capacidad de utilizar los recursos computacionales de forma eficiente y su escalabilidad, permitiendo así obtener modelos de; por ejemplo, 100 mil millones de parámetros.

Según Vaswani et al. (2017) los Transformer tiene una arquitectura similar a los modelos Neural Sequence Transduction que contienen dos partes importantes: encoder y decoder. El encoder se encarga de realizar un mapeo de las entradas a secuencias continuas. El decoder recibe este último para finalmente generar un secuencia de símbolos de forma autorregresiva, generando un símbolo a la vez mientras utiliza los anteriores símbolos ya generados como entrada. El modelo Transformer sigue este mismo comportamiento a través de su mecanismo característico de Self-Attention junto a sus capas conectadas completamente.

En el encoder de las arquitecturas Transformer se tienen 6 capas que se componen de otras 2 subcapas: Multi-Head Self-Attention y Fully Connected Feedforward en función a su posición. Además, entre estas dos subcapas, se aplicaron conexiones residuales y capas de

normalización. Finalmente, también se definió una capa de Embedding. La dimensión de los vectores generados por todas estas capas es de 512.

En el decoder también se tienen 6 capas idénticas entre sí. Además, similar al encoder, las dos capas Multi-Head y Fully Connected también están presentes con la diferencia de una nueva subcapa Multi-head que recibe como entrada la salida del encoder. Otra característica distintiva del decoder es el masking aplicado a la capa Multi-Head que surge como medio para evitar que las posiciones tengan la capacidad de atender a las siguientes posiciones. Finalmente, también se emplearon conexiones residuales y capas de normalización de igual manera que en el encoder.

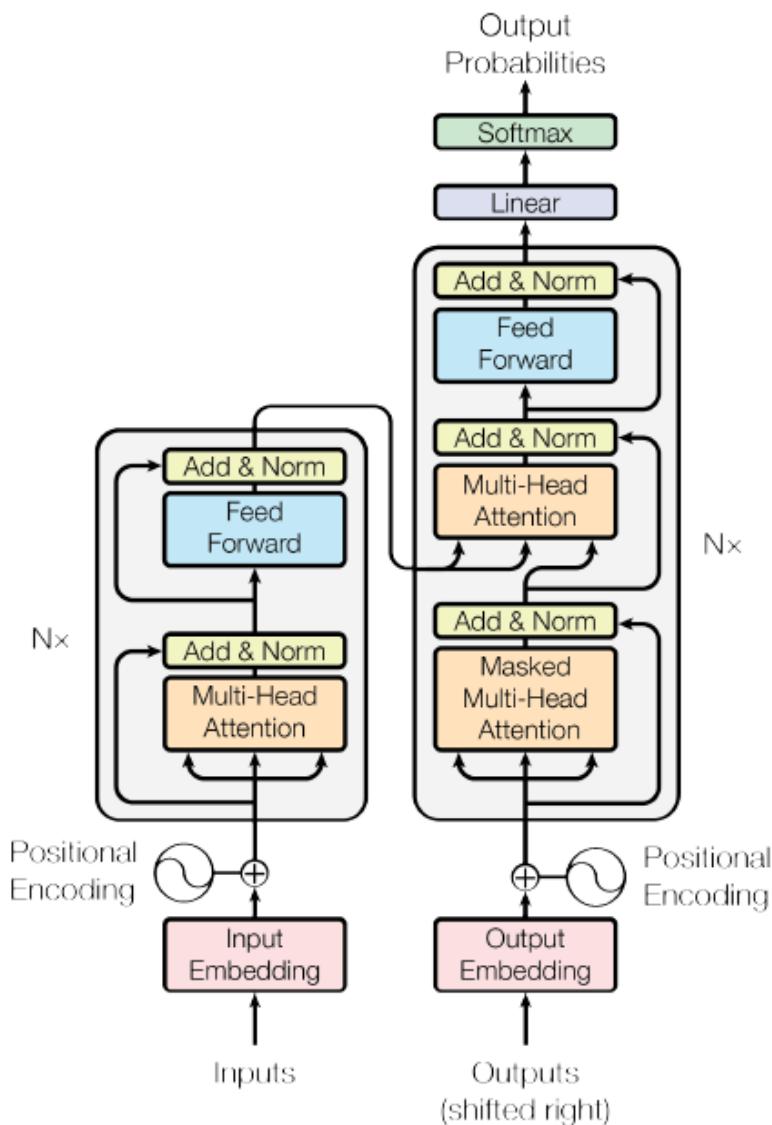
El proceso de Attention en los Transformers se define como el mapeo de vectores de alta dimensionalidad (query, key y value) a una salida (también vector de alta dimensión). Esta salida se calcula mediante a través de pesos, que se obtienen con una función de compatibilidad entre el query y el key, y que son asignados al vector value para finalmente realizar una suma ponderada con estos. Todo este proceso se realiza de forma paralela en sus proyecciones lineales donde sus salidas serán concatenadas y proyectadas nuevamente. A este proceso, se le conoce como Multi-Head Attention.

Además de las capas de atención, tanto el encoder como decoder del Transformer tienen una subcapa de Fully Connected Feedforward en función a la posición. Este tipo de capa aplica una transformación lineal usando además la función de activación ReLU.

Las capas Embedding son usadas para llevar los token de entradas a vectores de alta dimensionalidad. Mientras que las funciones de transformación lineal y softmax cambian la salida del decoder para obtener las probabilidades predichas.

Junto a los Embedding, y tanto en el encoder y decoder del Transformer, también se definen los Positional Encoding. Estos se encargan de otorgar información sobre la posición de los token en la secuencia a la que pertenecen. Estos tienen la misma dimensión que los propios Embedding.

Cada una de las partes de la arquitectura de los Transformer descritos anteriormente se puede observar de forma gráfica en la Figura 17.



**Figura 17.** Arquitectura del modelo Transformer.

Fuente: Vaswani et al. (2017). *Attention is all you need*.

En el contexto de la visión por computadora, Dosovitskiy et al. (2021) menciona que los primeros intentos de uso de esta arquitectura se basaban en aprovechar la capacidad de Self-Attention de los Transformer y combinarlo con los dominantes modelos CNN. Estos intentos de obtener los excelentes resultados de los Transformer en las tareas de visión por computadora no obtenían los resultados de los modelos de mayor desempeño como son los basados en la arquitectura de ResNet.

Debido a esto y al gran potencial que tenían las arquitecturas basadas en Transformers, propusieron el uso de estas arquitecturas aplicadas de forma directa a imágenes; es decir, aplicando la menor cantidad de cambios antes de ingresar al mismo Transformer. El proceso exacto

de cómo funciona su arquitectura se explicará más adelante.

La gran ventaja de este nuevo tipo de modelos se basa en la cantidad de datos con la que es entrenado.

En el caso de usar una pequeña cantidad de imágenes para el entrenamiento se obtienen resultados no favorables, llegando a tener un bajo rendimiento comparado a los modelos basados en CNN. Sin embargo, al realizar su entrenamiento con una mayor cantidad de datos, los Vision Transformer (como se le nombró a este nuevo tipo de arquitectura) obtuvieron mejores resultados que los modelos dominantes en las mismas tareas de visión por computadora.

A diferencia de los Transformer dedica a las tareas de NLP que reciben como entrada token embeddings de una sola dimensión, las imágenes son de dos dimensiones ( $H$ ,  $W$ ) y una cantidad de canales ( $C$ ), lo que dificulta su ingreso directo a este tipo de arquitecturas. Es por esto que es necesario aplicar una división de las imágenes en patches 2D que posteriormente serán aplanados e ingresados al Transformer como haría normalmente los token embeddings.

Estos patches deben tener un tamaño constante  $P$ , es decir que se deben obtener divisiones de las imágenes de tamaño  $P \times P$ . Esto conlleva a que el número de patches final sea igual a  $N = HW/P^2$ .

Una vez se tienen los  $N$  patches, se realiza su aplanamiento y, posteriormente, se mapean en  $D$  dimensiones (tamaño del vector latente usado en el Transformer) a través de una proyección lineal capaz de ser entrenada.

Ya obtenidos estos patch embedding, se añade uno nuevo pero con la capacidad de ser aprendible. Estos se agrupan en una secuencia que ingresarán posteriormente al Transformer encoder. Cabe resaltar que, luego de pasar por el encoder, este último embedding añadido representará a la imagen.

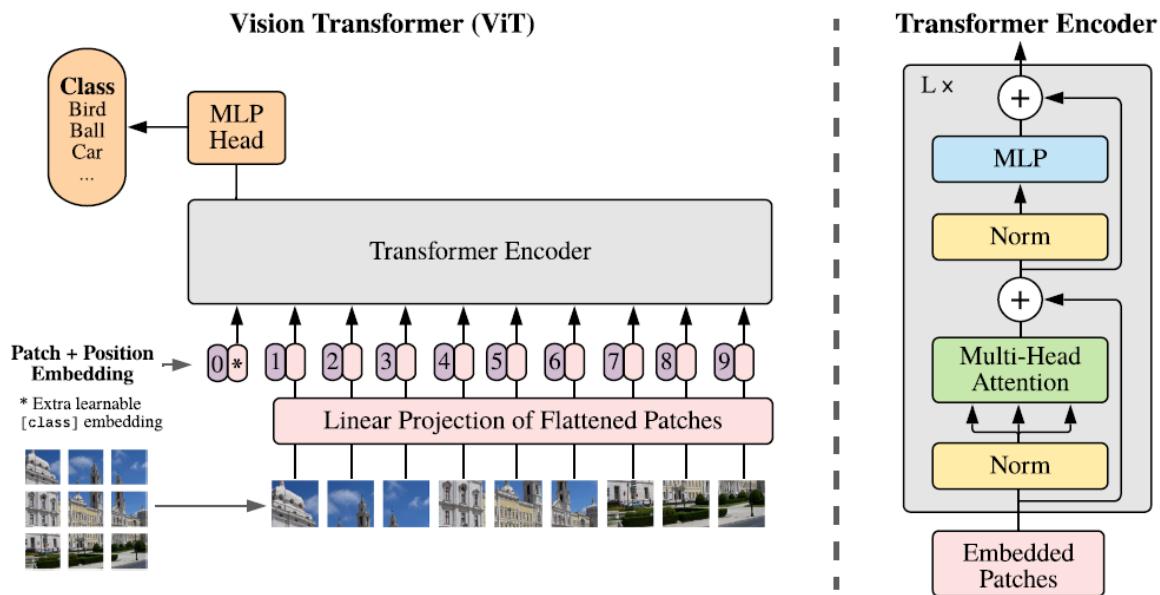
A los ya mencionados patch embedding también se le agrega otro tipo de embedding capaz de mantener información relacionada a la posición de cada patch. Estos embedding son 1D y aprendibles. Finalmente, estos serán ingresados juntos con los originales embeddings a Transformer encoder.

En la parte final de la arquitectura se tiene un MLP con la principal tarea de realizar la clasificación de las imágenes. Este MLP consta de una sola capa oculta en la etapa de entrenamiento inicial del modelo, y con una sola capa lineal en la etapa de fine-tuning.

El bloque de Transformer encoder está compuesto por distintos bloques como los multi headed self attention (MSA), MLP y los Layernorm (LN). Estos distintos bloques se van alternando de forma específica, introduciendo además conexiones residuales luego del MSA y

MLP.

Toda la arquitectura de los ViT se puede resumir con la Figura 18.



**Figura 18.** Arquitectura del modelo ViT.

Fuente: Dosovitskiy et al. (2021). *AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE*.

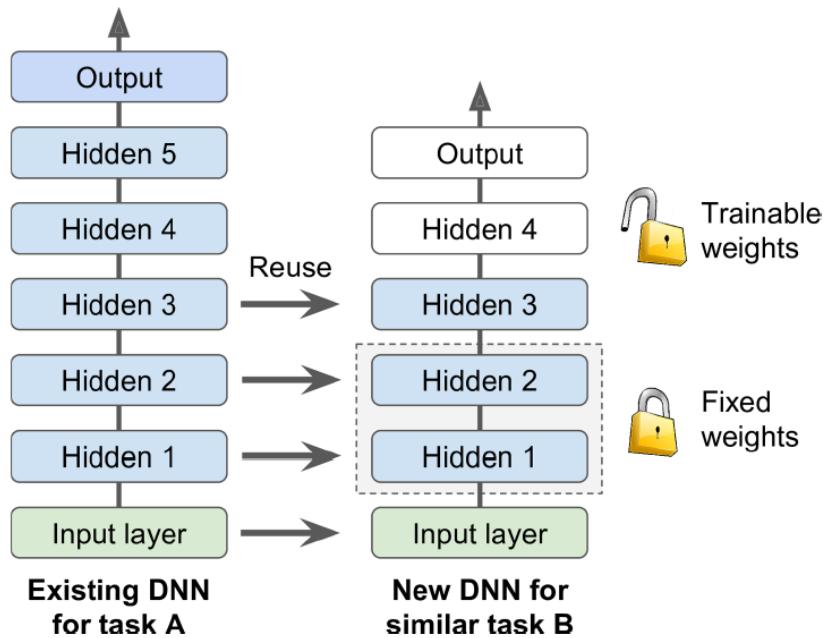
## 2.2.2 Transfer Learning

Géron (2022) nos menciona que el Transfer Learning o Transferencia de Aprendizaje es un técnica usada en el campo de Deep Learning que permite el uso de algunas capas de un modelo ya definido y entrenado previamente en un nuevo modelo que necesite ser entrenado en una tarea similar al que se desarrolla el modelo original. Las capas destinadas al reuso son normalmente las más cercanas a la entrada o también conocidas como capas inferiores. El beneficio de usar esta técnica radica en dos puntos importantes: cantidad de datos requeridos y velocidad de entrenamiento del modelo; es decir, la cantidad de datos que se deben usar para entrenar un modelo de alto desempeño se reduce considerablemente, mientras que el tiempo requerido para terminar este proceso es menor comparándolo a si lo entrenaran desde cero.

Para que esta técnica funcione debidamente, las capas más cercanas a la salida, conocidas también como capas de alto nivel, deben ser reemplazadas, esto debido a que son más específicas de las tareas del modelo original. Esto también incluye a la capa final, ya que posi-

blemente no tenga la cantidad de salidas necesarias para completar satisfactoriamente la nueva tarea.

En la Figura 19 se presenta de forma gráfica la técnica.



**Figura 19.** Ejemplo de Transfer Learning.

Fuente: Géron (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*.

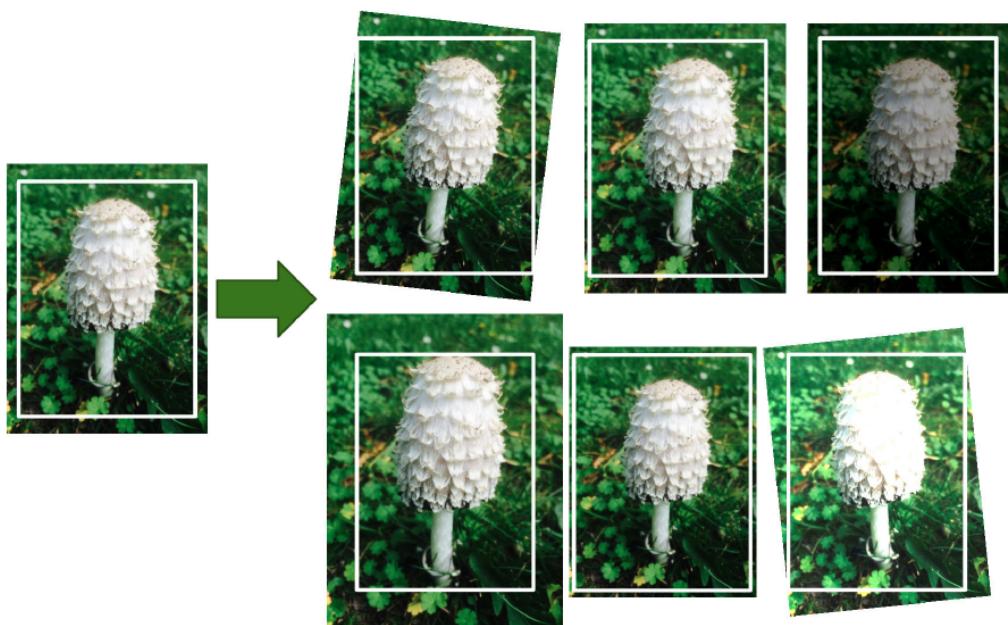
### 2.2.3 Data Augmentation

Según Géron (2022) el Aumento de Datos o Data Augmentation es una técnica de regularización que permite reforzar la cantidad de muestras en un conjunto de datos. Esto se realiza a través de la generación de nuevas instancias similares a los originales; es decir, las personas no deberían ser capaces de diferenciar una imagen generada de una del propio conjunto de datos.

Para generar estas nuevas muestras, normalmente se aplican diferentes transformaciones a las instancias del conjunto de datos original. Estas transformaciones pueden ser; por ejemplo, una simple rotación o recorte de la imagen, siempre y cuando no altere por completo su sentido como es el caso de voltear una imagen de texto de forma horizontal.

El principal beneficio de esta técnica es que permite reducir el sobreajuste de los modelos entrenados.

En la Figura 20 se muestran algunas transformaciones que se pueden hacer al aplicar el Aumento de Datos.



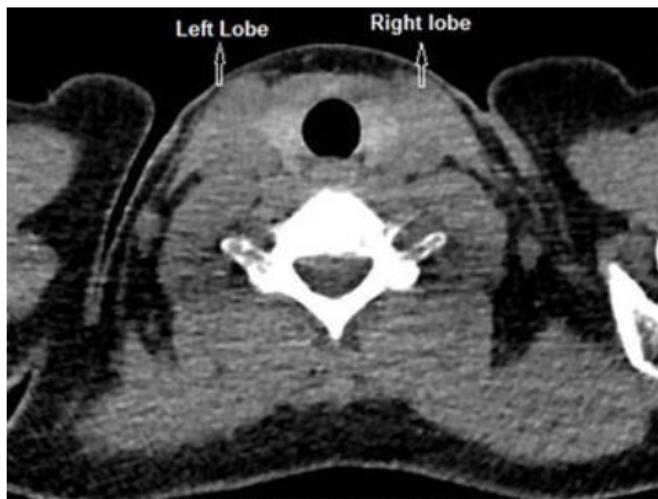
**Figura 20.** Ejemplo de Data Augmentation.

Fuente: Géron (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*.

## 2.2.4 Nódulos Tiroideos

Para entender qué es un nódulo tiroideo, primero se debe saber qué es la tiroides. Este es una glándula localizada en el cuello del cuerpo humano que se encarga de producir hormonas que regulan el metabolismo. Los nódulos que aparecen en esta glándula son los más comunes, y se generan debido a un excesivo crecimiento de las células en esa área, estos pueden ser de textura dura o quísticas. (Deng et al., 2022)

La Figura 21 muestra a la glándula en una imagen tomográfica.



**Figura 21.** Imagen tomográfica de glándula de tiroides.

Fuente: Binboga et al. (2019). *Thyroid Anatomy*.

Shin et al. (2016) mencionan que existen algunas características de los nódulos en la tiroides que pueden ser analizados para realizar una clasificación si es de carácter benigno o maligno. A continuación, se presentan aquellos más resaltantes.

El tamaño de nódulos en ciertos casos puede determinar el tipo con el que se está tratando; sin embargo, esto no está debidamente comprobado, a pesar de que normalmente el riesgo de ser maligno de un nódulo es más alto en aquellos de mayor tamaño. Esto principalmente se debe a que los nódulos benignos también pueden llegar a un tamaño considerable, pero con una mayor cantidad de tiempo. Lo que sí puede significar un probable nódulo maligno es un rápido crecimiento de un nódulo sólido.

El contenido interno de un nódulo está determinado por el ratio entre su parte quística y su parte sólida. Los nódulos malignos con contenido sólido poseen mayor riesgo de ser malignos que aquellos con contenido parcialmente quístico.

La ecogenicidad de los nódulos se mide de acuerdo al nivel de brillo en las imágenes de ultrasonido con respecto a otras partes de la imagen. Esto quiere decir que un nódulo es hipoecoico si su nivel de brillo en la imagen es menor con respecto a otra parte de la imagen, específicamente se compara con el músculo anterior del cuello o el parénquima tiroideo. La mayoría de los tumores tiroideos son hipoecoicos, y existe mayor riesgo de que los nódulos con la misma característica sean malignos.

La forma y orientación de los nódulos también pueden determinar si son o no malignos. La manera en que los nódulos malignos crecen es de manera centrífuga y a través del plano

del tejido, al igual que los benignos; sin embargo, estos crecen de manera paralela. La forma redonda u ovoide de los nódulos se encuentra en aquellos de carácter benigno; sin embargo, no son propias de este. Una característica mucho más específica de los nódulos de carácter malignos es su orientación no paralela, es decir, son más altos que anchos.

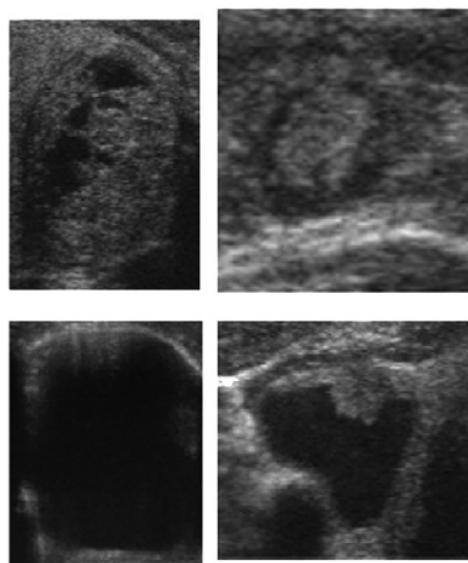
Finalmente, el margen de los nódulos también puede dar información del tipo con el que se trata. Una característica que sugiere que un nódulo es maligno es un margen espiculado o micro tubulado.

### **2.2.5 Ecografía y las imágenes de ultrasonido**

Según Herrera Gajardo (2017), la ecografía, que es una técnica de diagnóstico en donde se usan imágenes generadas por ultrasonido, es comúnmente desarrollado en las áreas de cardiología, ginecología, y otras más relacionadas. La popularidad de esta técnica se basa en la capacidad de las imágenes de alta calidad que se obtienen de este proceso, además de no ser un método invasivo o de radiación como muchos otros de su tipo.

