UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Comparación y Evaluación de Modelos Transformer en la Clasificación de Imágenes Médicas del Dataset NIH Chest X-rays

Trabajo Fin de Máster

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

AUTOR/A: Huallpa Vargas, Yuri Vladimir

Tutor/a: Paredes Palacios, Roberto

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DPTO. DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

“ Comparación y Evaluación de Modelos Transformer en la Clasificación de Imágenes Médicas del Dataset NIH Chest X-rays”

TRABAJO FIN DE MÁSTER

Autor/a:
Huallpa Vargas, Yuri Vladimir

Tutor:
Paredes Palacios, Roberto

CURSO ACADÉMICO, 2023

Agradecimientos

Deseo expresar mi más sincero agradecimiento a todas las personas que he conocido a lo largo de este Máster, sin su ayuda no hubiera sido fácil de afrontarlo, mas aun estando fuera de mi país natal.

En primer lugar, quiero agradecer a mis padres, Filomena y Damian, por su apoyo incondicional y constante. A mis hermanos, quienes me han motivado a seguir adelante y a convertirme en una mejor persona, brindándome su apoyo emocional en todo momento.

En segundo lugar, un agradecimiento especial a mi tutor, Roberto Pardes Palacios, por brindarme la oportunidad de realizar esta investigación bajo su tutoría y por proporcionarme un tema excepcional que me ha servido bastante en mi desarrollo profesional.

También deseo expresar mi gratitud a Roberto y al Pattern Recognition and Human Language Technology Center (*PRHLT*), por facilitarme el poder de computo, sin esas increíbles maquinas nada de esto hubiera sido posible.

Por último, quiero agradecer al Programa Nacional de Becas y Crédito Educativo (*PRONABEC*, Perú) por su confianza y por hacer posible que pudiera realizar este posgrado. También, quiero agradecer a la Universidad Politécnica de Valencia (*UPV*) y al Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital (*MIARFID*) por brindarme esta valiosa oportunidad de aprendizaje y por las amistades enriquecedoras que he hecho en el camino.

- Yuri Vladimir Huallpa Vargas

Resumen

Este trabajo de fin de máster se centra en una exhaustiva evaluación y comparación de tres modelos Transformer: *ViT*, *Swin* y *MaxViT*, que fueron preentrenados en *ImageNet* y adaptados al conjunto de datos médicos *NIH Chest X-rays*. El objetivo principal es analizar en profundidad el rendimiento de estas arquitecturas en la clasificación de 14 patologías en radiografías de tórax. Se busca una comprensión más detallada explorando métricas clave como el área bajo la curva ROC (*AUC*), la velocidad de inferencia (*Throughput*), la cantidad de parámetros y el numero de operaciones aritméticas de punto flotante (*FLOPs*).

Para lograrlo, se establecen objetivos específicos que incluyen una revisión exhaustiva del estado del arte en la clasificación de imágenes y la adaptación de los modelos preentrenados al conjunto de datos médicos. Los modelos se ajustan en cuatro escalas diferentes y se evalúan para tres resoluciones de imagen distintas. La evaluación se realiza en términos de *AUC* y se compara el rendimiento de cada arquitectura en diversas configuraciones.

Además, se realiza un análisis detallado del rendimiento en función del número de parámetros, *FLOPs* y *throughput*, lo que brinda una comprensión más profunda de las capacidades de cada arquitectura. Este trabajo contribuye al campo de la clasificación de imágenes médicas al proporcionar información valiosa sobre el rendimiento de las arquitecturas Transformer en términos de rendimiento y eficiencia computacional.

Abstract

This master's thesis focuses on a comprehensive evaluation and comparison of three Transformer models: *ViT*, *Swin*, and *MaxViT*, which were pretrained on *ImageNet* and adapted to the medical dataset *NIH Chest X-rays*. The main objective is to deeply analyze the performance of these architectures in the classification of 14 pathologies in chest X-ray images. A more detailed understanding is sought by exploring key metrics such as the Area Under the ROC Curve (*AUC*), inference speed (*throughput*), the number of parameters, and floating-point arithmetic operations (*FLOPs*).

To achieve this, specific objectives are set, including a comprehensive review of the state of the art in image classification and the adaptation of pretrained models to the medical dataset. The models are fine-tuned at four different scales and evaluated for three different image resolutions. The evaluation is conducted in terms of *AUC*, and the performance of each architecture is compared under various configurations.

Furthermore, a detailed analysis of performance in terms of the number of parameters, *FLOPs*, and *throughput* is carried out, providing a deeper understanding of the capabilities of each architecture. This work contributes to the field of medical image classification by providing valuable insights into the performance of Transformer architectures in terms of both performance and computational efficiency.