Los sistema encargados de extraer las imágenes de ultrasonido son compuestos de distintas sensores que generan ondas de sonido para posteriormente analizar la respuesta de la interacción física con el campo de interés. Estas señales recibidas de regreso son digitalizados por una parte electrónica delantera que también transforman estos datos crudos en la imagen final. El funcionamiento de este proceso depende de la configuración de los sensores, el método usado para obtener las imágenes y las características de área de interés. (Camacho et al., 2022)

Algunas imágenes de ultrasonido de nódulos tiroideos se muestran en la Figura 22.



**Figura 22.** Imágenes de ultrasonido de nódulos tiroideos.

Fuente: JERBI et al. (2023). *Automatic classification of ultrasound thyroids images using vision transformers and generative adversarial networks.*

## 2.3 Marco Conceptual

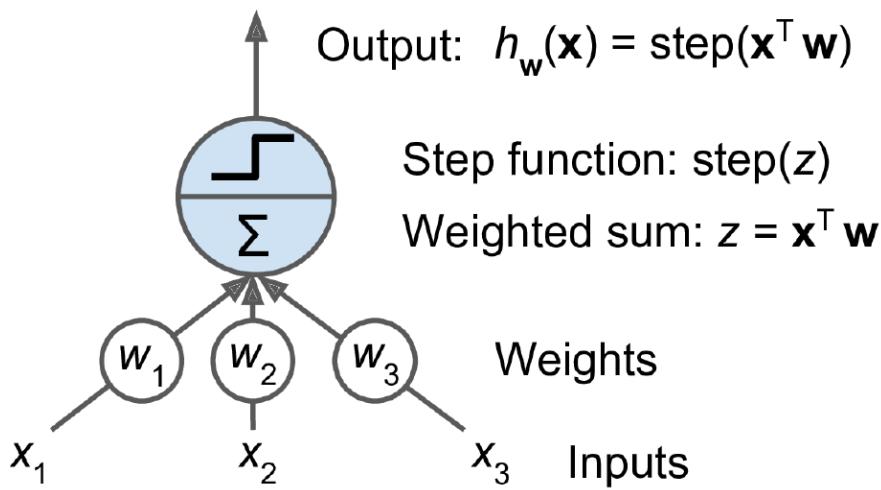
### 2.3.1 Inteligencia Artificial

Definir a la Inteligencia Artificial llega a ser complicado, esto debido a la dificultad que se tiene en definir lo que en verdad es inteligencia. Algunos grandes personajes pensaban de la inteligencia como la ambición, mientras que otros mencionaban a la imaginación como gran impulsador de la inteligencia. También lo consideraban como la capacidad de adaptarse a los cambios. Estas definiciones hacen pensar que no existe un consenso claro de lo que es la inteligencia, menos aun de la Inteligencia Artificial; sin embargo, en términos generales, se puede calificar de “Inteligencia Artificial” a todo aquel sistema sintético que muestra algún tipo de comportamiento “inteligente”. Esto quiere decir que se puede considerar como IA a todo a aquellos que simule los sentidos naturales como el visión o audición, incluso a la capacidad de algunos sistemas de aprender de forma autónoma según su entorno lo requiera. (Hurbans, 2020)

### 2.3.2 Perceptrón

Inventado por Frank Rosenblatt en 1957, el Perceptrón es la unidad básica de una red neuronal e incluso es considerado como la más simple red neuronal artificial (ANN). (Géron, 2022)

El Perceptrón se muestra en la Figura 23.



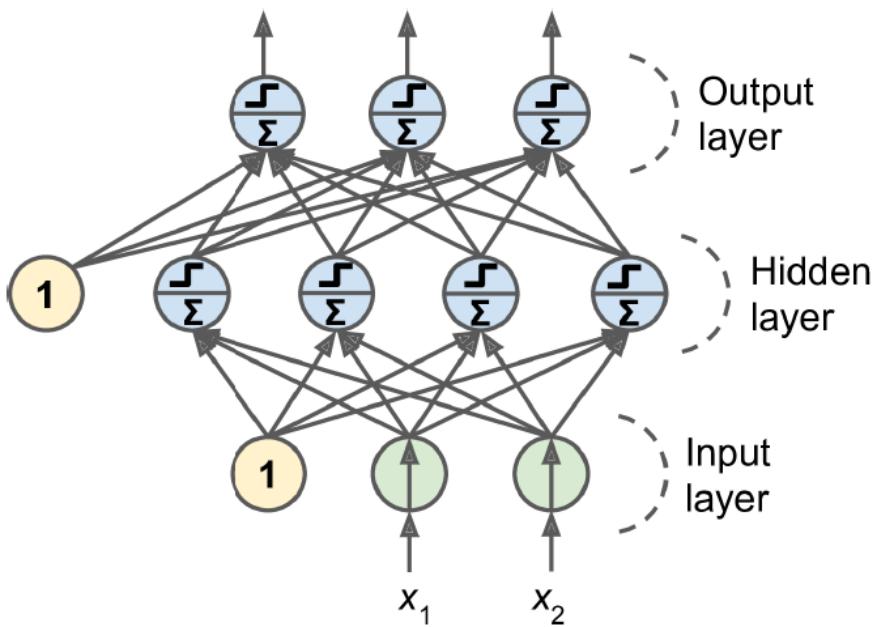
**Figura 23.** Perceptrón.

Fuente: Géron (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*.

### 2.3.3 Perceptrón Multicapa

Un Multi-Layer Perceptron (MLP) consiste en la unión de varios perceptrones, con la característica de estar organizados por capas. Existe una capa de entrada y otra de salida, todas las capas, a excepción de la salida, contiene una neurona adicional llamada bias. (Géron, 2022)

Una representación gráfica de un MLP se muestra en la Figura 24.



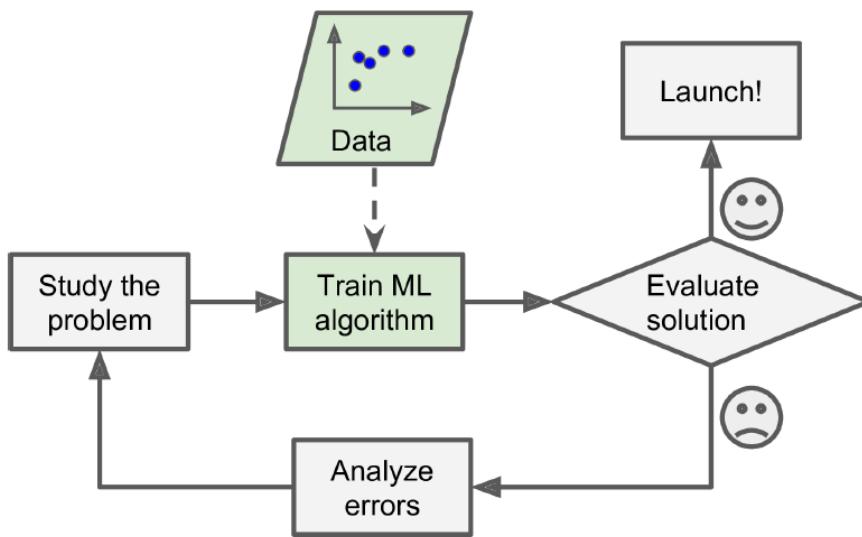
**Figura 24.** Perceptrón Multicapa.

Fuente: Géron (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*.

### 2.3.4 Machine Learning

El Aprendizaje Automático o Machine Learning es considerado como una ciencia y arte de dar instrucciones a una computadora para que pueda aprender por sí sola de una data. (Géron, 2022)

En la Figura 25 se presenta el enfoque tradicional del Machine Learning.



**Figura 25.** Enfoque del Machine Learning.

Fuente: Géron (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*.

## 2.4 Hipótesis

### 2.4.1 Hipótesis General

HG: A través del diseño de un modelo de Deep Learning se pre-diagnóstica los nódulos tiroideos a través de imágenes de ultrasonido.

### 2.4.2 Hipótesis Específicas

- HE1: Las mejores características del conjunto de datos usadas para el proceso de entrenamiento y prueba permiten un mejor desarrollo del modelo de Deep Learning.
- HE2: Las técnicas de preprocessamiento a aplicar a las imágenes de ultrasonido del conjunto de datos mejora el desempeño del modelo de Deep Learning.
- HE3: Las métricas de evaluación de rendimiento de los modelos de Deep Learning logra una correcta comparación de modelos en la tarea de clasificación de imágenes de ultrasonido.
- HE4: Las arquitecturas de Deep Learning con más alto desempeño en la clasificación de imágenes de ultrasonido demuestran superioridad frente a los demás modelos.

## **Capítulo III: Metodología de la Investigación**

### **3.1 Diseño de la investigación**

Según Sampieri et al. (2014) el manipular y encontrar una relación de causa-efecto entre las variables independientes y dependientes son propias de los diseños experimentales.

En el caso de la presente investigación se tiene un diseño experimental porque se pretende establecer la relación entre modelos y técnicas de Deep Learning con el pre-diagnóstico de nódulos tiroideos, esto a través de imágenes de ultrasonido.

#### **3.1.1 Tipo de la investigación**

Ya definido el diseño que tendrá la investigación, se consideró al tipo experimental puro para seguir en la presente investigación, esto ya que la variable independiente (técnicas, modelos y herramientas de Deep Learning) será manipulada constantemente con el fin de encontrar una relación de causalidad ideal con la variable dependiente. Es decir, todo lo relacionado con el Deep Learning será manipulado reiteradas veces con el objetivo de encontrar el impacto en el pre-diagnóstico de nódulos tiroideos. La manipulación en las técnicas o modelos de Deep Learning también incluye a la de la data a usar, ya que se aplicarán diversas técnicas para realizar una limpieza y extracción de características que ayudarán a la posterior elaboración de un algoritmo de diagnóstico asistido.

#### **3.1.2 Enfoque de la investigación**

Según Sampieri et al. (2014) el enfoque que sigue una forma secuencial donde no se puede evitar ningún paso, y se usan herramientas estadísticas para realizar mediciones, es el enfoque cuantitativo.

La presente investigación tiene un enfoque cuantitativo, esto dado que la variable independiente usa valores numéricos y/o estadísticos para su medición. Los resultados de la variable de Deep Learning deberán ser medidos a través de valores numéricos y, en mayor medida, estadísticos.

#### **3.1.3 Población**

La población consiste en imágenes de ultrasonido de la glándula de tiroides con presencia de nódulos. Estas imágenes fueron recolectadas de 2421 pacientes por el Zhujiang Hospital of Southem Medical University y debidamente validadas por los comités de ética institucionales en China.

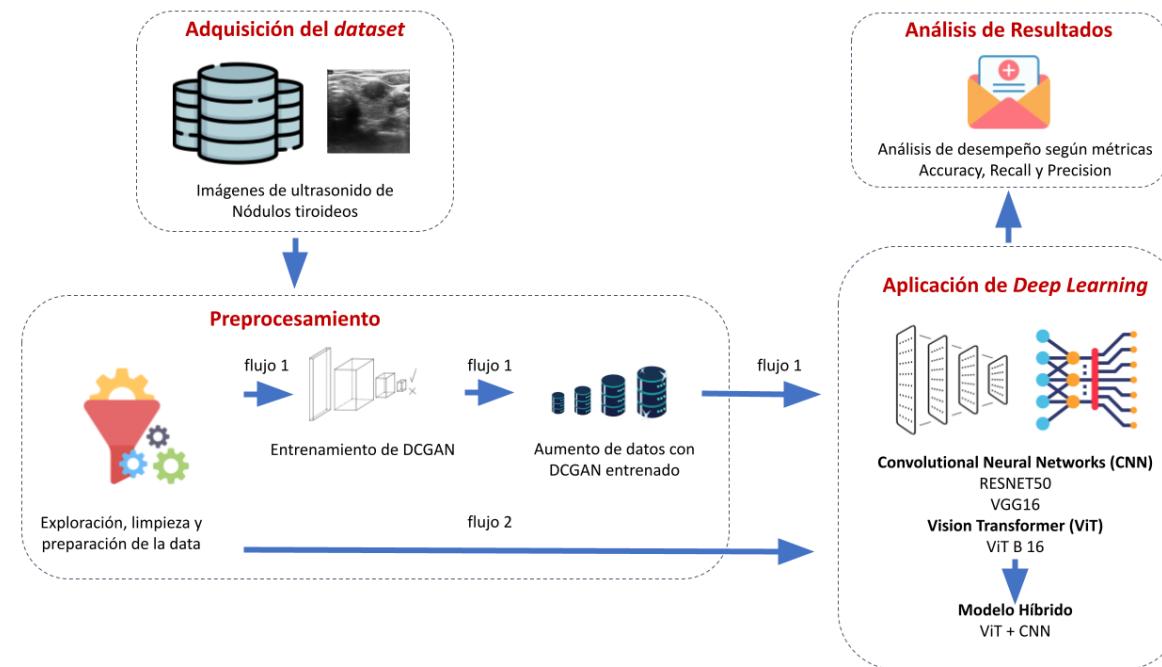
### 3.1.4 Muestra

La muestra consiste en 3493 imágenes de ultrasonido de la glándula de tiroides recopiladas de 2421 pacientes.

## 3.2 Metodología de Implementación de la Solución

La metodología por seguir para implementar un modelo de Deep Learning está basado en gran medida en el conocido ciclo de vida de desarrollo de modelos de Inteligencia Artificial, del cual gran parte de los antecedentes mostrados anteriormente siguen con algunas variaciones.

En el presente caso, se pretende desarrollar un modelo de Deep Learning capaz de brindar ayuda en el diagnóstico médico; por tal motivo, se optó por basar metodología de esta investigación en la de Monroy Malca (2021), modificándolo en cierta medida en el contexto de nódulos tiroideos e imágenes de ultrasonido. En la Figura 26 se presenta de forma gráfica la metodología a seguir.



**Figura 26.** Metodología de implementación.

Fuente: Elaboración propia.

La adquisición del conjunto de datos, es decir, imágenes de ultrasonido, consiste en la búsqueda y revisión de las bases de datos sobre imágenes de nódulos tiroideos, donde finalmente se seleccionará y descargará la de mayor utilidad. La base de datos debe cumplir con ciertos requerimientos para realizar un posterior entrenamiento del modelo que otorgue buenos

resultados. Las imágenes deben ser debidamente etiquetadas según el carácter del nódulo al que representen, es decir, se debe indicar si cada una de las imágenes pertenece a la categoría benigno o maligno. Además, para evitar discrepancias entre calidad de imágenes en futuras evaluaciones, el conjunto de datos debe poseer imágenes de distintas instituciones de salud y de distintas calidades. Esto permitirá que el modelo entrenado no dependa de la una alta o baja calidad de las imágenes para realizar una correcta predicción. Finalmente, la cantidad de datos debe ser relativamente alta con el fin de aumentar la capacidad de generalización del modelo y evitar posibles sobreajustes o bajo rendimiento.

En la etapa de preprocessamiento, se realizará en primera instancia una exploración del conjunto de datos con el fin de entender su composición y las características a mejorar; por ejemplo, un posible desbalanceo de clases o presencia de imágenes corruptas o sin etiquetar. Posteriormente, se realizará limpieza datos en caso de imágenes corruptas o de nula utilidad. Además, se usarán técnicas de Aumento de Datos en la situación de desbalanceo de datos, con el fin de evitar una baja generalización y sobreajuste en la clase mayoritaria. Además, se aplicará un reescalado y normalización en las imágenes destinadas al entrenamiento del modelo con el objetivo de reducir la complejidad computacional y así obtener menor tiempo de ejecución.

Una vez se tenga la data ya preprocessada, una parte de este se usará para el entrenamiento de modelos de Deep Learning. Inicialmente, se planea usar algunos de los diversos tipos y arquitecturas de Redes Neuronales Convolucionales (CNN), específicamente los más utilizados en este tipo de tareas como lo son VGG y ResNet, pues son ideales para la extracción de características de las imágenes, facilitando así el proceso final de clasificación. Además, se desarrollarán modelos basados en arquitecturas de Vision Transformer debido al potencial que han demostrado en anteriores investigaciones. Finalmente, también se realizará el desarrollo de modelos híbridos de CNN con Vision Transformer.

Cada uno de los modelos será probado en la parte restante de la data, específicamente en la data de prueba o test. De aquí se obtendrán las predicciones de los modelos clasificando las imágenes en benigno (0) o maligno (1).

### 3.3 Metodología para la Medición de Resultados de la Implementación

Los resultados obtenidos de la clasificación de los modelos previamente entrenados deberán ser evaluados para una correcta elección final.

Antes de presentar las métricas a usar, es necesario conocer a la matriz de confusión y las partes que lo conforman, pues servirá como base para entenderlas.

Según Izco (2018) la matriz de confusión es una herramienta que permite ver de forma

más clara el rendimiento de nuestro modelo. Este se presenta en la Figura 27.

		Actual Positive	Actual Negative
Predicted Positive	TP	FP	
Predicted Negative	FN		TN

**Figura 27.** Matriz de Confusión.

Fuente: Elaboración propia.

La matriz consta de cuatro partes importantes: TP, FP, FN y TN. Estos serán usados para presentar las fórmulas de las métricas más adelante. El primero (TP o true positive) se refiere a la cantidad de observaciones que se han predicho como positivos y que en verdad sí son positivos; por el contrario, FP (false positive) se refiere a aquellas predicciones dadas como positivos, pero en verdad son negativos. FN (false negative) es la cantidad de observaciones predichas como negativas; sin embargo, estas en realidad son positivas. Finalmente, TN (true negative) es la cantidad de observaciones predichas como negativas y que en realidad son también negativas.

A continuación, se presenta las métricas para medir el desempeño de la clasificación del modelo.

El accuracy representa aquella proporción del total de predicciones que se ha obtenido correctamente (Izco, 2018). Este se calcula a través de la siguiente fórmula 1.

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

El recall representa la proporción de solo los positivos reales predichos de manera acertada (Izco, 2018). Se calcula con la siguiente fórmula.

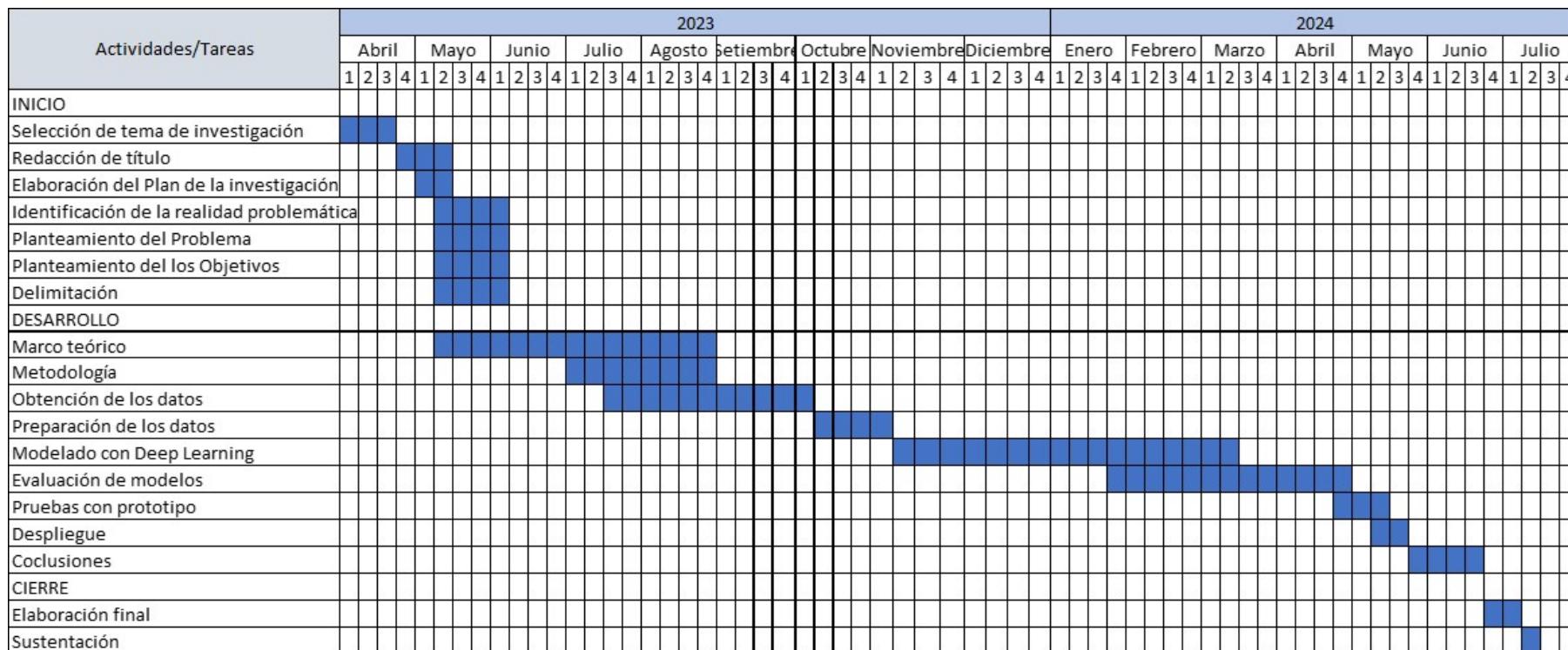
$$\text{recall} = \frac{TP}{TP + FN} \quad (2)$$

Precision representa aquella proporción de lo predicho positivamente que es positiva (Izco, 2018). Se calcula con la fórmula a continuación.

$$precision = \frac{TP}{TP + FP} \quad (3)$$

### **3.4 Cronograma de actividades y presupuesto**

Se propuso un cronograma para la investigación. Conforma desde el inicio hasta ser terminada con la sustentación final planeada para mediados del año 2024. Este se presenta en la Figura 28.



**Figura 28.** Cronograma de actividades.

Fuente: Elaboración propia.

Además, se determinó el presupuesto necesario para la elaboración completa de la investigación. Este se presenta en la Tabla 29.

Tabla 29  
*Presupuesto.*

Grupo	Item	Costo (soles)	Subtotal
Recursos materiales	Laptop	S/ 6,500.00	
	Materiales de escritorio	S/ 100.00	S/ 6,600.00
Software y trámites	Matrícula de Trabajo de Tesis II	S/ 375.00	
	Cuotas de Trabajo de Tesis II	S/ 1,044.00	
	Software	S/ 50.00	
	Renta de servidor en la nube	S/ 224.15	S/ 1,693.15
Extras	Consultorías	S/ 100.00	
	Movilidad	S/ 200.00	S/ 300.00
Total			S/ 8,593.15

Fuente: Elaboración propia.

## Capítulo IV: Desarrollo de la Solución

En el presente capítulo se describe las herramientas y procesos de desarrollo del modelo de Deep Learning.

### 4.1 Determinación y evaluación de alternativas de solución

En esta sección se presentarán las arquitecturas base CNN y ViT que se usarán en el entrenamiento de los modelos de Deep Learning para la tarea de clasificación de nódulos tiroideos a través de imágenes de ultrasonido. Además, también se introducirá la idea de modelo híbrido que junta las capacidades de extracción de características de los CNN con la actualmente popular arquitectura ViT.

En el grupo de modelos CNN, se tiene en primera instancia a la arquitectura VGG16, ya presentada anteriormente en el Capítulo 2. Este modelo posee 16 capas de profundidad junto con un tamaño de 3 x 3 en los filtros. Se ha reconocido que VGG16, y las arquitecturas VGGNet en general, poseen una alta capacidad de generalización, lo cual lo vuelve ideal para tareas de clasificación distintas a las que originalmente fue entrenado. (Simonyan & Zisserman, 2015)

En segundo lugar, también parte del grupo de modelos CNN, se tiene a ResNet 50, también ya presentado inicialmente en el Capítulo 2. Esta arquitectura introdujo el concepto de redes residuales que permitieron obtener de manera más optimizada un mayor desempeño en arquitecturas de alta profundidad. (He et al., 2016) La versión de ResNet usada en esta investigación fue la de 50 capas de profundidad.

En el conjunto de modelos ViT, el usado en la presente investigación fue la ViT-Base con patch de tamaño 16 x 16.

Los modelos ViT también ya fueron presentados en el Capítulo 2.

Esta arquitectura, basada en los originales Transformers, ha demostrado tener mejor desempeño en tareas de clasificación comparado a los modelos basados en CNNs, aunque ciertamente con una mayor necesidad de datos de entrenamiento.

En la Figura 18, ya se muestra la forma de la arquitectura del modelo ViT. Cabe volver a resaltar que la versión del modelo ViT utilizado es el Base que consta de 86 millones de parámetros y recibe patch de tamaño 16 x 16.

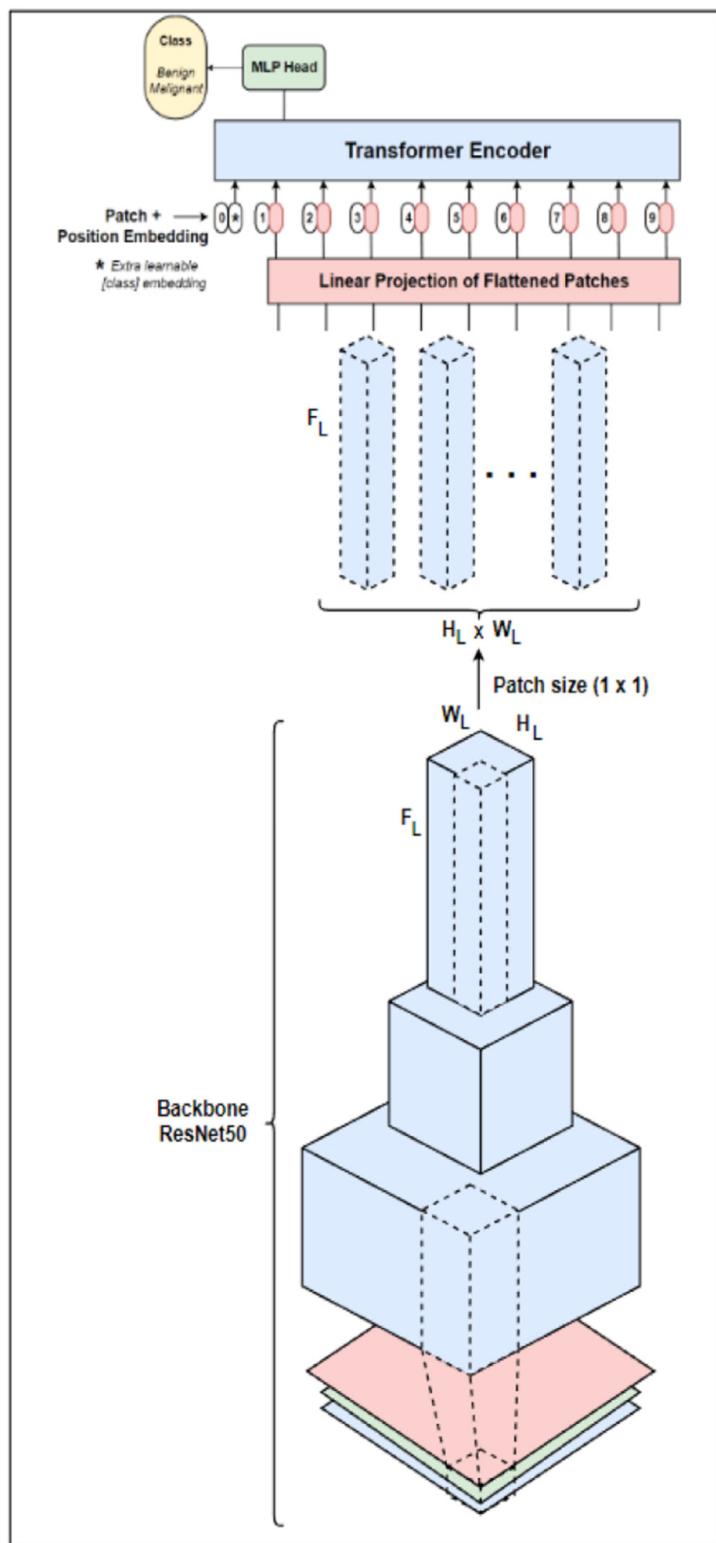
Finalmente, el último modelo presente en la investigación es uno que combina la capacidad de los CNN y los ViT para obtener un modelo híbrido.

Este modelo utiliza al CNN como backbone que alimentará los features extraídos en

forma de patch de tamaño 1 x 1 al modelo ViT. Para lograr esto se es necesario quitar el cabezal clasificador de la arquitectura CNN, pues esta tarea ya será desarrollada por el propio ViT.

Debido a las limitaciones de tiempo y recursos, solamente se desarrollará un modelo híbrido con el CNN de mejor desempeño en los entrenamientos iniciales.

Cabe destacar que este modelo híbrido fue inspirado por el trabajo presentado por JER-BI et al. (2023). A continuación se presenta la Figura 29, donde se muestra la arquitectura híbrida.



**Figura 29.** Imágenes de ultrasonido de la tiroides generadas.

Fuente: JERBI et al. (2023). *Automatic classification of ultrasound thyroids images using vision transformers and generative adversarial networks.*

## 4.2 Propuesta solución

### 4.2.1 Planeamiento y descripción de Actividades

La metodología descrita en el capítulo anterior consta de tres etapas principales: adquisición de datos, preprocessamiento de los datos y la aplicación de Deep Learning; es decir, el entrenamiento de los diferentes tipos de algoritmos. En la siguiente parte de esta investigación se procederá describir las actividades necesarias para desarrollar y completar cada una de estas etapas para, finalmente, obtener los resultados de cada uno de los modelos entrenados.

La metodología inicia con la adquisición del conjunto de datos necesario para realizar el entrenamiento de los modelos de Deep Learning. Este consta de las siguientes actividades.

#### **Actividad 1: Buscar y revisar bases de datos sobre imágenes de ultrasonido de nódulos tiroideos**

En esta actividad se investigarán y analizarán aquellos conjunto de datos que cumplan con las características básicas requeridas: poseer imágenes de ultrasonido de nódulos tiroideos benignos y malignos.

**Entregable:** Diversos conjuntos de datos de imágenes de ultrasonido de nódulos tiroideos benignos y malignos.

#### **Actividad 2: Filtrado de los conjuntos de datos**

El conjunto de datos seleccionado debe cumplir con los siguientes requerimientos: las imágenes deben poseer información del carácter del nódulo; en otras palabras, se debe tener información sobre la imagen cada imagen de ultrasonido donde se indique si se está ante un nódulo de carácter benigno o maligno. Otra importante característica, necesaria para evitar obtener completamente distintos resultados en futuras investigaciones, es la fuente de origen de las imágenes del conjunto de datos, ya que este debe poseer imágenes de diferentes instituciones y herramientas. Esta característica permitirá a los modelos a entrenar tener la capacidad de afrontar distintas calidades y tipos de imágenes de ultrasonido. La última característica radica en la cantidad de las imágenes, ya que el conjunto de datos debe poseer una moderada cantidad de datos que permitan a los modelos interpretar mejor las imágenes de ultrasonido, y así finalmente aumentar la capacidad de generalización del modelo.

**Entregable:** Conjunto de datos de imágenes de ultrasonido de nódulos tiroideos benignos y malignos que cumple con todas las características requeridas.

En la etapa de preprocessamiento, necesaria para realizar un correcto entrenamiento de los modelos de Deep Learning, se realizará un análisis y limpieza de los datos, además, se pasará por

un proceso de preparamiento de los datos para facilitar el flujo de los datos a través de los de los modelos a entrenar. Esta etapa consta de dos flujos, donde uno de estos incluye el Aumento de Datos con el modelo DCGAN; mientras que el otro, evita el paso por este proceso debido a la fiabilidad de este tipo de red para generar datos que puedan ser útiles para el entrenamiento de los modelos. Ambos flujos constan de las tres primeras actividades descritas a continuación.

#### **Actividad 1: Explorar características del conjunto de datos**

Se inicia con la exploración del conjunto de datos seleccionado, extrayendo gráficas y tablas que muestren su composición y características, permitiendo así determinar qué técnicas se podrían aplicar para mejorar las flaquesas del conjunto de datos. Aquí se puede obtener información sobre la necesidad o no de aplicar el Aumento de Datos a una clase minoritaria si lo hubiera.

**Entregable:** Gráficas y tablas de características y composición del conjunto de datos.

#### **Actividad 2: Filtrar y organizar imágenes poco útiles del conjunto de datos**

Se realizará un filtrado de aquellas imágenes corruptas, atípicas o sin etiqueta, dejando solo en el conjunto de datos a aquellas de mayor utilidad, al mismo tiempo que se reorganiza en un nuevo formato de ficheros para facilitar su posterior carga a los modelos de Deep Learning.

**Entregable:** Conjunto de datos organizados y sin imágenes corruptas, atípicas o sin etiquetas.

#### **Actividad 3: Redimensionar, normalizar y dividir los datos según los requerimientos para cada modelo de Deep Learning**

Inicialmente se aplicará a cada imagen un redimensionamiento y normalización según se vea conveniente, esto con el fin de evitar mayor complejidad computacional y reducir los tiempos de ejecución de los modelos. Luego se hará una división del conjunto de datos total en data de entrenamiento, validación y prueba.

**Entregable:** Conjunto de datos de entrenamiento, validación y prueba preprocesados. El segundo flujo de la metodología, al evitar el proceso de Aumento de Datos, obvia la siguiente actividad y continúa directamente en la siguiente etapa.

#### **Actividad 4: Mejorar composición del conjunto de datos**

El Aumento de Datos se debe utilizar en caso de desbalance de clases en el conjunto de datos con el objetivo de aumentar la capacidad de generalización del modelo y evitar su sobreajuste.

Debido a las características que determinan si una imagen de ultrasonido es benigno o

maligno (forma, orientación, tamaño y nivel de brillo), usar las técnicas comunes de Aumento de Datos de hacer diferentes transformaciones en las imágenes originales de forma aleatoria, no tendría sentido. Por este motivo, en esta actividad, se entrenará un modelo DCGAN personalizado que logre aprender la distribución de las imágenes reales y pueda generar posteriormente imágenes sintéticas. El entrenamiento de este modelo se realizará a través del conjunto de datos de entrenamiento obtenido de la anterior actividad, seleccionando aquellos pertenecientes a la clase de minoritaria.

Ya con el modelo entrenado, este será utilizado para generar y aumentar la cantidad de datos final para la clase minoritaria en la data de entrenamiento.

**Entregable:** Conjunto de datos con ambas clases balanceadas.

Una vez se termina el preprocesamiento y se obtiene la data final, se continúa con el entrenamiento y validación de modelos de Deep Learning, seguido de una prueba final con la data respectiva. Esta etapa se divide en las siguientes actividades.

### **Actividad 1: Seleccionar algoritmos a entrenar según usos y resultados de las investigaciones presentadas en los antecedentes**

Se revisará aquellas arquitecturas utilizadas en las investigaciones mencionadas en el Capítulo 2. De todas las recolectadas, se seleccionarán las de mejor desempeño y de mayor presencia.

**Entregable:** Arquitecturas de Deep Learning ideales para la tarea de clasificación de imágenes de ultrasonido de nódulos tiroideos.

### **Actividad 2: Entrenar y probar los modelos de Deep Learning previamente definidos**

Se realizará el entrenamiento de cada uno de las diferentes arquitecturas con configuraciones establecidas por anteriores investigaciones. También se aplicarán dos técnicas por separado para este proceso: Transfer Learning y Fine Tuning. Además, en cada época de entrenamiento de todos los modelos, se usará el conjunto de datos de validación para verificar el avance de la capacidad predictiva de los modelos.

Finalmente, cada uno de estos modelos será puesto a prueba en el conjunto de datos restantes (data de prueba), así finalmente se obtendrán las métricas correspondientes que permitirán luego realizar el análisis de desempeño.

**Entregable:** Modelos CNN y ViT entrenados. Métricas de cada uno de los modelos.

### **Actividad 3: Elaborar un modelo híbrido entre ViT y CNN**

Se desarrollará un modelo híbrido entre CNN y ViT. El modelo CNN funcionará como backbone o modelo base que recibirá las imágenes y posteriormente alimentará al modelo ViT.

Debido a la limitación de tiempo y poder computacional al que se puede acceder, solamente se seleccionará el modelo CNN de mejor desempeño para elaborar un único modelo híbrido.

**Entregable:** Modelo híbrido CNN + ViT.

#### **Actividad 4: Entrenar y probar modelo híbrido CNN + ViT**

Al igual que los anteriores modelos, este nuevo modelo híbrido será entrenado con la data de entrenamiento y con las mismas configuraciones, a excepción de algunas variaciones.

En este caso solo se aplicará la técnica de Fine Tuning debido a la capacidad de procesamiento disponible.

En cada época del entrenamiento, se usará la data de validación para verificar su desempeño en este proceso.

Finalmente, al igual que los otros modelos, el híbrido también será sometido al conjunto de datos de prueba para obtener sus métricas correspondientes.

**Entregable:** Modelo híbrido CNN + ViT entrenando. Métricas del modelo híbrido.

### **4.2.2 Desarrollo de actividades**

En esta sección de la investigación se procederá a desarrollar cada una de las actividades aplicando las técnicas y herramientas ya mencionadas en anteriores capítulos siguiendo la metodología de implementación previamente definida.

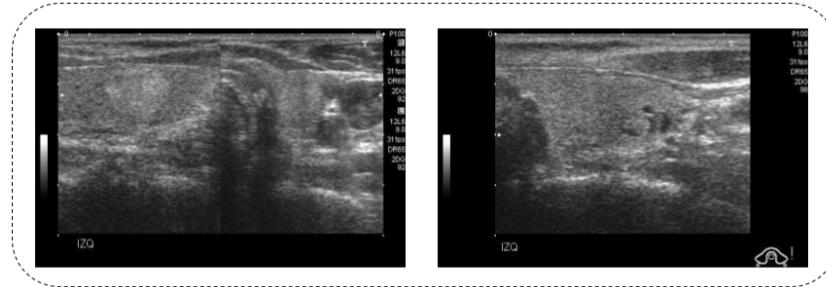
La primera etapa de Adquisición del conjunto de datos se desarrollaron las ya definidas actividades de la siguiente forma.

#### **Actividad 1: Buscar y revisar bases de datos sobre imágenes de ultrasonido de nódulos tiroideos**

En los antecedentes presentados en la sección 2.1, la mayoría de investigaciones relacionadas a imágenes de ultrasonido de nódulos tiroideos, no posee datos de acceso libre, esto conlleva a que se tenga una limitada cantidad de opciones de donde escoger y filtrar el conjunto de datos a usar para entrenar y probar los modelos de Deep Learning.

La base de datos más popular y de acceso libre de imágenes de ultrasonido de nódulos

los tiroides es el dado por la Universidad de Colombia, CIM@LAB y IDIME (Instituto de Diagnóstico Médico). Este conjunto de datos consta de 472 imágenes, de las cuales, la amplia mayoría (397) son de nódulos malignos, mientras que los restantes pertenecen a aquellos de carácter benigno. Este conjunto de datos posee, además, anotaciones de expertos en archivos de formato XML, donde indica diversas características de las imágenes, además de datos personales de los pacientes. En la Figura 30 se muestra algunas imágenes de este conjunto de datos, mientras que en la Figura 31 se muestra un ejemplo de archivo XML también presente.



**Figura 30.** Ejemplo de imágenes del conjunto de datos de la Universidad de Colombia.

Fuente: Elaboración propia.

```
<?xml version="1.0" encoding="UTF-8"?>
<case>
  <number>10</number>
  <age>74</age>
  <sex>F</sex>
  <composition>solid</composition>
  <echogenicity>hyperechogenicity</echogenicity>
  <margins>spiculated</margins>
  <calcifications>microcalcifications</calcifications>
  <tirads>4b</tirads>
  <reportbacaf/>
  <reportteco/>
  <mark>
    <image>1</image>
    <svg>[{"points": [{"x": 326, "y": 172}, {"x": 332, "y": 172}, {"x": 341, "y": 167}, {"x": 364, "y": 165}, {"x": 364, "y": 168}, {"x": 371, "y": 15131}, {"x": 392, "y": 126}, {"x": 389, "y": 120}, {"x": 386, "y": 115}, {"x": 370, "y": 91}, {"x": 364, "y": 89}, {"x": 354, "y": 90}, {"x": 350, "y": 89}, {"x": 313, "y": 87}, {"x": 308, "y": 91}, {"x": 303, "y": 95}, {"x": 300, "y": 121}, {"x": 277, "y": 128}, {"x": 280, "y": 133}, {"x": 283, "y": 1362}, {"x": 296, "y": 162}, {"x": 300, "y": 162}, {"x": 306, "y": 164}, {"annotation": {}, "regionType": "freehand"}]}</svg>
  </mark>
</case>
```

**Figura 31.** Ejemplo de archivo XML del conjunto de datos de la Universidad de Colombia.

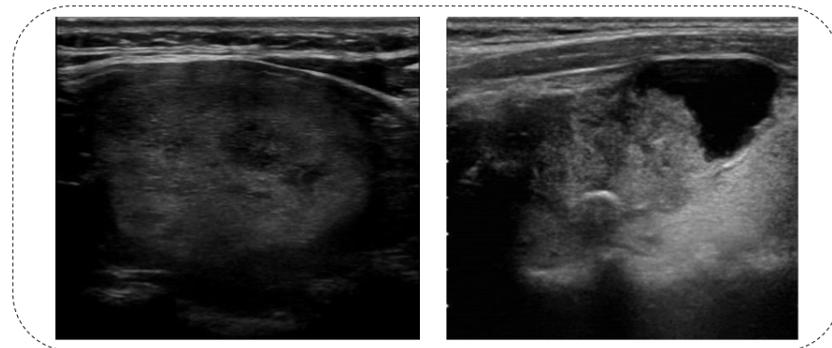
Fuente: Elaboración propia.

Otro conjunto de datos, también de acceso libre, aunque no tan difundido como el presentado anteriormente es el otorgado por Zhujiang Hospital of Southem Medical University que recopila imágenes de 2421 pacientes obteniendo un total de 3493 imágenes de ultrasonido, de los cuales, 2283 pertenecen a imágenes de glándulas de tiroides con nódulos de carácter

benigno, mientras que 1210 pertenecen a nódulos malignos. Además, este conjunto de datos también posee archivos CSV donde especifica el nombre de la imagen y la etiqueta a la que pertenece: benigno (0) o maligno (1).

Cabe destacar que este conjunto de datos posee características particulares que lo diferencian del primero, y es que este, además de estar validado por distintos comités de ética en China, procede de distintas fuentes; es decir, las imágenes pueden variar de tamaño y/o calidad entre sí.

En la Figura 32 se muestran algunas imágenes de este conjunto de datos, mientras que en la Figura 33 se muestra una sección de los archivos CSV.



**Figura 32.** Ejemplo de imágenes del conjunto de datos de Zhujiang Hospital of Southem Medical University.

Fuente: Elaboración propia.

	A	B
1	0130.jpg	0
2	1882.jpg	0
3	2207.jpg	1
4	0690.jpg	0
5	1898.jpg	0
6	0100.jpg	0
7	1013.jpg	0
8	0695.jpg	0
9	1179.jpg	0
10	2305.jpg	0

**Figura 33.** Ejemplo de archivo CSV del conjunto de datos de Zhujiang Hospital of Southem Medical University.

Fuente: Elaboración propia.

### Actividad 2: Filtrado de los conjuntos de datos

Los requerimientos para un buen conjunto de datos radica principalmente en la información del tipo o carácter del nódulo, el origen de las imágenes de ultrasonido y la cantidad de datos.

El conjunto de datos otorgado por la Universidad de Colombia, CIM@LAB y IDI-ME poseía buena información, superando el requerimiento base de tipo o carácter del nódulo, pues otorgaba datos del paciente que podrían ser útiles a la hora de realizar el proceso de pre-diagnóstico; sin embargo, la gran debilidad del este conjunto de datos es la poca cantidad de imágenes que posee. Además, el hecho de no definirse el origen específico de estas imágenes en relación a si se obtuvo de distintas instituciones con diferentes herramientas o no, collevó finalmente a su descarte.