Resum

Aquest treball de fi de màster es centra en una exhaustiva avaluació i comparació de tres models Transformer: *ViT*, *Swin* i *MaxViT*, que van ser preentrenats en *ImageNet* i adaptats al conjunt de dades mèdiques *NIH Chest X-rays*. L'objectiu principal és analitzar en profunditat el rendiment d'aquestes arquitectures en la classificació de 14 patologies en radiografies de tòrax. Es busca una comprensió més detallada explorant mètriques clau com l'àrea sota la corba ROC (*AUC*), la velocitat d'infència (*throughput*), la quantitat de paràmetres i el nombre d'operacions aritmètiques de punt flotant (*FLOPs*).

Per a aconseguir-ho, s'estableixen objectius específics que inclouen una revisió exhaustiva de l'estat de l'art en la classificació d'imatges i l'adaptació dels models preentrenats al conjunt de dades mèdiques. Els models s'ajusten a quatre escales diferents i s'avaluen per a tres resolucions d'imatge diferents. L'avaluació es realitza en termes de *AUC* i es compara el rendiment de cada arquitectura en diverses configuracions.

A més, es realitza una anàlisi detallada del rendiment en funció del número de paràmetres, *FLOPs* i *throughput*, el que ofereix una comprensió més profunda de les capacitats de cada arquitectura. Aquest treball contribueix al camp de la classificació d'imatges mèdiques en proporcionar informació valuosa sobre el rendiment de les arquitectures Transformer en termes de rendiment i eficiència computacional.

Índice general

Índice de figuras

Índice de Tablas

Parte I.

Introducción

1. Generalidades

1.1. Imágenes médicas

Son aquellas obtenidas mediante un conjunto de técnicas y procesos utilizados para capturar imágenes del cuerpo humano o de partes específicas de él [6] [14]. Estas técnicas se emplean con el propósito de diagnosticar o prevenir patologías, o con fines netamente científicos.

Una de las técnicas más comúnmente utilizadas es la radiografía, que es una técnica de diagnóstico por imagen. Consiste en obtener una imagen del interior del cuerpo humano utilizando radiación electromagnética ionizante, conocida como rayos-X. Estos rayos-X tienen la capacidad de atravesar los diferentes tejidos y órganos del cuerpo, y dependiendo de la densidad de cada estructura, pueden producir distintos tonos de gris. Los tejidos más densos, como los huesos, se representan en tonos más claros, cerca del blanco. Por otro lado, los tejidos menos densos, como los tejidos blandos o el aire en los pulmones, aparecen en tonos más oscuros, cercanos al negro.

Con la radiografía, los médicos pueden visualizar estructuras óseas, tejidos blandos y órganos internos, lo que les permite identificar fracturas, detectar anomalías, evaluar el estado de los órganos y proporcionar diagnósticos médicos de manera mas eficiente y menos invasivo.

1.2. Clasificación de imágenes

La clasificación de imágenes es el proceso de asignar una etiqueta a una imagen de entrada [13] [7] [18]. En este contexto, los modelos matemáticos tienen la capacidad de reconocer y utilizar características particulares, denominadas *features*[1], para distinguir una clase de objetos del resto.

En la actualidad, la clasificación de imágenes juega un papel crucial en diversos sectores, como la robótica, la medicina, la videovigilancia y la industria. Estos modelos de visión por computadora han permitido automatizar tareas repetitivas y de alto esfuerzo en diferentes industrias. Por ejemplo, en el campo de la medicina, los potentes modelos de visión por computadora actúan como asistentes y respaldo para los profesionales médicos [21].

La importancia de la clasificación de imágenes radica en su capacidad para identificar y reconocer patrones visuales, lo que brinda oportunidades para el análisis y la

[1] Característica particular de un objeto, generalmente representado numéricamente en un espacio dimensional mas pequeño.

toma de decisiones basadas en imágenes. En el transcurrir de los años, la clasificación de imágenes se ha vuelto indispensable para extraer información significativa y facilitar la automatización en diversas aplicaciones. En este contexto, la optimización de los algoritmos de clasificación y el desarrollo de nuevos enfoques, como los modelos Transformers [5], se han convertido en áreas de investigación muy activas y de gran relevancia.

1.3. Motivación

El presente trabajo surge como un tema acordado con mi asesor y los motivos que me han llevado a realizarlo es el siguiente: el deseo de experimentar y entender la complejidad que implica trabajar con modelos de gran escala. Al mismo tiempo, conocer mas sobre métodos y técnicas necesarios para su entrenamiento. Además, este proyecto también representa una oportunidad para adentrarme de manera más profunda en este campo, con la meta de expandir mis conocimientos y habilidades, que seguro los aplicare en futuro cercano.