El conjunto de datos de Zhujiang Hospital of Southem Medical University cumple exactamente con otorgar información sobre el carácter de los nódulos de cada imagen que posee, además, también se especifica el origen variado de las imágenes, pues fueron recopiladas de distintas instituciones y con diferentes herramientas. Finalmente, la cantidad de datos, aunque con clases desbalanceadas, supera en gran medida al primer conjunto de datos.

Al cumplir con todos los requerimientos para el entrenamiento de un buen modelo Deep Learning, el conjunto de datos finalmente seleccionado fue el de Zhujiang Hospital of Southem Medical University.

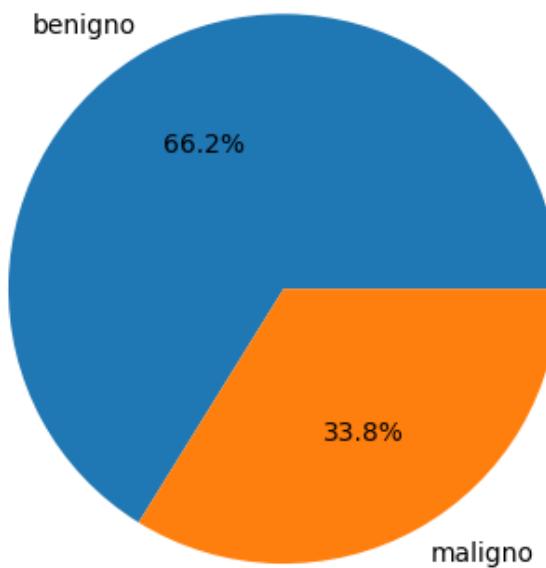
Como se mencionó en la sección anterior, la etapa siguiente es la de preprocesamiento. En este se realizó un completo análisis, limpieza y preparación de los datos para el entrenamiento de los modelos.

Cabe recordar además la existencia de dos flujos en este proceso igualmente descritos en la sección anterior. Ambos inician de igual forma con las tres siguientes actividades.

#### **Actividad 1: Explorar características del conjunto de datos**

Ya con el conjunto de datos seleccionado, se utilizó la data almacenada en los archivos CSV (mostrado en la Figura 33) para contabilizar la cantidad de imágenes por cada clase y así determinar si existía o no la presencia de desbalance. La Figura 34 y Figura 35 muestran gráficas circulares donde se aprecia la proporción del total de datos por cada clase y por cada tipo de datos (entrenamiento y prueba).

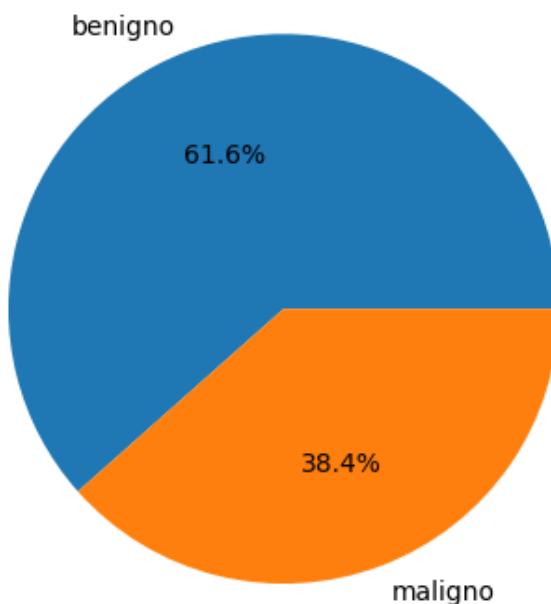
Proporción de las clases en TRAIN



**Figura 34.** Gráfica circular de conjunto de datos de entrenamiento.

Fuente: Elaboración propia.

Proporción de las clases en TEST

**Figura 35.** Gráfica circular de conjunto de datos de prueba.

Fuente: Elaboración propia.

En ambas figuras se puede verificar la presencia de desbalance en clases, observando además como la clase de nódulos benignos duplica aproximadamente a la cantidad de imágenes de la clase de nódulos malignos. Este comportamiento se ve reflejado tanto para los datos de entrenamiento como para los de prueba.

Este descubrimiento afirmó la necesidad de aplicar una técnica de balanceo de datos.

#### **Actividad 2: Filtrar y organizar imágenes poco útiles del conjunto de datos**

A través del algoritmo presentado en la Figura 36 se logró filtrar y obviar aquellas imágenes corruptas (en el caso de este conjunto de datos no se tuvo ninguno). Además, este también permitió la lectura de solo las imágenes con nombre presente en los archivos CSV, lo cual logró evitar el cargado de imágenes sin etiqueta alguna (en el caso de este conjunto de datos tampoco se tuvieron imágenes sin etiquetas). Finalmente, todas aquellos datos que cumplieran con los anteriores pasos serían trasladados a la nueva distribución de ficheros mostrada en la Figura 37.

```

def moveImages(pd_train, pd_test, source_dir_train, source_dir_test, train_dir, test_dir):
    try:
        # Trasladar imágenes del train
        img_train_names = list(pd_train.name)
        d = pd_train.set_index('name')[['label']].to_dict()

        for im in img_train_names:
            if d[im] == 1:
                copyfile(os.path.join(source_dir_train, im), os.path.join(train_dir, 'malignant', im))
            elif d[im] == 0:
                copyfile(os.path.join(source_dir_train, im), os.path.join(train_dir, 'benign' ,im))

        # Trasladar imágenes del test
        img_test_names = list(pd_test.name)

        dt = pd_test.set_index('name')[['label']].to_dict()

        for im in img_test_names:
            if dt[im] == 1:
                copyfile(os.path.join(source_dir_test, im), os.path.join(test_dir, 'malignant', im))
            elif dt[im] == 0:
                copyfile(os.path.join(source_dir_test, im), os.path.join(test_dir, 'benign' ,im))

        print('SE TRASLADARON TODAS LAS IMÁGENES AL NUEVO DIRECTORIO!')
    except:
        print('Ocurrió un error al trasladar las imágenes... ')

```

**Figura 36.** Algoritmo de organización de imágenes.

Fuente: Elaboración propia.

```

for dir, subdir, files in os.walk('./data/finalData/'):
    print(dir)

./data/finalData/
./data/finalData/test
./data/finalData/test\benign
./data/finalData/test\malignant
./data/finalData/train
./data/finalData/train\benign
./data/finalData/train\malignant

```

**Figura 37.** Nueva distribución de ficheros para los datos.

Fuente: Elaboración propia.

### Actividad 3: Redimensionar, normalizar y dividir los datos según los requerimientos para cada modelo de Deep Learning

A partir de esta actividad, los datos usados fueron aquellos anteriormente trasladados desde los ficheros originales a los nuevos directorios.

El tamaño a redimensionar de las imágenes del conjunto de datos dependió únicamente del modelo en el que este se usó.

La normalización de los valores de píxeles de cada imagen se hizo con un media y desviación estándar de 0.5 aplicadas a cada uno de los 3 canales de las imágenes. Esto llevó a

obtener finalmente píxeles con valores dentro del rango de [-1, 1], lo cual se recomienda para realizar entrenamientos de los modelos de forma más eficiente.

Finalmente, el conjunto de datos de entrenamiento se dividió en dos partes: data de validación, el cual representa el 15 % del total, y data de entrenamiento, compuesta del 85 % restante. La distribución de los datos se hizo de forma aleatoria manteniendo los mismos porcentajes.

La cantidad de la data de prueba se mantuvo igual a como originalmente fue obtenida.

Todo este proceso se realizó a través del algoritmo presentado en la Figura 38, donde finalmente se otorgaron los tres conjuntos de datos en tipo DataLoader de Pytorch que posteriormente facilitó la ingesta de datos a los modelos.

```
def generators(train_dir, test_dir, img_size, b_size):
    # Transformaciones para las imágenes
    transform = transforms.Compose([
        transforms.Resize(img_size),
        transforms.CenterCrop(img_size),
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
    ])

    # Carga las imágenes de los directorios
    train_data = datasets.ImageFolder(train_dir, transform=transform)
    test_data = datasets.ImageFolder(test_dir, transform=transform)

    val_size = int(0.15 * len(train_data))
    train_size = len(train_data) - val_size

    train_data, val_data = random_split(train_data, [train_size, val_size])

    # Crea los generadores de datos
    train_gen = DataLoader(train_data, batch_size=b_size, shuffle=True)
    val_gen = DataLoader(val_data, batch_size=b_size, shuffle=True)
    test_gen = DataLoader(test_data, batch_size=b_size, shuffle=True)

    return train_gen, val_gen, test_gen
```

**Figura 38.** Algoritmo para lectura de los datos en DataLoader.

Fuente: Elaboración propia.

#### Actividad 4: Mejorar composición del conjunto de datos

Como se ha mencionado en la sección anterior de este capítulo, las características que determinan si un nódulo puede ser benigno o maligno en un imagen de ultrasonido impiden el uso de las técnicas convencionales de Aumento de Datos. Por este motivo, se construyó un modelo DCGAN con el objetivo de generar imágenes sintéticas.

Inicialmente, la arquitectura del modelo DCGAN siguió los mismos lineamientos defi-

nidos en la página oficial de Pytorch por Inkawich (2024); sin embargo, debido a la posible pérdida de información al reducir considerablemente el tamaño de las imágenes (ya que se pedía como entrada imágenes de 64 x 64), se decidió por personalizar el modelo original para que pueda recibir imágenes de 128 x 128, de la misma manera como se hizo en la investigación presentada por JERBI et al. (2023). Para lograr esto se tuvo que modificar tanto la red generadora como la discriminadora.

En la red generadora se agregaron un bloque nuevo de capas similar a las que ya poseída, modificando únicamente el tamaño de canales de salida de la capa ConvTranspose2d al doble de las que poseía la capa inicial original. El tamaño de kernel, stride y padding de esta misma capa se mantuvieron del formato original. En la capa de BatchNorm2d también se duplicó la cantidad de features que recibía. La función de activación ReLU se mantuvo con la misma configuración inicial. Las capas siguientes a este nuevo bloque, fueron adaptándose a la cambiada cantidad de canales de salida.

El tamaño del vector latente que ingresa a la red generadora se definió en 200, con 3 como número de canales y 128 como cantidad base de features map.

En el caso de la red discriminadora, se introdujo un bloque intermedio de capas en la parte final de todo el grupo de bloques de capas similares. La capa Conv2d de este nuevo bloque recibiría la cantidad de canales de salida del bloque anterior, mientras que sus canales de salida serían el doble de los que tenía la capa Conv2d anterior. De igual forma, la capa BatchNorm2d duplicó la cantidad de features que recibía originalmente. Se mantuvo la misma configuración de la función de activación LeakyReLU. La capa siguiente a este bloque recién creado se modificó para recibir el doble de canales de los que recibía originalmente.

La cantidad de canales que recibía el modelo se estableció en 3, mientras que el número de features map base a generar se estableció a 32.

Todas las demás configuraciones en ambas redes se dejaron por defecto.

La Figura 39 muestra la arquitectura final de la red generadora.

```

Generador(
    (main): Sequential(
        (0): ConvTranspose2d(200, 2048, kernel_size=(4, 4), stride=(1, 1), bias=False)
        (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): ConvTranspose2d(2048, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
        (4): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): ReLU(inplace=True)
        (6): ConvTranspose2d(1024, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
        (7): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (8): ReLU(inplace=True)
        (9): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
        (10): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (11): ReLU(inplace=True)
        (12): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
        (13): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (14): ReLU(inplace=True)
        (15): ConvTranspose2d(128, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
        (16): Tanh()
    )
)

```

**Figura 39.** Arquitectura de Red Generadora.

Fuente: Elaboración propia.

De igual forma, la Figura 40, muestra la arquitectura de la red discriminadora.

```

Discriminador(
    (main): Sequential(
        (0): Conv2d(3, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
        (1): LeakyReLU(negative_slope=0.2, inplace=True)
        (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
        (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (4): LeakyReLU(negative_slope=0.2, inplace=True)
        (5): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
        (6): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (7): LeakyReLU(negative_slope=0.2, inplace=True)
        (8): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
        (9): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (10): LeakyReLU(negative_slope=0.2, inplace=True)
        (11): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
        (12): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (13): LeakyReLU(negative_slope=0.2, inplace=True)
        (14): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
        (15): Sigmoid()
    )
)

```

**Figura 40.** Arquitectura de Red Discriminadora.

Fuente: Elaboración propia.

Para la función de pérdida se usó el Binary Cross Entropy Loss, mientras que la función de optimización se estableció en Adam con una tasa de aprendizaje de 0.0002 y betas de 0.5 para el primero y 0.999 para el segundo. Todo esto según lo establecido por la página oficial de Pytorch.

También se definió un tensor fijo de tamaño 200 (tamaño anteriormente definido de vector latente) inicializado con números aleatorios pero que siguen una distribución normal. Este tensor tenía un lote de tamaño 64.

Para ya iniciar con el entrenamiento del modelo DCGAN se procedió con la creación

del DataLoader de Pytorch que contenía todas las imágenes del conjunto de datos de entrenamiento. En este proceso, se definió que las imágenes deberían ser redimensionadas a tamaño 128 y con leídas en un tamaño de lote 64.

El proceso de entrenamiento siguió las características normales del DCGAN original, estableciendo únicamente la cantidad de épocas a 200.

Se usó las capacidades de la GPU NVIDIA A100-SXM4-40GB obtenido a través de Google Colab para este proceso de entrenamiento.

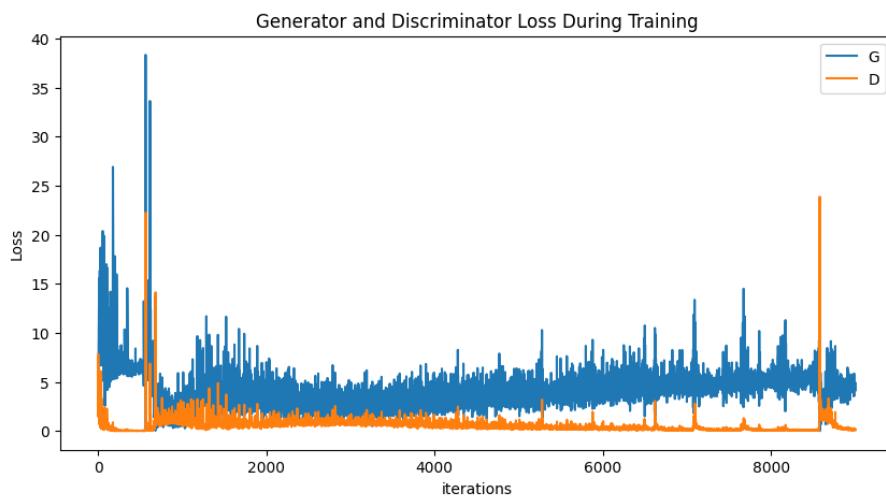
En la Figura 41 se muestran las métricas de la parte final del entrenamiento.

[193/200][12/45]	Loss_D: 0.9329	Loss_G: 4.9711	D(x): 0.8268	D(G(z)): 0.3861 / 0.0154
[193/200][24/45]	Loss_D: 1.3177	Loss_G: 3.7289	D(x): 0.4678	D(G(z)): 0.0460 / 0.0918
[193/200][36/45]	Loss_D: 1.1743	Loss_G: 3.2154	D(x): 0.5488	D(G(z)): 0.1099 / 0.1011
[194/200][0/45]	Loss_D: 0.7755	Loss_G: 6.3231	D(x): 0.8304	D(G(z)): 0.3657 / 0.0056
[194/200][12/45]	Loss_D: 0.3612	Loss_G: 4.7047	D(x): 0.8011	D(G(z)): 0.0835 / 0.0199
[194/200][24/45]	Loss_D: 0.5590	Loss_G: 4.0930	D(x): 0.7220	D(G(z)): 0.0862 / 0.0439
[194/200][36/45]	Loss_D: 0.7441	Loss_G: 5.7310	D(x): 0.8732	D(G(z)): 0.3288 / 0.0068
[195/200][0/45]	Loss_D: 0.3181	Loss_G: 5.2338	D(x): 0.9074	D(G(z)): 0.1495 / 0.0154
[195/200][12/45]	Loss_D: 0.5000	Loss_G: 3.2941	D(x): 0.7293	D(G(z)): 0.0566 / 0.0875
[195/200][24/45]	Loss_D: 0.3884	Loss_G: 3.8248	D(x): 0.8192	D(G(z)): 0.1246 / 0.0390
[195/200][36/45]	Loss_D: 0.3102	Loss_G: 5.1148	D(x): 0.9061	D(G(z)): 0.1469 / 0.0122
[196/200][0/45]	Loss_D: 0.2596	Loss_G: 5.6353	D(x): 0.9674	D(G(z)): 0.1730 / 0.0070
[196/200][12/45]	Loss_D: 0.2568	Loss_G: 5.7155	D(x): 0.9619	D(G(z)): 0.1732 / 0.0053
[196/200][24/45]	Loss_D: 0.2564	Loss_G: 4.6884	D(x): 0.9289	D(G(z)): 0.1434 / 0.0184
[196/200][36/45]	Loss_D: 0.1572	Loss_G: 4.6008	D(x): 0.9246	D(G(z)): 0.0659 / 0.0233
[197/200][0/45]	Loss_D: 0.1648	Loss_G: 4.7549	D(x): 0.9327	D(G(z)): 0.0775 / 0.0190
[197/200][12/45]	Loss_D: 0.2026	Loss_G: 4.4653	D(x): 0.9404	D(G(z)): 0.1161 / 0.0199
[197/200][24/45]	Loss_D: 0.1449	Loss_G: 4.2249	D(x): 0.9231	D(G(z)): 0.0562 / 0.0249
[197/200][36/45]	Loss_D: 0.1430	Loss_G: 4.1032	D(x): 0.9278	D(G(z)): 0.0587 / 0.0300
[198/200][0/45]	Loss_D: 0.2594	Loss_G: 5.9415	D(x): 0.9500	D(G(z)): 0.1662 / 0.0057
[198/200][12/45]	Loss_D: 0.2385	Loss_G: 3.6982	D(x): 0.8441	D(G(z)): 0.0517 / 0.0529
[198/200][24/45]	Loss_D: 0.1866	Loss_G: 4.9649	D(x): 0.9299	D(G(z)): 0.0914 / 0.0156
[198/200][36/45]	Loss_D: 0.2030	Loss_G: 4.1795	D(x): 0.9273	D(G(z)): 0.1074 / 0.0304
[199/200][0/45]	Loss_D: 0.1119	Loss_G: 4.4837	D(x): 0.9500	D(G(z)): 0.0545 / 0.0211
[199/200][12/45]	Loss_D: 0.2235	Loss_G: 4.3183	D(x): 0.8310	D(G(z)): 0.0195 / 0.0336
[199/200][24/45]	Loss_D: 0.1580	Loss_G: 4.5593	D(x): 0.9082	D(G(z)): 0.0433 / 0.0192
[199/200][36/45]	Loss_D: 0.1250	Loss_G: 4.5769	D(x): 0.9679	D(G(z)): 0.0787 / 0.0158

**Figura 41.** Parte final del entrenamiento del DCGAN.

Fuente: Elaboración propia.

Además, se realizó un plot de las pérdidas de ambas redes a través de cada iteración del entrenamiento del modelo. Este se muestra en la Figura 42.

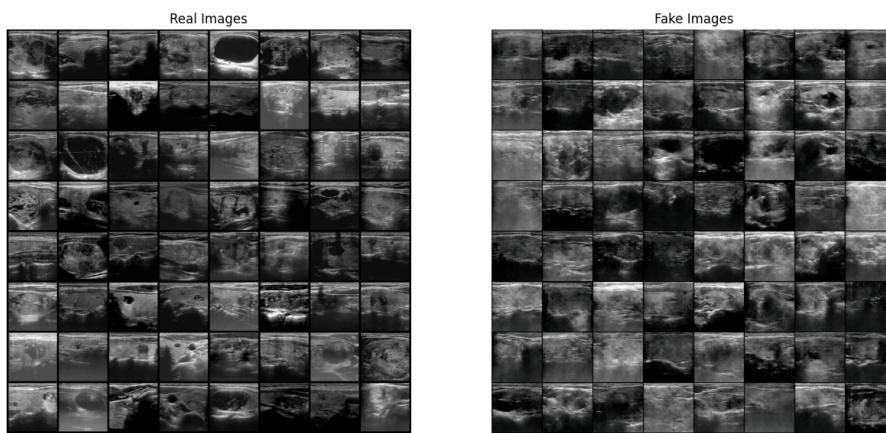


**Figura 42.** Plot de pérdidas de las red Generadora y Discriminadora.

Fuente: Elaboración propia.

En esta última gráfica se puede observar cómo ambas pérdidas van convergiendo a más iteraciones se tiene; sin embargo, ya en la parte final se logra observar la divergencia de ambos valores. Esto podría deberse a un sobreajuste por parte de la red discriminadora, al mismo tiempo que la red generadora deja de mejorar en la tarea de generar imágenes sintéticas.

Las imágenes reales y generadas se muestran en la Figura 43.



**Figura 43.** Imágenes de ultrasonido de nódulos tiroideos generados y reales.

Fuente: Elaboración propia.

Según un rápido análisis visual, las imágenes generadas sí tienen un poco de similitud a las originales, y fácilmente, para alguien no experto, estos podrían pasar como imágenes reales.

Ya que el entrenamiento se realizó con todas las imágenes la data de entrenamiento, todas las imágenes generadas podrían pertenecer a cualquiera de las clases (benigno o maligno) y se es difícil sino imposible determinar a qué tipo pertenecen (benigno o maligno). Por este motivo, y viendo el potencial del DCGAN, se volvió a entrenar el modelo ahora solo con imágenes de la clase maligno (clase minoritaria) del conjunto de datos de entrenamiento para que la red solo pueda generar imágenes de ultrasonido de esta clase en específico.

En este caso se modificó el tamaño de vector latente a 300 y los features map base a 256 en la red generadora. Las demás configuraciones se mantuvieron, incluyendo aquellas de la red discriminadora.

Se usaron la misma función de pérdida y optimización con los mismos parámetros definidos anteriormente.

El tensor fijo de números aleatorio de distribución normal tuvo para este entrenamiento un tamaño de 300 (igual al tamaño del vector latente).

Para el DataLoader se mantuvo el mismo tamaño de redimensionamiento y lote.

Debido a una prueba rápida inicial con una menor cantidad de épocas, donde se obtuvieron pobres resultados, se decidió aumentar la cantidad de iteraciones totales estableciendo el número de épocas a 370.

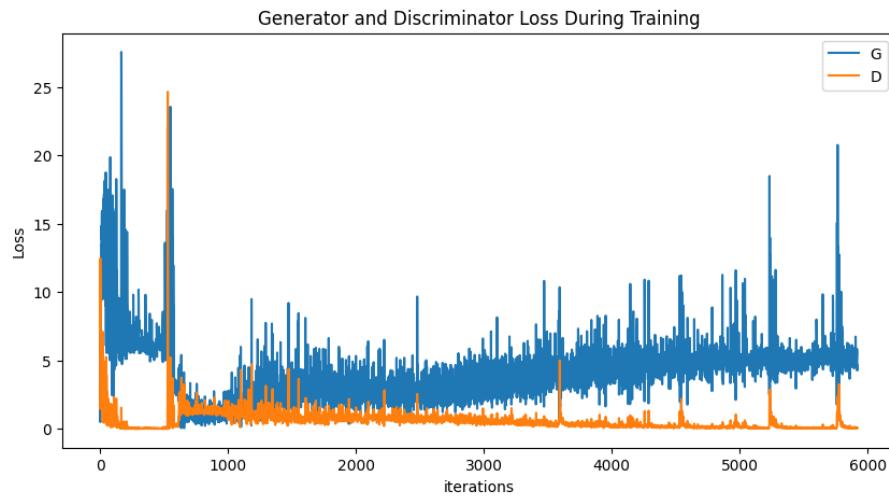
La Figura 44 muestra la parte final del entrenamiento.

```
[357/370][12/16] Loss_D: 0.0833 Loss_G: 5.5940 D(x): 0.9683 D(G(z)): 0.0461 / 0.0082
[358/370][0/16] Loss_D: 0.2755 Loss_G: 9.7520 D(x): 0.9916 D(G(z)): 0.2037 / 0.0002
[358/370][12/16] Loss_D: 0.0470 Loss_G: 4.3363 D(x): 0.9884 D(G(z)): 0.0339 / 0.0186
[359/370][0/16] Loss_D: 0.0375 Loss_G: 5.2440 D(x): 0.9854 D(G(z)): 0.0221 / 0.0124
[359/370][12/16] Loss_D: 0.0677 Loss_G: 4.9654 D(x): 0.9722 D(G(z)): 0.0369 / 0.0126
[360/370][0/16] Loss_D: 0.9652 Loss_G: 15.0713 D(x): 0.9989 D(G(z)): 0.4774 / 0.0000
[360/370][12/16] Loss_D: 0.6778 Loss_G: 8.4812 D(x): 0.8195 D(G(z)): 0.2315 / 0.0022
[361/370][0/16] Loss_D: 3.2693 Loss_G: 12.7542 D(x): 0.9747 D(G(z)): 0.7195 / 0.0001
[361/370][12/16] Loss_D: 0.3831 Loss_G: 5.9367 D(x): 0.7973 D(G(z)): 0.0664 / 0.0114
[362/370][0/16] Loss_D: 0.5647 Loss_G: 10.0312 D(x): 0.9745 D(G(z)): 0.3305 / 0.0002
[362/370][12/16] Loss_D: 0.3276 Loss_G: 7.1568 D(x): 0.9746 D(G(z)): 0.2275 / 0.0023
[363/370][0/16] Loss_D: 0.2405 Loss_G: 6.0655 D(x): 0.9381 D(G(z)): 0.1303 / 0.0055
[363/370][12/16] Loss_D: 0.1617 Loss_G: 5.3671 D(x): 0.9200 D(G(z)): 0.0540 / 0.0104
[364/370][0/16] Loss_D: 0.1563 Loss_G: 5.4059 D(x): 0.9934 D(G(z)): 0.1257 / 0.0088
[364/370][12/16] Loss_D: 0.1329 Loss_G: 5.7556 D(x): 0.9664 D(G(z)): 0.0861 / 0.0076
[365/370][0/16] Loss_D: 0.1250 Loss_G: 5.2169 D(x): 0.9763 D(G(z)): 0.0872 / 0.0103
[365/370][12/16] Loss_D: 0.0713 Loss_G: 5.1367 D(x): 0.9475 D(G(z)): 0.0139 / 0.0126
[366/370][0/16] Loss_D: 0.0458 Loss_G: 5.1208 D(x): 0.9712 D(G(z)): 0.0158 / 0.0105
[366/370][12/16] Loss_D: 0.0569 Loss_G: 4.9588 D(x): 0.9817 D(G(z)): 0.0365 / 0.0112
[367/370][0/16] Loss_D: 0.0961 Loss_G: 5.2228 D(x): 0.9641 D(G(z)): 0.0523 / 0.0113
[367/370][12/16] Loss_D: 0.0822 Loss_G: 4.7507 D(x): 0.9696 D(G(z)): 0.0486 / 0.0124
[368/370][0/16] Loss_D: 0.0608 Loss_G: 4.8219 D(x): 0.9706 D(G(z)): 0.0290 / 0.0132
[368/370][12/16] Loss_D: 0.0572 Loss_G: 4.9233 D(x): 0.9681 D(G(z)): 0.0237 / 0.0129
[369/370][0/16] Loss_D: 0.0351 Loss_G: 5.3250 D(x): 0.9875 D(G(z)): 0.0219 / 0.0126
[369/370][12/16] Loss_D: 0.0829 Loss_G: 5.8338 D(x): 0.9801 D(G(z)): 0.0581 / 0.0070
```

**Figura 44.** Parte final del entrenamiento del DCGAN con imágenes de nódulos malignos.

Fuente: Elaboración propia.

De igual manera al entrenamiento inicial, se realizó una gráfica donde se muestra el comportamiento de la pérdida en ambas redes. Esto se muestra en la Figura 45.

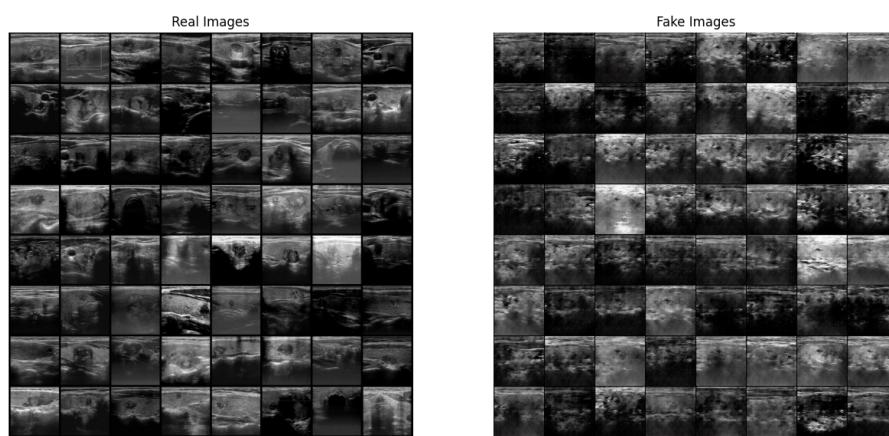


**Figura 45.** Plot de pérdidas de las red Generadora y Discriminadora de imágenes de nódulos malignos.

Fuente: Elaboración propia.

Esta gráfica muestra el mismo comportamiento que a más iteraciones, mayor divergencia entre las pérdidas.

En la Figura 46 se presentan las imágenes generadas junto a las originales.



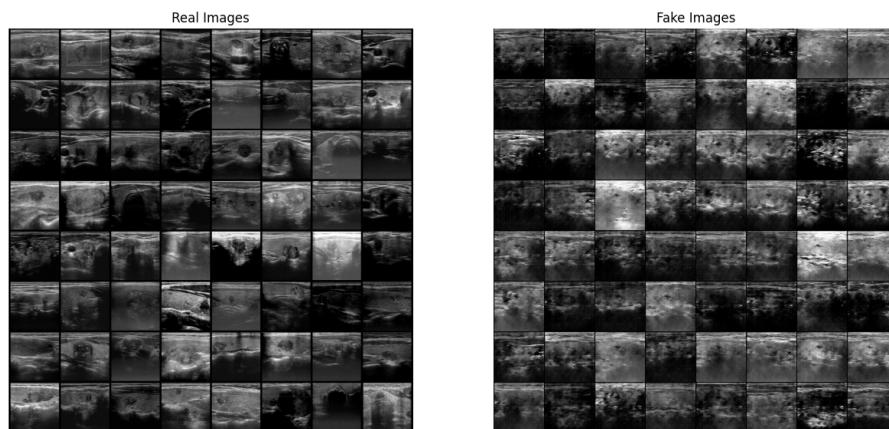
**Figura 46.** Imágenes de ultrasonido de nódulos tiroideos generados y reales (DCGAN de imágenes de nódulos malignos).

Fuente: Elaboración propia.

Los resultados de las imágenes generadas en este modelo muestran mayor ruido comparado las generadas con el entrenamiento inicial, esto se debe principalmente a la menor cantidad de datos usada para el entrenamiento de este nuevo DCGAN. Sin embargo, luego de un rápido análisis visual, se puede observar un comportamiento similar entre todas estas imágenes nuevas, y es que en la mayoría de estas hay lo que parece ser la presencia de un nódulo, aunque difuminado con mucho ruido aleatorio, esto podría indicar un aprendizaje cercano a la distribución de las imágenes de nódulos malignos.

Intentando utilizar el potencial que ofrece el DCGAN, se realizó posteriormente dos tipos de entrenamiento a los modelos de Deep Learning: el primero sin el Aumento de Datos a través de DCGAN, mientras que el segundo combinará los datos generados con la data original de entrenamiento para realizar dicho proceso. Esto permitirá determinar si las imágenes generadas por el DCGAN, aunque con ruido, son capaces de aumentar la capacidad de predicción de los modelos.

La gráfica circula presentada en la Figura 47 muestra el nuevo balance entre ambas clases en la nueva data de entrenamiento.



**Figura 47.** Gráfica circula de data de entrenamiento balanceada.

Fuente: Elaboración propia.

Ya con la data preprocessada, se inició con la etapa de Aplicación de Deep Learning donde se entrenó, validó y puso a prueba los diversos modelos CNN y ViT. Esta etapa inicia con la siguiente actividad.

### **Actividad 1: Seleccionar algoritmos a entrenar según usos y resultados de las investigaciones presentadas en los antecedentes**

Según los datos analizados en las investigaciones presentadas en el Capítulo 2, el modelo con mayor presencia de los CNN es el ResNet 50, mientras que por el lado de los modelos ViT, los

más usados son los ViT-Base con tamaño de patch de 16 x 16.

Además, debido al tipo de problema de clasificación presente en esta investigación, y la similitud con el trabajo de JERBI et al. (2023), se decidió por probar con las redes VGG16, ya que ha demostrado ser capaz de obtener un alto desempeño.

En resumen, se ha optado por utilizar las arquitecturas CNN de ResNet50 y VGG16. Además, como parte de las arquitecturas ViT, se optó por el uso de ViT-Base (16).

### **Actividad 2: Entrenar y probar los modelos de Deep Learning previamente definidos**

Para todos los modelos en esta actividad se usó la capacidad de la GPU NVIDIA A100-SXM4-40GB obtenido a través de Google Colab para lograr un rápido y eficiente entrenamiento.

En este primer grupo, no se utilizará el conjunto de datos con imágenes generadas.

Se empezó con los modelos ViT-Base (16).

Para este proceso se crearon los DataLoader correspondientes con las características requeridas por estos modelos ViT: se redimensionaron las imágenes a 224 x 224 y se definió un tamaño de lote de 64 (este último tamaño será constante para todos los modelos mostrados en esta actividad).

En este primer modelo, se usó la técnica de entrenamiento de Fine Tuning junto con el modelo ViT-Base (16) con pesos de IMAGENET1KV1.

La función de pérdida aplicada fue el BCEWithLogitsLoss que evita el uso manual de la sigmoide y aplica lo directamente con la entropía cruzada. Nuevamente, esta función será utilizada en cada uno de los modelos en esta actividad.

Para lograr una clasificación binaria, la parte superior de estas redes debió ser reemplazada por una capa lineal que recibe todos los features del modelo y produce una única salida.

Finalmente, la función de optimización utilizada fue Adam con una tasa de aprendizaje de 0.0001, y se definió para el entrenamiento 20 épocas. Las demás configuraciones se dejaron por defecto.

Estas configuraciones se aplicarán de igual manera a los demás modelos en esta actividad.

A continuación se presenta en la Figura 48 la parte final del proceso de entrenamiento del modelo ViT-Base (16) con los pesos IMAGENET1KV1 y usando la técnica de Fine Tuning.

```

Epoch: 16/20 Loss: 0.018599607050418854
Epoch: 16/20 Loss: 0.04214214161038399
Epoch: 16/20 Loss: 0.2229091376066208
Accuracy TRAIN: 97.30392156862744%
Accuracy EVAL: 72.8538283062645%
Epoch: 17/20 Loss: 0.007329778280109167
Epoch: 17/20 Loss: 0.010187733918428421
Epoch: 17/20 Loss: 0.003171817399561405
Accuracy TRAIN: 98.77450980392157%
Accuracy EVAL: 73.54988399071925%
Epoch: 18/20 Loss: 0.05616264045238495
Epoch: 18/20 Loss: 0.005840306170284748
Epoch: 18/20 Loss: 0.003426502225920558
Accuracy TRAIN: 99.46895424836602%
Accuracy EVAL: 73.31786542923435%
Epoch: 19/20 Loss: 0.015321901999413967
Epoch: 19/20 Loss: 0.005017929710447788
Epoch: 19/20 Loss: 0.007157755084335804
Accuracy TRAIN: 99.22385620915033%
Accuracy EVAL: 76.33410672853829%
Epoch: 20/20 Loss: 0.0019829161465168
Epoch: 20/20 Loss: 0.0038447368424385786
Epoch: 20/20 Loss: 0.0018818116514012218
Accuracy TRAIN: 99.06045751633987%
Accuracy EVAL: 74.24593967517401%
Accuracy TEST: 62.37785016286645%

```

**Figura 48.** Proceso de entrenamiento del modelo n°1.

Fuente: Elaboración propia.

Finalmente, para este modelo también se calcularon su Recall y Precision. Estos son presentados en la Figura 49.

```

r, p = get_recall_and_precision(model=vitb16_net1kv1, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.7711864406779662      PRECISION: 0.5069637883008357

```

**Figura 49.** Recall y Precision del modelo n°1.

Fuente: Elaboración propia.

En el siguiente modelo, se usó la técnica de entrenamiento de Transfer Learning junto con el modelo ViT-Base (16) con pesos de IMAGENET1KV1.

El DataLoader constaba de las imágenes redimensionadas nuevamente a 224 x 224, y se

aplicó la función de pérdida BCEWithLogitsLoss junto con la función de optimización Adam.

A continuación se presenta en la Figura 50 la parte final del proceso de entrenamiento del modelo ViT-Base (16) con los pesos IMAGENET1KV1 y usando la técnica de Transfer Learning.

```
Epoch: 16/20 Loss: 0.530026912689209
Epoch: 16/20 Loss: 0.5008851885795593
Epoch: 16/20 Loss: 0.5877339839935303
Accuracy TRAIN: 73.32516339869281%
Accuracy EVAL: 70.06960556844548%
Epoch: 17/20 Loss: 0.5817161798477173
Epoch: 17/20 Loss: 0.6772599816322327
Epoch: 17/20 Loss: 0.5808808207511902
Accuracy TRAIN: 73.2843137254902%
Accuracy EVAL: 70.53364269141531%
Epoch: 18/20 Loss: 0.5176637768745422
Epoch: 18/20 Loss: 0.545374870300293
Epoch: 18/20 Loss: 0.524877667427063
Accuracy TRAIN: 73.48856209150327%
Accuracy EVAL: 70.99767981438515%
Epoch: 19/20 Loss: 0.5568345785140991
Epoch: 19/20 Loss: 0.5816813111305237
Epoch: 19/20 Loss: 0.5614938735961914
Accuracy TRAIN: 73.8970588235294%
Accuracy EVAL: 71.22969837587007%
Epoch: 20/20 Loss: 0.5740368366241455
Epoch: 20/20 Loss: 0.518339216709137
Epoch: 20/20 Loss: 0.5050729513168335
Accuracy TRAIN: 73.93790849673202%
Accuracy EVAL: 71.22969837587007%
Accuracy TEST: 66.77524429967427%
```

**Figura 50.** Proceso de entrenamiento del modelo n°2.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 51.

```
r, p = get_recall_and_precision(model=vitb16_net1kv1_t1, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')

RECALL: 0.4110169491525424      PRECISION: 0.5987654320987654
```

**Figura 51.** Recall y Precision del modelo n°2.

Fuente: Elaboración propia.

Este modelo usó la técnica de entrenamiento de Fine Tuning junto con el modelo ViT-Base (16) con pesos de IMAGENET1KSWAGE2EV1.

En este caso, el DataLoader constaba de las imágenes redimensionadas a 384 x 384.

Se presenta en la Figura 52 la parte final del proceso de entrenamiento del modelo ViT-Base (16) con los pesos IMAGENET1KSWAGE2EV1 y usando la técnica de Fine Tuning.

```
Epoch: 16/20 Loss: 0.283130943775177
Epoch: 16/20 Loss: 0.2402610182762146
Epoch: 16/20 Loss: 0.22038871049880981
Accuracy TRAIN: 87.7859477124183%
Accuracy EVAL: 67.51740139211137%
Epoch: 17/20 Loss: 0.24309808015823364
Epoch: 17/20 Loss: 0.341283917427063
Epoch: 17/20 Loss: 0.11790888011455536
Accuracy TRAIN: 90.52287581699346%
Accuracy EVAL: 65.19721577726219%
Epoch: 18/20 Loss: 0.1338098645210266
Epoch: 18/20 Loss: 0.22880220413208008
Epoch: 18/20 Loss: 0.26838237047195435
Accuracy TRAIN: 91.83006535947712%
Accuracy EVAL: 67.05336426914153%
Epoch: 19/20 Loss: 0.17114253342151642
Epoch: 19/20 Loss: 0.281033456325531
Epoch: 19/20 Loss: 0.1961381733417511
Accuracy TRAIN: 93.05555555555556%
Accuracy EVAL: 61.02088167053364%
Epoch: 20/20 Loss: 0.10676979273557663
Epoch: 20/20 Loss: 0.07043120265007019
Epoch: 20/20 Loss: 0.12087588757276535
Accuracy TRAIN: 94.8529411764706%
Accuracy EVAL: 62.87703016241299%
Accuracy TEST: 60.26058631921824%
```

**Figura 52.** Proceso de entrenamiento del modelo n°3.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 53.

```
r, p = get_recall_and_precision(model=vitb16_net1k_swag_v1, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.3516949152542373
PRECISION: 0.47701149425287354
```

**Figura 53.** Recall y Precision del modelo n°3.

Fuente: Elaboración propia.

El modelo n°4 empleó la técnica de entrenamiento de Transfer Learning junto con el modelo ViT-Base (16) con pesos de IMAGENET1KSWAGE2EV1.

En este caso, nuevamente el DataLoader constaba de las imágenes redimensionadas a 384 x 384.

Se presenta en la Figura 54 la parte final del proceso de entrenamiento del modelo ViT-Base (16) con los pesos IMAGENET1KSWAGE2EV1 y usando la técnica de Transfer Learning.

```
Epoch: 16/20 Loss: 0.5322492122650146
Epoch: 16/20 Loss: 0.46538013219833374
Epoch: 16/20 Loss: 0.47468578815460205
Accuracy TRAIN: 71.97712418300654%
Accuracy EVAL: 70.99767981438515%
Epoch: 17/20 Loss: 0.5311040878295898
Epoch: 17/20 Loss: 0.5346704721450806
Epoch: 17/20 Loss: 0.5077158212661743
Accuracy TRAIN: 72.30392156862744%
Accuracy EVAL: 70.53364269141531%
Epoch: 18/20 Loss: 0.49707743525505066
Epoch: 18/20 Loss: 0.5542110800743103
Epoch: 18/20 Loss: 0.5227112174034119
Accuracy TRAIN: 72.2222222222223%
Accuracy EVAL: 71.46171693735499%
Epoch: 19/20 Loss: 0.5346338748931885
Epoch: 19/20 Loss: 0.5712888240814209
Epoch: 19/20 Loss: 0.4370526075363159
Accuracy TRAIN: 72.9983660130719%
Accuracy EVAL: 71.69373549883991%
Epoch: 20/20 Loss: 0.54349684715271
Epoch: 20/20 Loss: 0.6065188646316528
Epoch: 20/20 Loss: 0.48935550451278687
Accuracy TRAIN: 72.79411764705883%
Accuracy EVAL: 71.46171693735499%
Accuracy TEST: 68.40390879478828%
```

**Figura 54.** Proceso de entrenamiento del modelo n°4.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 55.

```
r, p = get_recall_and_precision(model=vitb16_net1k_swag_v1_t1, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.4110169491525424      PRECISION: 0.6381578947368421
```

**Figura 55.** Recall y Precision del modelo n°4.

Fuente: Elaboración propia.

El modelo n°5 empleó la técnica de entrenamiento de Fine Tuning junto con el modelo ViT-Base (16) con pesos de IMAGENET1KSWAGLINEARV1.

En este caso, el DataLoader constaba de las imágenes redimensionadas a 224 x 224.