1.4. Objetivos

El objetivo principal de este trabajo de fin de máster es realizar un análisis mas profundo del rendimiento de tres modelos Transformer (*ViT*, *Swin* y *MaxViT*) preentrenados en *ImageNet* [3] y adaptados al conjunto de datos *NIH Chest X-rays* [25] para clasificar 14 patologías en radiografías de tórax. Se busca evaluar la eficacia de estos modelos en diferentes escalas y tamaños de imágenes, utilizando métricas como el área bajo la curva ROC (*AUC*), la velocidad de inferencia (*throughput*), la cantidad de parámetros y el numero de operaciones de punto flotante (*FLOPs*). Para lograrlo, se realizan los siguientes objetivos específicos:

1. Explorar el estado del arte en clasificación de imágenes para obtener un conocimiento más profundo sobre el funcionamiento de cada arquitectura Transformer empleada en este proyecto.
2. Adaptar y entrenar modelos preentrenados *ViT*, *Swin* y *MaxViT* en el conjunto de datos de imágenes médicas *NIH Chest X-rays*.
3. Configurar cada modelo en cuatro escalas distintas (*tiny*, *small*, *base*, *large*) y para tres resoluciones de imagen (224×224, 384×384 y 512×512) de entrada.
4. Evaluar el rendimiento de los modelos entrenados en términos del *AUC* utilizando el conjunto de datos previamente mencionado. Se compararán los resultados obtenidos por cada arquitectura en sus distintas escalas y tamaños de imágenes, lo que permitirá identificar la configuración con mayor desempeño.
5. Realizar un análisis exhaustivo del rendimiento en relación al número de parámetros, *FLOPs* y *throughput*, lo que permitirá obtener una comprensión más detallada de cada arquitectura.

1.5. Limitaciones

1. Únicamente se hará uso de modelos preentrenado dentro la librería Timm [26]

1.6. Estructura de la memoria

El presente trabajo se divide en 6 capítulos los cuales se detallan a continuación:

En el primer capítulo 1, se aborda una introducción general sobre imágenes médicas y la clasificación de imágenes, así como los objetivos y limitaciones de este trabajo. En el segundo capítulo 2, se realiza una revisión exhaustiva de la literatura que abarca trabajos previos relacionados con la clasificación de imágenes y componentes que forman parte de las arquitecturas *ViT*, *Swin* y *MaxViT*. El tercer capítulo 3 se explica los términos y métricas de evaluación en clasificación de imágenes, la transferencia de aprendizaje y las transformaciones utilizadas para aumentar los datos de entrenamiento, además de proporcionar una descripción detallada de las arquitecturas en estudio. En el cuarto capítulo 4, se detalla la metodología adoptada, incluyendo la descripción del conjunto de datos, su distribución, las técnicas de aumento de datos aplicadas, la adaptación de los modelos y los hiperparámetros empleados. El quinto capítulo 5 presenta los experimentos realizados, desde la obtención de hiperparámetros específicos hasta la evaluación de las arquitecturas mencionadas, lo que permitirá un análisis comparativo de sus rendimientos. Finalmente, se presentara las conclusiones 5.4 obtenidas a lo largo de este estudio.

1.7. Recursos técnicos

Para los experimentos realizados se hace uso de una maquina con un procesador 13th Gen Intel(R) Core(TM) i7-13700K, con 64 GB de memoria RAM, sistema operativo Ubuntu 22.04.1 LTS y una GPU RTX 4090.

Parte II.

Revisión de la literatura

2. Literatura

Con el transcurrir de los años, el crecimiento exponencial de la cantidad de información ha dado lugar al surgimiento de nuevas técnicas que nos permiten procesarlas de manera efectiva, el cual nos permiten comprenderlas y con ellas poder tomar decisiones mas acertadas. Una de estas técnicas es la clasificación de imágenes, la cual desempeña un papel fundamental en muchos sistemas automatizados de visión por computador. Sin embargo, este aumento masivo de datos ha llevado a la necesidad de desarrollar de manera mas frecuente modelos cada vez mucho mas enormes, con millones e incluso billones de parámetros. Por consiguiente, la elección de un modelo adecuado para una tarea determinada se hace muy complejo debido a la gran variedad de opciones disponibles, cada una requiriendo recursos computacionales en constante aumento.