Se presenta en la Figura 56 la parte final del entrenamiento del modelo ViT-Base (16) con los pesos IMAGENET1KSWAGLINEARV1 y usando la técnica de Fine Tuning.

```
Epoch: 16/20 Loss: 0.4894776940345764
Epoch: 16/20 Loss: 0.35874974727630615
Epoch: 16/20 Loss: 0.419121116399765
Accuracy TRAIN: 77.77777777777777%
Accuracy EVAL: 65.4292343387471%
Epoch: 17/20 Loss: 0.44508349895477295
Epoch: 17/20 Loss: 0.5296388864517212
Epoch: 17/20 Loss: 0.4508635401725769
Accuracy TRAIN: 77.5326797385621%
Accuracy EVAL: 61.25290023201856%
Epoch: 18/20 Loss: 0.4076199531555176
Epoch: 18/20 Loss: 0.320592999458313
Epoch: 18/20 Loss: 0.49208664894104004
Accuracy TRAIN: 81.20915032679738%
Accuracy EVAL: 66.12529002320186%
Epoch: 19/20 Loss: 0.21434906125068665
Epoch: 19/20 Loss: 0.4003680944442749
Epoch: 19/20 Loss: 0.4300868809223175
Accuracy TRAIN: 82.31209150326798%
Accuracy EVAL: 59.86078886310905%
Epoch: 20/20 Loss: 0.4742778539657593
Epoch: 20/20 Loss: 0.2638590931892395
Epoch: 20/20 Loss: 0.3958551585674286
Accuracy TRAIN: 84.84477124183006%
Accuracy EVAL: 59.39675174013921%
Accuracy TEST: 54.88599348534202%
```

**Figura 56.** Proceso de entrenamiento del modelo n°5.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 57.

```
r, p = get_recall_and_precision(model=vitb16_net1k_swag_linear, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.673728813559322      PRECISION: 0.4428969359331476
```

**Figura 57.** Recall y Precision del modelo n°5.

Fuente: Elaboración propia.

El modelo n°6 empleó la técnica de entrenamiento de Transfer Learning junto con el

modelo ViT-Base (16) con pesos de IMAGENET1KSWAGLINEARV1.

En este caso, el DataLoader constaba nuevamente de las imágenes redimensionadas a 224 x 224.

Se presenta en la Figura 58 la parte final del entrenamiento del modelo ViT-Base (16) con los pesos IMAGENET1KSWAGLINEARV1 y usando la técnica de Transfer Learning.

```
Epoch: 16/20 Loss: 0.5694300532341003
Epoch: 16/20 Loss: 0.642823338508606
Epoch: 16/20 Loss: 0.5589892864227295
Accuracy TRAIN: 69.28104575163398%
Accuracy EVAL: 69.1415313225058%
Epoch: 17/20 Loss: 0.5747979283332825
Epoch: 17/20 Loss: 0.5256456732749939
Epoch: 17/20 Loss: 0.5795984268188477
Accuracy TRAIN: 69.11764705882354%
Accuracy EVAL: 69.83758700696056%
Epoch: 18/20 Loss: 0.5791229009628296
Epoch: 18/20 Loss: 0.5886937379837036
Epoch: 18/20 Loss: 0.5171201825141907
Accuracy TRAIN: 69.77124183006536%
Accuracy EVAL: 69.60556844547564%
Epoch: 19/20 Loss: 0.5332614183425903
Epoch: 19/20 Loss: 0.6141523122787476
Epoch: 19/20 Loss: 0.568691611289978
Accuracy TRAIN: 69.3218954248366%
Accuracy EVAL: 70.06960556844548%
Epoch: 20/20 Loss: 0.6115080118179321
Epoch: 20/20 Loss: 0.598221480846405
Epoch: 20/20 Loss: 0.6333624124526978
Accuracy TRAIN: 69.48529411764706%
Accuracy EVAL: 70.53364269141531%
Accuracy TEST: 63.843648208469055%
```

**Figura 58.** Proceso de entrenamiento del modelo nº6.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 59.

```
r, p = get_recall_and_precision(model=vitb16_net1k_swag_linear_t1, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')

RECALL: 0.2584745762711864      PRECISION: 0.5648148148148148
```

**Figura 59.** Recall y Precision del modelo nº6.

Fuente: Elaboración propia.

El modelo nº7 empleó la técnica de entrenamiento de Fine Tuning junto con el modelo VGG16 con pesos de IMAGENET1KV1.

En este caso, el DataLoader fue cargado con imágenes redimensionadas a 128 x 128. Este tamaño será constante para los demás modelos CNN.

Se presenta en la Figura 60 la parte final del entrenamiento del modelo VGG16 con los pesos IMAGENET1KV1 y usando la técnica de Fine Tuning.

```
Epoch: 16/20 Loss: 0.004385572858154774
Epoch: 16/20 Loss: 0.006463947240263224
Epoch: 16/20 Loss: 0.0026259245350956917
Accuracy TRAIN: 99.01960784313725%
Accuracy EVAL: 73.78190255220417%
Epoch: 17/20 Loss: 0.0010360963642597198
Epoch: 17/20 Loss: 0.00457008508965373
Epoch: 17/20 Loss: 0.0554203987121582
Accuracy TRAIN: 99.63235294117646%
Accuracy EVAL: 76.56612529002321%
Epoch: 18/20 Loss: 0.0008224823977798223
Epoch: 18/20 Loss: 0.04864876717329025
Epoch: 18/20 Loss: 0.0021803402341902256
Accuracy TRAIN: 99.59150326797386%
Accuracy EVAL: 77.26218097447796%
Epoch: 19/20 Loss: 0.00017787513206712902
Epoch: 19/20 Loss: 0.0004122729878872633
Epoch: 19/20 Loss: 0.0003881763550452888
Accuracy TRAIN: 99.83660130718954%
Accuracy EVAL: 74.47795823665894%
Epoch: 20/20 Loss: 0.04901563376188278
Epoch: 20/20 Loss: 0.0008530798368155956
Epoch: 20/20 Loss: 0.0007317373529076576
Accuracy TRAIN: 99.83660130718954%
Accuracy EVAL: 76.33410672853829%
Accuracy TEST: 69.86970684039088%
```

**Figura 60.** Proceso de entrenamiento del modelo nº7.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 61.

```
r, p = get_recall_and_precision(model=cnnvgg16_net1k_v1, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')

RECALL: 0.7245762711864406      PRECISION: 0.5876288659793815
```

**Figura 61.** Recall y Precision del modelo n°7.

Fuente: Elaboración propia.

El modelo n°8 empleó la técnica de entrenamiento de Transfer Learning junto con el modelo VGG16 con pesos de IMAGENET1KV1.

Se presenta en la Figura 62 la parte final del entrenamiento del modelo VGG16 con los pesos IMAGENET1KV1 y usando la técnica de Transfer Learning.

```
Epoch: 16/20 Loss: 0.02932560257613659
Epoch: 16/20 Loss: 0.00912750605493784
Epoch: 16/20 Loss: 0.007482304237782955
Accuracy TRAIN: 99.22385620915033%
Accuracy EVAL: 74.47795823665894%
Epoch: 17/20 Loss: 0.007980391383171082
Epoch: 17/20 Loss: 0.02309158258140087
Epoch: 17/20 Loss: 0.020087245851755142
Accuracy TRAIN: 99.34640522875817%
Accuracy EVAL: 73.54988399071925%
Epoch: 18/20 Loss: 0.015662919729948044
Epoch: 18/20 Loss: 0.00492052361369133
Epoch: 18/20 Loss: 0.018227143213152885
Accuracy TRAIN: 99.63235294117646%
Accuracy EVAL: 74.47795823665894%
Epoch: 19/20 Loss: 0.006181559059768915
Epoch: 19/20 Loss: 0.00617906404659152
Epoch: 19/20 Loss: 0.020157750695943832
Accuracy TRAIN: 98.93790849673202%
Accuracy EVAL: 74.47795823665894%
Epoch: 20/20 Loss: 0.0009541739709675312
Epoch: 20/20 Loss: 0.016870316118001938
Epoch: 20/20 Loss: 0.01126136165112257
Accuracy TRAIN: 99.50980392156863%
Accuracy EVAL: 74.70997679814386%
Accuracy TEST: 66.28664495114006%
```

**Figura 62.** Proceso de entrenamiento del modelo n°8.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 63.

```
r, p = get_recall_and_precision(model=cnnvgg16_net1k_v1_t1, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')

RECALL: 0.559322033898305      PRECISION: 0.5617021276595745
```

**Figura 63.** Recall y Precision del modelo n°8.

Fuente: Elaboración propia.

El modelo n°9 empleó la técnica de entrenamiento de Fine Tuning junto con el modelo RESNET50 con pesos de IMAGENET1KV1.

Se presenta en la Figura 64 la parte final del entrenamiento del modelo RESNET50 con los pesos IMAGENET1KV1 y usando la técnica de Fine Tuning.

```
Epoch: 16/20 Loss: 0.005303689744323492
Epoch: 16/20 Loss: 0.022068902850151062
Epoch: 16/20 Loss: 0.040470317006111145
Accuracy TRAIN: 98.69281045751634%
Accuracy EVAL: 73.78190255220417%
Epoch: 17/20 Loss: 0.001320847775787115
Epoch: 17/20 Loss: 0.03172039985656738
Epoch: 17/20 Loss: 0.00672328844666481
Accuracy TRAIN: 99.18300653594771%
Accuracy EVAL: 76.56612529002321%
Epoch: 18/20 Loss: 0.001990809105336666
Epoch: 18/20 Loss: 0.0045721144415438175
Epoch: 18/20 Loss: 0.002151206135749817
Accuracy TRAIN: 99.59150326797386%
Accuracy EVAL: 77.95823665893272%
Epoch: 19/20 Loss: 0.0012045479379594326
Epoch: 19/20 Loss: 0.020784474909305573
Epoch: 19/20 Loss: 0.007452369201928377
Accuracy TRAIN: 99.30555555555556%
Accuracy EVAL: 75.17401392111368%
Epoch: 20/20 Loss: 0.007671455852687359
Epoch: 20/20 Loss: 0.0017759010661393404
Epoch: 20/20 Loss: 0.018771717324852943
Accuracy TRAIN: 99.79575163398692%
Accuracy EVAL: 77.95823665893272%
Accuracy TEST: 72.47557003257329%
```

**Figura 64.** Proceso de entrenamiento del modelo n°9.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 65.

```
r, p = get_recall_and_precision(model=resnet50_net1k_v1, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')

RECALL: 0.6440677966101694      PRECISION: 0.6413502109704642
```

**Figura 65.** Recall y Precision del modelo n°9.

Fuente: Elaboración propia.

El modelo n°10 empleó la técnica de entrenamiento de Transfer Learning junto con el modelo RESNET50 con pesos de IMAGENET1KV1.

Se presenta en la Figura 66 la parte final del entrenamiento del modelo RESNET50 con los pesos IMAGENET1KV1 y usando la técnica de Transfer Learning.

```
Epoch: 16/20 Loss: 0.5105572938919067
Epoch: 16/20 Loss: 0.611905038356781
Epoch: 16/20 Loss: 0.594752311706543
Accuracy TRAIN: 71.48692810457516%
Accuracy EVAL: 67.51740139211137%
Epoch: 17/20 Loss: 0.5738492012023926
Epoch: 17/20 Loss: 0.599168062210083
Epoch: 17/20 Loss: 0.5019896030426025
Accuracy TRAIN: 71.48692810457516%
Accuracy EVAL: 68.67749419953596%
Epoch: 18/20 Loss: 0.5633417963981628
Epoch: 18/20 Loss: 0.6442838907241821
Epoch: 18/20 Loss: 0.6293825507164001
Accuracy TRAIN: 71.11928104575163%
Accuracy EVAL: 68.21345707656613%
Epoch: 19/20 Loss: 0.5429855585098267
Epoch: 19/20 Loss: 0.5822341442108154
Epoch: 19/20 Loss: 0.5251281261444092
Accuracy TRAIN: 71.11928104575163%
Accuracy EVAL: 67.51740139211137%
Epoch: 20/20 Loss: 0.5885058641433716
Epoch: 20/20 Loss: 0.5934745073318481
Epoch: 20/20 Loss: 0.5278187990188599
Accuracy TRAIN: 71.609477124183%
Accuracy EVAL: 67.98143851508121%
Accuracy TEST: 63.51791530944625%
```

**Figura 66.** Proceso de entrenamiento del modelo n°10.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 67.

```
r, p = get_recall_and_precision(model=resnet50_net1k_v1_t1, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')

RECALL: 0.19491525423728814      PRECISION: 0.575
```

**Figura 67.** Recall y Precision del modelo n°10.

Fuente: Elaboración propia.

El modelo n°11 empleó la técnica de entrenamiento de Fine Tuning junto con el modelo RESNET50 con pesos de IMAGENET1KV2.

Se presenta en la Figura 68 la parte final del entrenamiento del modelo RESNET50 con los pesos IMAGENET1KV2 y usando la técnica de Fine Tuning.

```
Epoch: 16/20 Loss: 0.005224029533565044
Epoch: 16/20 Loss: 0.001033734530210495
Epoch: 16/20 Loss: 0.002157044131308794
Accuracy TRAIN: 99.95915032679738%
Accuracy EVAL: 74.94199535962878%
Epoch: 17/20 Loss: 0.02650008350610733
Epoch: 17/20 Loss: 0.03744857758283615
Epoch: 17/20 Loss: 0.001588819082826376
Accuracy TRAIN: 99.83660130718954%
Accuracy EVAL: 74.24593967517401%
Epoch: 18/20 Loss: 0.0007790038362145424
Epoch: 18/20 Loss: 0.0012950035743415356
Epoch: 18/20 Loss: 0.0019384194165468216
Accuracy TRAIN: 99.83660130718954%
Accuracy EVAL: 74.47795823665894%
Epoch: 19/20 Loss: 0.0021631657145917416
Epoch: 19/20 Loss: 0.0006150682456791401
Epoch: 19/20 Loss: 0.0010430838447064161
Accuracy TRAIN: 99.95915032679738%
Accuracy EVAL: 75.87006960556845%
Epoch: 20/20 Loss: 0.001992696663364768
Epoch: 20/20 Loss: 0.0019480081973597407
Epoch: 20/20 Loss: 0.0025459127500653267
Accuracy TRAIN: 99.87745098039215%
Accuracy EVAL: 76.10208816705337%
Accuracy TEST: 69.54397394136808%
```

**Figura 68.** Proceso de entrenamiento del modelo n°11.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 69.

```
r, p = get_recall_and_precision(model=resnet50_net1k_v2, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')

RECALL: 0.5211864406779662      PRECISION: 0.6243654822335025
```

**Figura 69.** Recall y Precision del modelo n°11.

Fuente: Elaboración propia.

El modelo n°12 empleó la técnica de entrenamiento de Transfer Learning junto con el modelo RESNET50 con pesos de IMAGENET1KV2.

Se presenta en la Figura 70 la parte final del entrenamiento del modelo RESNET50 con los pesos IMAGENET1KV2 y usando la técnica de Transfer Learning.

```
Epoch: 16/20 Loss: 0.5811125040054321
Epoch: 16/20 Loss: 0.5049836039543152
Epoch: 16/20 Loss: 0.5251306295394897
Accuracy TRAIN: 69.48529411764706%
Accuracy EVAL: 67.05336426914153%
Epoch: 17/20 Loss: 0.5070241093635559
Epoch: 17/20 Loss: 0.5908627510070801
Epoch: 17/20 Loss: 0.6310464143753052
Accuracy TRAIN: 70.05718954248366%
Accuracy EVAL: 67.51740139211137%
Epoch: 18/20 Loss: 0.5192102193832397
Epoch: 18/20 Loss: 0.5765117406845093
Epoch: 18/20 Loss: 0.5636102557182312
Accuracy TRAIN: 70.38398692810458%
Accuracy EVAL: 67.05336426914153%
Epoch: 19/20 Loss: 0.5420467853546143
Epoch: 19/20 Loss: 0.5875532031059265
Epoch: 19/20 Loss: 0.5010385513305664
Accuracy TRAIN: 70.58823529411765%
Accuracy EVAL: 68.44547563805105%
Epoch: 20/20 Loss: 0.5328229665756226
Epoch: 20/20 Loss: 0.6016045808792114
Epoch: 20/20 Loss: 0.5663096904754639
Accuracy TRAIN: 70.71078431372548%
Accuracy EVAL: 68.67749419953596%
Accuracy TEST: 63.02931596091205%
```

**Figura 70.** Proceso de entrenamiento del modelo n°12.

Fuente: Elaboración propia.

También se calcularon su Recall y Precision. Estos son presentados en la Figura 71.

```
r, p = get_recall_and_precision(model=resnet50_net1k_v2_t1, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.2076271186440678      PRECISION: 0.550561797752809
```

**Figura 71.** Recall y Precision del modelo n°12.

Fuente: Elaboración propia.

Una vez culminado el entrenamiento de los modelos con el conjunto de datos sin imágenes generadas, se inició un nuevo proceso de entrenamiento, esta vez con el conjunto de datos aumentado.

Todos los modelos, tanto ViT como CNN, poseen las mismas configuraciones definidas antes, así que no hay necesidad de especificarlo nuevamente.

A continuación se presentan las siguientes Figuras donde se muestran todos los resultados de los modelos de manera similar a como se presentó anteriormente.

```
Epoch: 17/20 Loss: 0.0018437945982441306
Epoch: 17/20 Loss: 0.017262987792491913
Epoch: 17/20 Loss: 0.1247384250164032
Epoch: 17/20 Loss: 0.004013187251985073
Accuracy TRAIN: 99.32077801790676%
Accuracy EVAL: 83.53765323992995%
Epoch: 18/20 Loss: 0.0042344992980360985
Epoch: 18/20 Loss: 0.0008112872601486742
Epoch: 18/20 Loss: 0.0014449895825237036
Epoch: 18/20 Loss: 0.00030301022343337536
Accuracy TRAIN: 99.6295152824946%
Accuracy EVAL: 81.9614711033275%
Epoch: 19/20 Loss: 0.025745956227183342
Epoch: 19/20 Loss: 0.020785998553037643
Epoch: 19/20 Loss: 0.04456275328993797
Epoch: 19/20 Loss: 0.03337020054459572
Accuracy TRAIN: 98.76505094164865%
Accuracy EVAL: 77.23292469352015%
Epoch: 20/20 Loss: 0.13117393851280212
Epoch: 20/20 Loss: 0.01555025763809681
Epoch: 20/20 Loss: 0.020157115533947945
Epoch: 20/20 Loss: 0.004772854037582874
Accuracy TRAIN: 99.16640938561284%
Accuracy EVAL: 82.31173380035027%
Accuracy TEST: 71.9869706840391%
```

**Figura 72.** Proceso de entrenamiento del modelo nº13.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=vitb16_net1kv1_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.6016949152542372      PRECISION: 0.6454545454545455
```

**Figura 73.** Recall y Precision del modelo nº13.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.38195639848709106
Epoch: 17/20 Loss: 0.40367603302001953
Epoch: 17/20 Loss: 0.47299855947494507
Epoch: 17/20 Loss: 0.3416247069835663
Accuracy TRAIN: 78.97499228156839%
Accuracy EVAL: 78.45884413309983%
Epoch: 18/20 Loss: 0.4761893153190613
Epoch: 18/20 Loss: 0.39824771881103516
Epoch: 18/20 Loss: 0.4668539762496948
Epoch: 18/20 Loss: 0.47438275814056396
Accuracy TRAIN: 79.71596171657919%
Accuracy EVAL: 77.5831873905429%
Epoch: 19/20 Loss: 0.4072750508785248
Epoch: 19/20 Loss: 0.46924304962158203
Epoch: 19/20 Loss: 0.5574346780776978
Epoch: 19/20 Loss: 0.4324144124984741
Accuracy TRAIN: 79.68508799012041%
Accuracy EVAL: 78.45884413309983%
Epoch: 20/20 Loss: 0.381963849067688
Epoch: 20/20 Loss: 0.3534873127937317
Epoch: 20/20 Loss: 0.4090680480003357
Epoch: 20/20 Loss: 0.31531375646591187
Accuracy TRAIN: 79.59246681074406%
Accuracy EVAL: 78.80910683012259%
Accuracy TEST: 68.40390879478828%
```

**Figura 74.** Proceso de entrenamiento del modelo n°14.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=vitb16_net1kv1_t1_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.4406779661016949      PRECISION: 0.6265060240963856
```

**Figura 75.** Recall y Precision del modelo n°14.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.012691482901573181
Epoch: 17/20 Loss: 0.026975855231285095
Epoch: 17/20 Loss: 0.05777008831501007
Epoch: 17/20 Loss: 0.0382189005613327
Accuracy TRAIN: 98.20932386539056%
Accuracy EVAL: 76.00700525394045%
Epoch: 18/20 Loss: 0.054992832243442535
Epoch: 18/20 Loss: 0.029735935851931572
Epoch: 18/20 Loss: 0.11086469888687134
Epoch: 18/20 Loss: 0.10173512250185013
Accuracy TRAIN: 97.86971287434393%
Accuracy EVAL: 78.10858143607706%
Epoch: 19/20 Loss: 0.0918850302696228
Epoch: 19/20 Loss: 0.054762184619903564
Epoch: 19/20 Loss: 0.02914651855826378
Epoch: 19/20 Loss: 0.007521785330027342
Accuracy TRAIN: 98.64155603581352%
Accuracy EVAL: 80.38528896672504%
Epoch: 20/20 Loss: 0.02292933501303196
Epoch: 20/20 Loss: 0.002245347946882248
Epoch: 20/20 Loss: 0.04722041264176369
Epoch: 20/20 Loss: 0.02269451878964901
Accuracy TRAIN: 98.27107131830812%
Accuracy EVAL: 81.08581436077058%
Accuracy TEST: 64.82084690553746%
```

**Figura 76.** Proceso de entrenamiento del modelo nº15.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=vitb16_net1k_swag_v1_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')

RECALL: 0.4533898305084746      PRECISION: 0.5515463917525774
```

**Figura 77.** Recall y Precision del modelo nº15.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.464161217212677
Epoch: 17/20 Loss: 0.4310188591480255
Epoch: 17/20 Loss: 0.27112913131713867
Epoch: 17/20 Loss: 0.5140191912651062
Accuracy TRAIN: 78.35751775239271%
Accuracy EVAL: 75.83187390542908%
Epoch: 18/20 Loss: 0.4107484817504883
Epoch: 18/20 Loss: 0.46519404649734497
Epoch: 18/20 Loss: 0.4072154760360718
Epoch: 18/20 Loss: 0.41439422965049744
Accuracy TRAIN: 78.60450756406298%
Accuracy EVAL: 76.5323992994746%
Epoch: 19/20 Loss: 0.40266287326812744
Epoch: 19/20 Loss: 0.4220483899116516
Epoch: 19/20 Loss: 0.4156891703605652
Epoch: 19/20 Loss: 0.41606831550598145
Accuracy TRAIN: 78.45013893176906%
Accuracy EVAL: 75.13134851138354%
Epoch: 20/20 Loss: 0.3581312894821167
Epoch: 20/20 Loss: 0.3793090283870697
Epoch: 20/20 Loss: 0.3939703702926636
Epoch: 20/20 Loss: 0.4181946814060211
Accuracy TRAIN: 78.63538129052176%
Accuracy EVAL: 76.18213660245183%
Accuracy TEST: 67.91530944625407%
```

**Figura 78.** Proceso de entrenamiento del modelo n°16.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=vitb16_net1k_swag_v1_t1_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.4110169491525424      PRECISION: 0.6258064516129033
```

**Figura 79.** Recall y Precision del modelo n°16.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.11066392809152603
Epoch: 17/20 Loss: 0.1520497351884842
Epoch: 17/20 Loss: 0.276885986328125
Epoch: 17/20 Loss: 0.3152613639831543
Accuracy TRAIN: 91.78758876196356%
Accuracy EVAL: 69.52714535901926%
Epoch: 18/20 Loss: 0.4457387328147888
Epoch: 18/20 Loss: 0.15457245707511902
Epoch: 18/20 Loss: 0.1120663732290268
Epoch: 18/20 Loss: 0.08388233184814453
Accuracy TRAIN: 92.6829268292683%
Accuracy EVAL: 70.2276707530648%
Epoch: 19/20 Loss: 0.06783290952444077
Epoch: 19/20 Loss: 0.13288091123104095
Epoch: 19/20 Loss: 0.15645065903663635
Epoch: 19/20 Loss: 0.1395682692527771
Accuracy TRAIN: 94.01049706699598%
Accuracy EVAL: 70.05253940455341%
Epoch: 20/20 Loss: 0.09525355696678162
Epoch: 20/20 Loss: 0.15056952834129333
Epoch: 20/20 Loss: 0.20368246734142303
Epoch: 20/20 Loss: 0.171127051115036
Accuracy TRAIN: 94.35010805804261%
Accuracy EVAL: 69.87740805604203%
Accuracy TEST: 57.817589576547235%
```

**Figura 80.** Proceso de entrenamiento del modelo n°17.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=vitb16_net1k_swag_linear_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')

RECALL: 0.2923728813559322      PRECISION: 0.42857142857142855
```

**Figura 81.** Recall y Precision del modelo n°17.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.4781123399734497
Epoch: 17/20 Loss: 0.3988746404647827
Epoch: 17/20 Loss: 0.5207511186599731
Epoch: 17/20 Loss: 0.4337102174758911
Accuracy TRAIN: 75.23927138005557%
Accuracy EVAL: 82.48686514886164%
Epoch: 18/20 Loss: 0.4622308909893036
Epoch: 18/20 Loss: 0.5357308387756348
Epoch: 18/20 Loss: 0.44084465503692627
Epoch: 18/20 Loss: 0.4622189998626709
Accuracy TRAIN: 75.5788823711022%
Accuracy EVAL: 82.66199649737302%
Epoch: 19/20 Loss: 0.5025667548179626
Epoch: 19/20 Loss: 0.5047532916069031
Epoch: 19/20 Loss: 0.48604559898376465
Epoch: 19/20 Loss: 0.4101216793060303
Accuracy TRAIN: 75.64062982401975%
Accuracy EVAL: 82.48686514886164%
Epoch: 20/20 Loss: 0.559292197227478
Epoch: 20/20 Loss: 0.4169818162918091
Epoch: 20/20 Loss: 0.4791163206100464
Epoch: 20/20 Loss: 0.4391438961029053
Accuracy TRAIN: 75.5480086446434%
Accuracy EVAL: 82.66199649737302%
Accuracy TEST: 64.82084690553746%
```

**Figura 82.** Proceso de entrenamiento del modelo n°18.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=vitb16_net1k_swag_linear_tl_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.3008474576271186      PRECISION: 0.5819672131147541
```

**Figura 83.** Recall y Precision del modelo n°18.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.004923205357044935
Epoch: 17/20 Loss: 0.03410135209560394
Epoch: 17/20 Loss: 0.0375346839427948
Epoch: 17/20 Loss: 0.09197960048913956
Accuracy TRAIN: 99.22815683853041%
Accuracy EVAL: 81.26094570928196%
Epoch: 18/20 Loss: 0.0210318211466074
Epoch: 18/20 Loss: 0.029437826946377754
Epoch: 18/20 Loss: 0.024942565709352493
Epoch: 18/20 Loss: 0.013501969166100025
Accuracy TRAIN: 99.16640938561284%
Accuracy EVAL: 81.43607705779334%
Epoch: 19/20 Loss: 0.01496062520891428
Epoch: 19/20 Loss: 0.0009758005617186427
Epoch: 19/20 Loss: 0.0048784855753183365
Epoch: 19/20 Loss: 8.679293387103826e-05
Accuracy TRAIN: 99.87650509416487%
Accuracy EVAL: 79.33450087565674%
Epoch: 20/20 Loss: 0.01123636681586504
Epoch: 20/20 Loss: 0.0030794916674494743
Epoch: 20/20 Loss: 9.418829722562805e-05
Epoch: 20/20 Loss: 0.00024066197511274368
Accuracy TRAIN: 99.87650509416487%
Accuracy EVAL: 80.03502626970227%
Accuracy TEST: 71.9869706840391%
```

**Figura 84.** Proceso de entrenamiento del modelo nº19.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=cnnvgg16_net1k_v1_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.6059322033898306      PRECISION: 0.6441441441441441
```

**Figura 85.** Recall y Precision del modelo nº19.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.016005145385861397
Epoch: 17/20 Loss: 0.043513223528862
Epoch: 17/20 Loss: 0.004371688235551119
Epoch: 17/20 Loss: 0.00602799654006958
Accuracy TRAIN: 99.44427292374189%
Accuracy EVAL: 80.03502626970227%
Epoch: 18/20 Loss: 0.0035689601209014654
Epoch: 18/20 Loss: 0.01128181628882885
Epoch: 18/20 Loss: 0.001788821304216981
Epoch: 18/20 Loss: 0.012780282646417618
Accuracy TRAIN: 99.6295152824946%
Accuracy EVAL: 81.08581436077058%
Epoch: 19/20 Loss: 0.062172334641218185
Epoch: 19/20 Loss: 0.002858190331608057
Epoch: 19/20 Loss: 0.0020239679142832756
Epoch: 19/20 Loss: 0.01529325544834137
Accuracy TRAIN: 99.6295152824946%
Accuracy EVAL: 79.8598949211909%
Epoch: 20/20 Loss: 0.0004984557162970304
Epoch: 20/20 Loss: 0.004373940173536539
Epoch: 20/20 Loss: 0.0105860885232687
Epoch: 20/20 Loss: 0.02890620566904545
Accuracy TRAIN: 99.56776782957704%
Accuracy EVAL: 80.38528896672504%
Accuracy TEST: 65.47231270358306%
```

**Figura 86.** Proceso de entrenamiento del modelo n°20.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=cnnvgg16_net1k_v1_t1_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.3983050847457627      PRECISION: 0.573170731707317
```

**Figura 87.** Recall y Precision del modelo n°20.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.006452740170061588
Epoch: 17/20 Loss: 0.03724350407719612
Epoch: 17/20 Loss: 0.024478401988744736
Epoch: 17/20 Loss: 0.006969411391764879
Accuracy TRAIN: 99.38252547082433%
Accuracy EVAL: 80.56042031523643%
Epoch: 18/20 Loss: 0.0019181323004886508
Epoch: 18/20 Loss: 0.005137878470122814
Epoch: 18/20 Loss: 0.0029807542450726032
Epoch: 18/20 Loss: 0.0031090311240404844
Accuracy TRAIN: 99.25903056498919%
Accuracy EVAL: 83.0122591943958%
Epoch: 19/20 Loss: 0.0033956337720155716
Epoch: 19/20 Loss: 0.004397546406835318
Epoch: 19/20 Loss: 0.0007181295659393072
Epoch: 19/20 Loss: 0.0022213836200535297
Accuracy TRAIN: 99.69126273541217%
Accuracy EVAL: 83.71278458844134%
Epoch: 20/20 Loss: 0.0006394522497430444
Epoch: 20/20 Loss: 0.001545166946016252
Epoch: 20/20 Loss: 0.0004280325083527714
Epoch: 20/20 Loss: 0.0010608811862766743
Accuracy TRAIN: 99.44427292374189%
Accuracy EVAL: 82.31173380035027%
Accuracy TEST: 69.05537459283387%
```

**Figura 88.** Proceso de entrenamiento del modelo n°21.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=resnet50_net1k_v1_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.4957627118644068      PRECISION: 0.6223404255319149
```

**Figura 89.** Recall y Precision del modelo n°21.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.47348710894584656
Epoch: 17/20 Loss: 0.4605488181114197
Epoch: 17/20 Loss: 0.5172529220581055
Epoch: 17/20 Loss: 0.47495096921920776
Accuracy TRAIN: 75.5788823711022%
Accuracy EVAL: 74.08056042031524%
Epoch: 18/20 Loss: 0.4677823781967163
Epoch: 18/20 Loss: 0.4908680021762848
Epoch: 18/20 Loss: 0.5322849154472351
Epoch: 18/20 Loss: 0.4759504795074463
Accuracy TRAIN: 75.85674590923125%
Accuracy EVAL: 74.78108581436076%
Epoch: 19/20 Loss: 0.41299867630004883
Epoch: 19/20 Loss: 0.4438205361366272
Epoch: 19/20 Loss: 0.47680234909057617
Epoch: 19/20 Loss: 0.4401399791240692
Accuracy TRAIN: 76.35072553257179%
Accuracy EVAL: 75.13134851138354%
Epoch: 20/20 Loss: 0.38278910517692566
Epoch: 20/20 Loss: 0.4603692293167114
Epoch: 20/20 Loss: 0.4988930821418762
Epoch: 20/20 Loss: 0.4640847146511078
Accuracy TRAIN: 76.19635690027786%
Accuracy EVAL: 75.13134851138354%
Accuracy TEST: 63.51791530944625%
```

**Figura 90.** Proceso de entrenamiento del modelo n°22.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=resnet50_net1k_v1_t1_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.3516949152542373      PRECISION: 0.538961038961039
```

**Figura 91.** Recall y Precision del modelo n°22.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.002161073964089155
Epoch: 17/20 Loss: 0.001162888715043664
Epoch: 17/20 Loss: 0.0008490591426379979
Epoch: 17/20 Loss: 0.0005417070351541042
Accuracy TRAIN: 99.87650509416487%
Accuracy EVAL: 85.11383537653239%
Epoch: 18/20 Loss: 0.06000712513923645
Epoch: 18/20 Loss: 0.0005283989594317973
Epoch: 18/20 Loss: 0.008304095827043056
Epoch: 18/20 Loss: 0.0012332570040598512
Accuracy TRAIN: 99.96912627354122%
Accuracy EVAL: 83.71278458844134%
Epoch: 19/20 Loss: 0.0005131851648911834
Epoch: 19/20 Loss: 0.0008879387169145048
Epoch: 19/20 Loss: 0.00372330448590219
Epoch: 19/20 Loss: 0.0007746859919279814
Accuracy TRAIN: 99.96912627354122%
Accuracy EVAL: 85.11383537653239%
Epoch: 20/20 Loss: 0.0007542960811406374
Epoch: 20/20 Loss: 0.0007264170562848449
Epoch: 20/20 Loss: 0.0019065765663981438
Epoch: 20/20 Loss: 0.0019423146732151508
Accuracy TRAIN: 99.90737882062365%
Accuracy EVAL: 82.83712784588441%
Accuracy TEST: 64.65798045602605%
```

**Figura 92.** Proceso de entrenamiento del modelo n°23.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=resnet50_net1k_v2_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.6186440677966102      PRECISION: 0.5347985347985348
```

**Figura 93.** Recall y Precision del modelo n°23.

Fuente: Elaboración propia.

```
Epoch: 17/20 Loss: 0.37703657150268555
Epoch: 17/20 Loss: 0.5201420187950134
Epoch: 17/20 Loss: 0.4633752107620239
Epoch: 17/20 Loss: 0.5867621898651123
Accuracy TRAIN: 75.7641247298549%
Accuracy EVAL: 77.23292469352015%
Epoch: 18/20 Loss: 0.5297441482543945
Epoch: 18/20 Loss: 0.5357515811920166
Epoch: 18/20 Loss: 0.5242770910263062
Epoch: 18/20 Loss: 0.5246537327766418
Accuracy TRAIN: 75.11577647422044%
Accuracy EVAL: 76.88266199649738%
Epoch: 19/20 Loss: 0.540317177772522
Epoch: 19/20 Loss: 0.5896735787391663
Epoch: 19/20 Loss: 0.481535941362381
Epoch: 19/20 Loss: 0.4626829922199249
Accuracy TRAIN: 75.42451373880827%
Accuracy EVAL: 76.35726795096322%
Epoch: 20/20 Loss: 0.42382216453552246
Epoch: 20/20 Loss: 0.40928202867507935
Epoch: 20/20 Loss: 0.3834560513496399
Epoch: 20/20 Loss: 0.5876160860061646
Accuracy TRAIN: 75.91849336214881%
Accuracy EVAL: 77.23292469352015%
Accuracy TEST: 63.02931596091205%
```

**Figura 94.** Proceso de entrenamiento del modelo n°24.

Fuente: Elaboración propia.

```
r, p = get_recall_and_precision(model=resnet50_net1k_v2_t1_GEN, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')
RECALL: 0.3220338983050847      PRECISION: 0.5314685314685315
```

**Figura 95.** Recall y Precision del modelo n°24.

Fuente: Elaboración propia.

### Actividad 3: Elaborar un modelo híbrido entre ViT y CNN

Luego del entrenamiento y un análisis rápido de resultados de los modelos CNN y ViT, se determinó que la arquitectura de mejor desempeño fue ResNet 50 con los pesos de IMAGENET1KV1 entrenado con la técnica de Fine Tuning y sin la data aumentada, pues su métrica Accuracy superó a los demás modelos.

Se decidió tomar este modelo CNN para la construcción de un modelo híbrido con ViT. Para lograr esto, se usó la librería “transformers” de Python que se encuentra actualmente mantenida por Hugging Face. Esta librería permitió personalizar un modelo ViT base a diferencia de las incluidas en Pytorch, donde los modelos similares venían de forma predefinida y dificultaban su personalización de acuerdo a las necesidades de esta investigación.

Es así que, para lograr una correcta integración del modelo ResNet 50, fue necesario configurar el ViT con un nuevo tamaño de patch, número de canales de entrada y el tamaño de la imagen de entrada que, al ser en este caso los vectores característicos extraídos por la arquitectura CNN, tuvieron la forma de 1 x 2048. En el caso número de canales, este se estableció en 1 debido nuevamente a la forma del vector característico. El tamaño de patch se estableció en 1 de igual forma a cómo se define el paper de JERBI et al. (2023), fuente original y de inspiración para esta investigación.

#### **Actividad 4: Entrenar y probar modelo híbrido CNN + ViT**

Ya con el modelo híbrido construido se pasó a la etapa de entrenamiento.

Debido a que el modelo CNN ResNet 50 ganador se entrenó con los pesos de IMAGENET1KV1 con Fine Tuning y sin la data aumentada, se usó esta misma configuración para el modelo híbrido. De igual forma, la función de pérdida y de optimización se mantuvieron como los anteriores modelos.

Con todas estas especificaciones, se inició el entrenamiento del modelo híbrido con 20 épocas. A continuación, en la Figura 96 presenta la parte final del entrenamiento del modelo.

```
Epoch: 19/20 Loss: 0.2771833539009094
Epoch: 19/20 Loss: 0.13229389488697052
Epoch: 19/20 Loss: 0.30232831835746765
Epoch: 19/20 Loss: 0.07152456045150757
Epoch: 19/20 Loss: 0.08891774713993073
Epoch: 19/20 Loss: 0.08088434487581253
Epoch: 19/20 Loss: 0.18291223049163818
Epoch: 19/20 Loss: 0.0143346656113863
Epoch: 19/20 Loss: 0.25516265630722046
Accuracy TRAIN: 93.75%
Accuracy EVAL: 74.70997679814386%
Epoch: 20/20 Loss: 0.016351595520973206
Epoch: 20/20 Loss: 0.013921474106609821
Epoch: 20/20 Loss: 0.07057715952396393
Epoch: 20/20 Loss: 0.1793939173221588
Epoch: 20/20 Loss: 0.06951679289340973
Epoch: 20/20 Loss: 0.3222339451313019
Epoch: 20/20 Loss: 0.024347994476556778
Epoch: 20/20 Loss: 0.04480632767081261
Epoch: 20/20 Loss: 0.013779292814433575
Epoch: 20/20 Loss: 0.029389213770627975
Epoch: 20/20 Loss: 0.008809763006865978
Epoch: 20/20 Loss: 0.016349663957953453
Accuracy TRAIN: 96.07843137254902%
Accuracy EVAL: 72.15777262180974%
Accuracy TEST: 67.91530944625407%
```

**Figura 96.** Proceso de entrenamiento del modelo híbrido.

Fuente: Elaboración propia.

Además, también se presenta las métricas de Recall y Precision en la Figura 97.

```
r, p = get_recall_and_precision(model=vithybrid, test_dataloader=test_data, device=device)
print(f'RECALL: {r} \tPRECISION: {p}')

RECALL: 0.788135593220339      PRECISION: 0.5585585585585585
```

**Figura 97.** Recall y Precision del modelo híbrido.

Fuente: Elaboración propia.

## 4.3 Medición de la solución

### 4.3.1 Análisis de Indicadores cuantitativo

Los indicadores a analizar en esta sección son los obtenidos en el entrenamiento de todos los modelos CNN y ViT, incluyendo además el modelo híbrido. Cada uno de estos dio como resultado principal tres indicadores: Accuracy, Recall y Precision. El primero de estos permitió determinar la capacidad de predicción para ambas clases. El segundo y tercer indicador ayudan a evaluar el modelo en cuestión de capacidad de predicción para la clase positiva. En el caso de esta investigación, la clase positiva es maligno (1).

En la Tabla 30 se presentan los resultados de los modelos.

Tabla 30

*Resultados de los modelos entrenados con los dos conjuntos de datos, técnicas de entrenamiento y pesos.*

Aumento de datos con DCGAN	Modelo	Pesos	Tipo de entrena- miento	Accuracy	Recall	Precision
No	ViT B 16	I-NET1K V1	FT	62.38 %	77.12 %	50.70 %
			TL	66.78 %	41.10 %	59.88 %
		I-NET1K SE V1	FT	60.26 %	35.17 %	47.70 %
			TL	68.40 %	41.10 %	63.82 %
	VGG16	I-NET1K SL V1	FT	54.89 %	67.37 %	44.29 %
			TL	63.84 %	25.85 %	56.48 %
		I-NET1K V1	FT	69.87 %	72.46 %	58.76 %
			TL	66.29 %	55.93 %	56.17 %
	RESNET50	I-NET1K V1	FT	72.48 %	64.41 %	64.14 %
			TL	63.52 %	19.49 %	57.50 %
		I-NET1K V2	FT	69.54 %	52.12 %	62.44 %
			TL	63.03 %	20.76 %	55.06 %
Sí	ViT B 16	I-NET1K V1	FT	71.99 %	60.17 %	64.55 %
			TL	68.40 %	44.07 %	62.65 %
		I-NET1K SE V1	FT	64.82 %	45.34 %	55.15 %
			TL	67.92 %	41.10 %	62.58 %
	VGG16	I-NET1K SL V1	FT	57.82 %	29.24 %	42.86 %
			TL	64.82 %	30.08 %	58.20 %
		I-NET1K V1	FT	71.99 %	60.59 %	64.41 %
			TL	65.47 %	39.83 %	57.32 %
	RESNET50	I-NET1K V1	FT	69.06 %	49.58 %	62.23 %
			TL	63.52 %	35.17 %	53.90 %
		I-NET1K V2	FT	64.66 %	61.86 %	53.48 %
			TL	63.03 %	32.20 %	53.15 %

Fuente: Elaboración propia.

En relación al indicador Accuracy, el modelo de mejor desempeño fue el ResNet 50 con

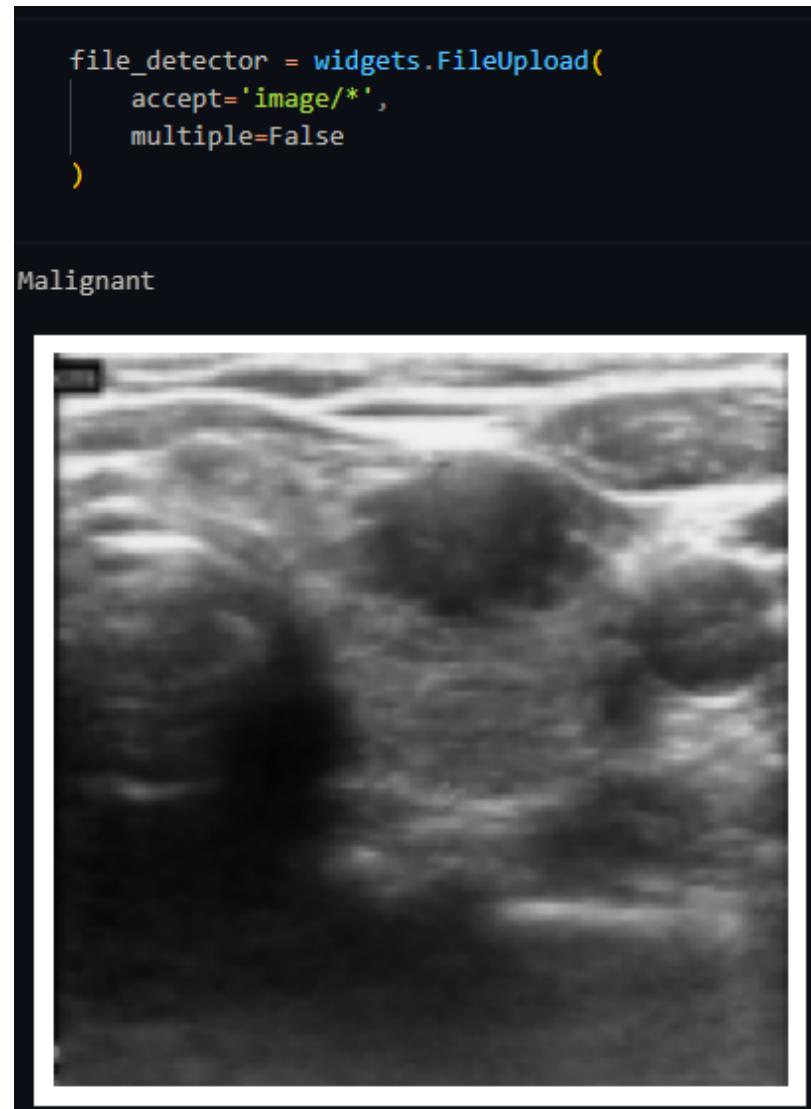
los pesos IMAGENET1KV1 y la técnica de entrenamiento Fine Tuning, junto con el uso del conjunto de datos original, alcanzado un valor de 72.48 %. En este mismo modelo, el Recall y Precision, aunque no destacables, pueden considerarse como relativamente altos si se compara con los demás modelos.

Algo que destacar sobre los demás modelos es el Recall del ViT CNN híbrido, pues superó en gran medida a los demás con un valor de 78.81 %, demostrando a primera vista una relativamente alta capacidad de predicción sobre todos los reales positivos; sin embargo, debido a la poca cantidad de datos de clase positiva en el conjunto de datos original, era lógico pensar que este valor podría llegar a ser alto en los modelos, por lo que este no puede ser un indicador decisivo en la toma de decisiones final. Por este motivo, se decidió elegir el modelo ResNet 50 con los pesos IMAGENET1KV1 y la técnica de entrenamiento Fine Tuning como el modelo para la simulación.

### 4.3.2 Simulación de solución

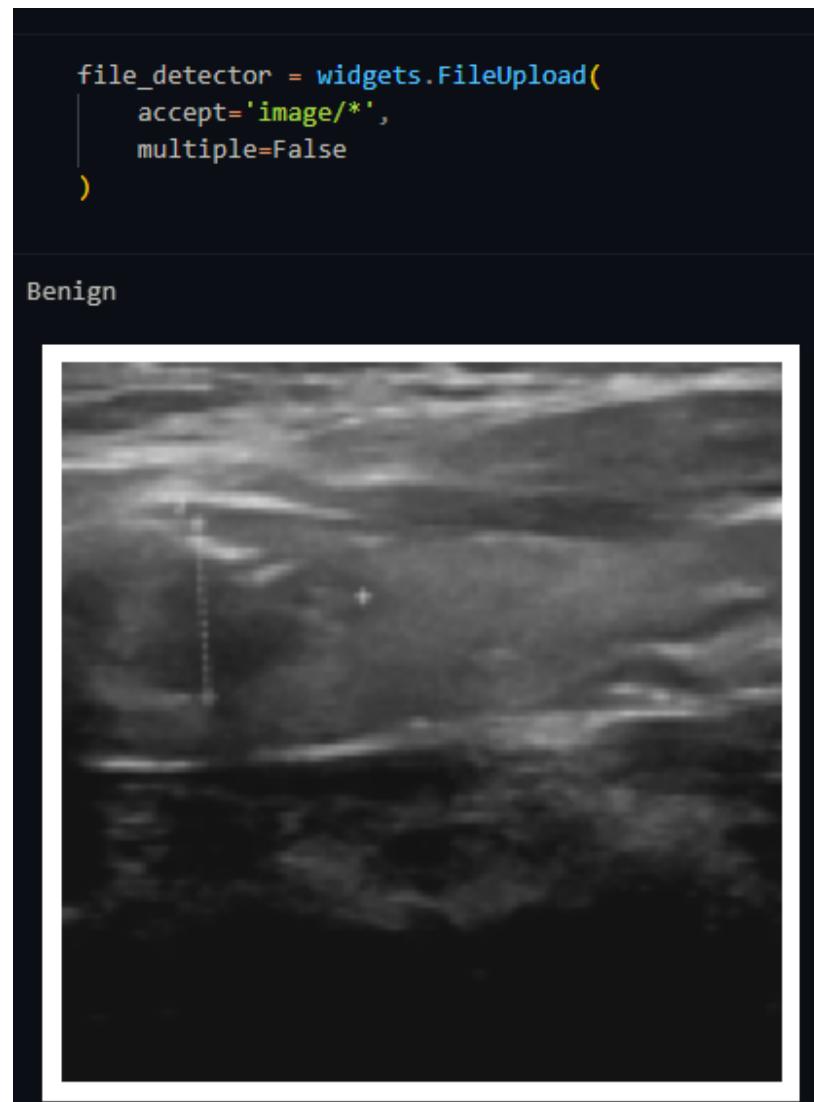
Luego del análisis de los indicadores de los modelos, y de la elección final, se decidió realizar una simulación de uso del modelo con imágenes del conjunto de datos de prueba.

Para lograr esto, se usó la librería ipywidgets de python que permitió, a través de un mismo jupyter notebook, subir una imagen e ingresarla al modelo ya cargado y obtener finalmente el resultado de su predicción. En la Figura 98 y Figura 99 se muestra la prueba con dos imágenes de ultrasonido de distintas clases.



**Figura 98.** Prueba de modelo con imagen de ultrasonido de nódulo maligno.

Fuente: Elaboración propia.



**Figura 99.** Prueba de modelo con imagen de ultrasonido de nódulo benigno.

Fuente: Elaboración propia.

## Referencias

- Ayana, G., Dese, K., Dereje, Y., Kebede, Y., Barki, H., Amdissa, D., Husen, N., Mulugeta, F., Habtamu, B., & Choe, S.-W. (2023). Vision-Transformer-Based Transfer Learning for Mammogram Classification. *Diagnostics*, 13(2). doi: 10.3390/diagnostics13020178.
- Bhattacharya, S., Reddy Maddikunta, P. K., Pham, Q. V., Gadekallu, T. R., Krishnan S, S. R., Chowdhary, C. L., Alazab, M., & Jalil Piran, M. (2021). Deep learning and medical image processing for coronavirus (COVID-19) pandemic: A survey. *Sustainable Cities and Society*, 65. doi: 10.1016/j.scs.2020.102589.
- Binboga, S., Gemici, E., & Binboga, E. (2019). Thyroid Anatomy. En *Knowledges on Thyroid Cancer* (pp. 1-11). IntechOpen.
- Camacho, J., Svilainis, L., & Gomez Alvarez-Arenas, T. (2022). Ultrasonic Imaging and Sensors. *Sensors*, 22(20), 7911-7911. doi: 10.3390/s22207911v.
- Deng, P., Han, X., Wei, X., & Chang, L. (2022). Automatic classification of thyroid nodules in ultrasound images using a multi-task attention network guided by clinical knowledge. *Computers in Biology and Medicine*, 150. doi: 10.1016/j.combiomed.2022.106172.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE. *ICLR 2021 - 9th International Conference on Learning Representations*.
- Felgueiras Carvalho, A. R. (2019). *3D Lung Nodule Classification in Computed Tomography Images* [Tesis de maestría]. Universidad de Oporto. Recuperado de <https://repositorio-aberto.up.pt/handle/10216/123391>.
- Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2<sup>a</sup> ed.). O'Reilly Media.
- Haugen, B. R., Alexander, E. K., Bible, K. C., Doherty, G. M., Mandel, S. J., Nikiforov, Y. E., Pacini, F., Randolph, G. W., Sawka, A. M., Schlumberger, M., Schuff, K. G., Sherman, S. I., Sosa, J. A., Steward, D. L., Tuttle, R. M., & Wartofsky, L. (2016). 2015 American Thyroid Association Management Guidelines for Adult Patients with Thyroid Nodules and Differentiated Thyroid Cancer: The American Thyroid Association Guidelines Task Force on Thyroid Nodules and Differentiated Thyroid Cancer. *Thyroid*, 26(1), 1-133. doi: 10.1089/thy.2015.0020.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. doi: 10.1109/cvpr.2016.90.

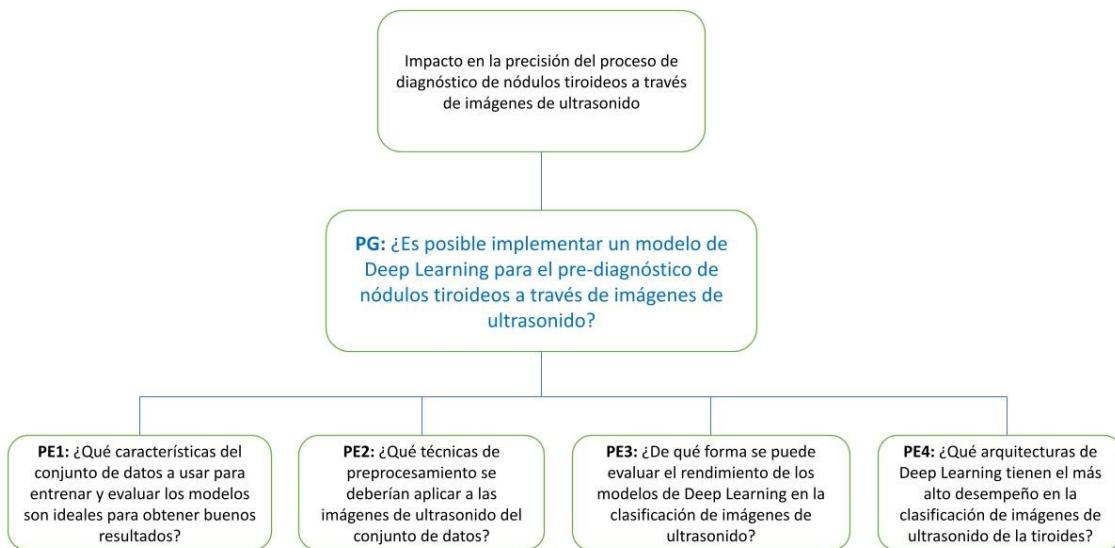
- Herrera Gajardo, J. I. (2017). *Diseño e implementación de técnicas de procesamiento de imágenes para dispositivo de ultrasonido portátil* [Tesis de pregrado]. Universidad de Chile. Recuperado de <https://repositorio.uchile.cl/handle/2250/148434>.
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2261-2269. doi: 10.1109/cvpr.2017.243.
- Hurbans, R. (2020). *Grokking Artificial Intelligence Algorithms*. Manning Publications.
- Inkawich, N. (2024). DCGAN Tutorial. Recuperado de [https://pytorch.org/tutorials/beginner/dcgan\\_faces\\_tutorial.html](https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html).
- Izco, F. (2018). Base de Datos Corporativa de Personas. Recuperado de [https://bookdown.org/f\\_izco/BDC-POC/metricas.html](https://bookdown.org/f_izco/BDC-POC/metricas.html).
- JERBI, F., ABOUDI, N., & KHLIFA, N. (2023). Automatic classification of ultrasound thyroids images using vision transformers and generative adversarial networks. *Scientific African*, 20. doi: 10.1016/j.sciaf.2023.e01679.
- Kang, Q., Lao, Q., Li, Y., Jiang, Z., Qiu, Y., Zhang, S., & Li, K. (2022). Thyroid nodule segmentation and classification in ultrasound images through intra- and inter-task consistent learning. *Medical Image Analysis*, 79. doi: 10.1016/j.media.2022.102443.
- Kim, J., Gosnell, J. E., & Roman, S. A. (2020). Geographic influences in the global rise of thyroid cancer. *Nature Reviews Endocrinology*, 16(1), 17-29. doi: 10.1038/s41574-019-0263-x.
- Manzari, O. N., Ahmadabadi, H., Kashiani, H., Shokouhi, S. B., & Ayatollahi, A. (2023). MedViT: A robust vision transformer for generalized medical image classification. *Computers in Biology and Medicine*, 157. doi: 10.1016/j.combiomed.2023.106791.
- Monroy Malca, V. d. P. (2021). *Diseño de un sistema de visión computacional para el pre-diagnóstico de la enfermedad de Parkinson a partir de la escritura de una persona* [Tesis de grado]. Universidad ESAN. Recuperado de <https://hdl.handle.net/20.500.12640/2262>.
- Moreira Aresta, G. (2021). *Detection of lung nodules in computed tomography images* [Tesis de grado]. Universidad de Oporto. Recuperado de <https://repositorio-aberto.up.pt/bitstream/10216/134135/2/473080.pdf>.
- Moroney, L. (2020). *AI and Machine Learning for Coders*. O'Reilly Media. Recuperado de <https://books.google.com.pe/books?id=gw4CEAAAQBAJ>.
- Organización Mundial de la Salud. (2022). Cancer Today. Recuperado de [https://gco.iarc.who.int/today/en/dataviz/maps-heatmap?mode=population&key=crude\\_rate&types=0&age\\_start=0&cancers=32&sexes=2](https://gco.iarc.who.int/today/en/dataviz/maps-heatmap?mode=population&key=crude_rate&types=0&age_start=0&cancers=32&sexes=2).
- Regmi, S., Subedi, A., Bagci, U., & Jha, D. (2023). Vision transformer for efficient chest X-ray and gastrointestinal image classification. doi: 10.48550/arxiv.2304.11529.

- Sampieri, R. H., Collado, C. F., & Lucio, P. B. (2014). *Metodología de la investigación* (6<sup>a</sup> ed.). McGraw-Hill Education. Recuperado de <https://books.google.com.pe/books?id=oLbjQEACAAJ>.
- Shin, J. H., Baek, J. H., Chung, J., Ha, E. J., Kim, J.-H., Lee, Y. H., Lim, H. K., Moon, W.-J., Na, D. G., Park, J. S., Choi, Y. J., Hahn, S. Y., Jeon, S. J., Jung, S. L., Kim, D. W., Kim, E.-K., Kwak, J. Y., Lee, C. Y., Lee, H. J., ... Sung, J. Y. (2016). Ultrasonography Diagnosis and Imaging-Based Management of Thyroid Nodules: Revised Korean Society of Thyroid Radiology Consensus Statement and Recommendations. *Korean Journal of Radiology*, 17(3), 370-395. doi: 10.3348/kjr.2016.17.3.370.
- Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. Recuperado de <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083953063&partnerID=40&md5=a5b2d6b3fc9f0a6f92864467079a1280>.
- Singh, S. P., Wang, L., Gupta, S., Goli, H., Padmanabhan, P., & Gulyás, B. (2020). 3d deep learning on medical images: A review. *Sensors (Switzerland)*, 20(18), 1-24. doi: 10.3390/s20185097.
- Sun, J., Li, C., Lu, Z., He, M., Zhao, T., Li, X., Gao, L., Xie, K., Lin, T., Sui, J., Xi, Q., Zhang, F., & Ni, X. (2022). TNSNet: Thyroid nodule segmentation in ultrasound imaging using soft shape supervision. *Computer Methods and Programs in Biomedicine*, 215. doi: 10.1016/j.cmpb.2021.106600.
- Sun, J., Wu, B., Zhao, T., Gao, L., Xie, K., Lin, T., Sui, J., Li, X., Wu, X., & Ni, X. (2023). Classification for thyroid nodule using ViT with contrastive learning in ultrasound images. *Computers in Biology and Medicine*, 152. doi: 10.1016/j.compbiomed.2022.106444.
- Supanta Zapata, C. A. (2021). *Desarrollo de un algoritmo orientado a la detección y extracción de características de nódulos pulmonares en imágenes radiográficas digitales* [Tesis de pregrado]. Universidad de San Martín de Porres. Recuperado de <https://hdl.handle.net/20.500.12727/7556>.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1-9. doi: 10.1109/cvpr.2015.7298594.
- Tampu, I. E., Eklund, A., Johansson, K., Gimm, O., & Haj-Hosseini, N. (2023). Diseased thyroid tissue classification in OCT images using deep learning: Towards surgical decision support. *Journal of Biophotonics*, 16(2). doi: 10.1002/jbio.202200227.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need, 5999-6009.

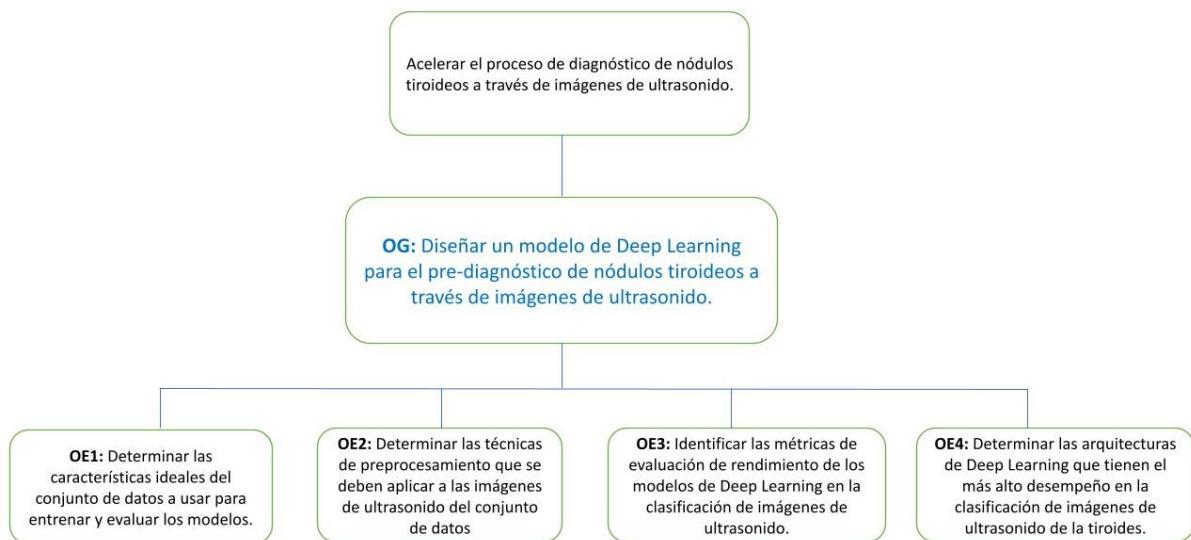
- Wang, L., Zhang, L., Zhu, M., Qi, X., & Yi, Z. (2020). Automatic diagnosis for thyroid nodules in ultrasound images by deep neural networks. *Medical Image Analysis*, 61. doi: 10.1016/j.media.2020.101665.
- Yang, G., Luo, S., & Greer, P. (2023). A Novel Vision Transformer Model for Skin Cancer Classification. *Neural Processing Letters*, 55(7), 9335-9351. doi: 10.1007/s11063-023-11204-5.
- Zhang, J., Mazurowski, M. A., Allen, B. C., & Wildman-Tobriner, B. (2023). Multistep Automated Data Labelling Procedure (MADLaP) for thyroid nodules on ultrasound: An artificial intelligence approach for automating image annotation. *Artificial Intelligence in Medicine*, 141. doi: 10.1016/j.artmed.2023.102553.
- Zhu, Y.-C., AlZoubi, A., Jassim, S., Jiang, Q., Zhang, Y., Wang, Y.-B., Ye, X.-D., & DU, H. (2021). A generic deep learning framework to classify thyroid and breast lesions in ultrasound images. *Ultrasonics*, 110. doi: 10.1016/j.ultras.2020.106300.

## **Anexos**

## A Árbol de Problemas



## B Árbol de Objetivos



## C Matriz de Consistencia

Título de la tesis	Diseño de un modelo de Deep Learning para el pre-diagnóstico de nódulos tiroideos a través de imágenes de ultrasonido			
Problema General	Objetivo General	Hipótesis General	Variables	Método
¿Es posible implementar un modelo de Deep Learning para el pre-diagnóstico de nódulos tiroideos a través de imágenes de ultrasonido?	Diseñar un modelo de Deep Learning para el pre-diagnóstico de nódulos tiroideos a través de imágenes de ultrasonido.	A través del diseño de un modelo de Deep Learning se pre-diagnóstica los nódulos tiroideos a través de imágenes de ultrasonido.	Dependiente: Pre-diagnóstico de nódulos tiroideos. Independiente: Variables dependientes: Modelo de Deep Learning.	
Problemas Específicos	Objetivos Específicos	Hipótesis Específicas	Variables	

¿Qué características del conjunto de datos a usar para entrenar y evaluar los modelos son ideales para obtener buenos resultados?	Determinar las características ideales del conjunto de datos a usar para entrenar y evaluar los modelos.	Las mejores características del conjunto de datos usadas para el proceso de entrenamiento y prueba permiten un mejor desarrollo del modelo de Deep Learning.	Dependiente: Desarrollo del modelo de Deep Learning. Independiente: Las características del conjunto de datos.	Tipo de Investigación: Experimental. Alcance de la investigación: Explicativo.
¿Qué técnicas de preprocessamiento se deberían aplicar a las imágenes de ultrasonido del conjunto de datos?	Determinar las técnicas de preprocessamiento que se deben aplicar a las imágenes de ultrasonido del conjunto de datos.	Las técnicas de preprocessamiento a aplicar a las imágenes de ultrasonido del conjunto de datos mejora el desempeño del modelo de Deep Learning.	Dependiente: Desempeño del modelo de Deep Learning. Independiente: Técnicas de preprocessamiento.	

¿De qué forma se puede evaluar el rendimiento de los modelos de Deep Learning en la clasificación de imágenes de ultrasonido?

Identificar las métricas de evaluación de rendimiento de los modelos de Deep Learning en la clasificación de imágenes de ultrasonido.

Las métricas de evaluación de rendimiento de los modelos de Deep Learning logra una correcta comparación de modelos en la tarea de clasificación de imágenes de ultrasonido.

Dependiente: Comparación de modelos en la tarea de clasificación de imágenes de ultrasonido.

---

¿Qué arquitecturas de Deep Learning tienen el más alto desempeño en la clasificación de imágenes de ultrasonido de la tiroides?

Determinar las arquitecturas de Deep Learning que tienen el más alto desempeño en la clasificación de imágenes de ultrasonido de la tiroides.

Las arquitecturas de Deep Learning con más alto desempeño en la clasificación de imágenes de ultrasonido demuestran superioridad frente a los demás modelos.

Dependiente: Desempeño en la clasificación de imágenes de ultrasonido. Independiente: Las arquitecturas de Deep Learning.

---