La amplia variedad, complejidad y el tamaño de estos modelos presentan desafíos significativos, especialmente cuando se introducen en dominios sensibles como la medicina y entornos de alta demanda. En este contexto, comprender las capacidades de cada modelo desempeña un papel indispensable para abordar las complejidades mencionadas.

Esta revisión de la literatura explorara la evolución de los modelos de clasificación de imágenes, proporcionará conceptos bases para entender de manera mas efectiva los nuevos modelos del estado de arte basado en Transformer y con ello establecer una base solida para comprender y abordar los objetivos planteados en este estudio.

2.1. Redes Neuronales Convolucionales

Las redes neuronales convolucionales *CNN*, son algoritmos de aprendizaje profundo que se han convertido en un enfoque dominante en el procesamiento de imágenes. Estos algoritmos han logrado el estado del arte en diversas tareas, como la clasificación, detección y segmentación de imágenes. Las CNN funcionan eficazmente porque pueden aprender características espaciales locales y jerárquicas de manera eficiente. Además, los pesos aprendidos son compartidos por muchas neuronas, lo que hace que el entrenamiento sea más eficiente y reduce drásticamente el número de parámetros.

El aprendizaje profundo era un área relativamente inactiva hasta antes del 2012, cuando se introdujo AlexNet [12], una arquitectura que logró mejoras significativas en comparación con sus predecesores. Este éxito llamo nuevamente el interés en el campo por parte de los investigadores y, posteriormente, se han publicado una serie de artículos importantes. A continuación, describiremos algunos de ellos para situarnos en nuestro objetivo principal.

2.1.1. AlexNet

En el artículo, se describe que la arquitectura propuesta consta de 8 capas, de las cuales 5 son capas convolucionales y 3 son capas completamente conectadas. La última capa es una capa *softmax* con 1000 neuronas. Durante el entrenamiento, se aplicaron técnicas para reducir el sobreajuste (*overfitting*) como el aumento de datos(*data augmentation*) para generar datos de entrenamiento sintéticos y métodos de regularización, como *Dropout*, en las capas completamente conectadas. Se uso *rectified linear unit ReLU*, que proporciona una eficiencia computacional superior en comparación con la función de activación *tanh*. No obstante, pese a sus beneficios se tuvo que usar conjuntamente a *local response normalization* (*LRN*) para mejorar la generalización de la red.

2.1.2. VGG16

AlexNet fue una arquitectura muy prometedora en su momento, pero presentaba diversas limitaciones y requería un alto costo computacional para su entrenamiento. Como solución a estas limitaciones, surge *VGG16* [18]. Los autores destacan que las *CNN* generalizan de manera más efectiva cuando se utilizan modelos más profundos. Además, mencionan que el uso de tamaños de kernel más pequeños conduce a mejores resultados y, al mismo tiempo, reduce los costos computacionales. Por último, una característica importante de *VGG16* es la eliminación completa del uso de *LRN* (Local Response Normalization), lo cual produce un entrenamiento mas eficiente.

2.1.3. ResNet

Con el éxito de las *CNN* profundas, surgió el problema del desvanecimiento de gradiente (*vanishing gradient*), el cual ralentiza el entrenamiento y dificulta el aprendizaje. Esto ocurre cuando el gradiente no puede propagarse de manera efectiva desde las capas más profundas hasta las capas más superficiales. [7] proponen el uso de conexiones residuales para sobrellevar los problemas de desvanecimiento de gradiente y sobre ajuste, haciendo que el entrenamiento de modelos mas profundos sean factibles.

Una conexión residual es básicamente el aprendizaje de una función de identidad. Esto significa que la conexión residual aprende a agregar la entrada de una capa a la salida de la capa, sin cambiar la entrada de ninguna manera. Donde $H(x)$ es la salida de la red residual, x la característica de entrada y $F(x)$ es el mapeo aprendido por las capas apiladas en la red residual. Ver figura 2.1.

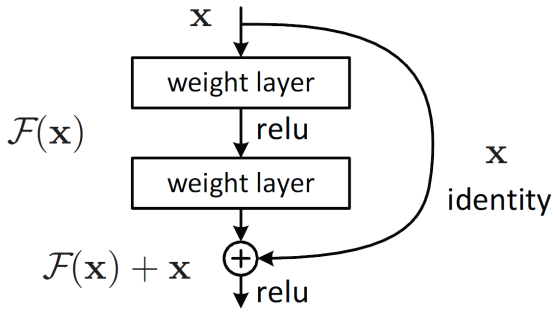


Figura 2.1.: Conexión residual
Fuente: [7]

2.1.4. MobileNets

Con el gran interés por parte de los investigadores en las *CNNs*, en el transcurso de los años se han propuesto diferentes arquitecturas para mejorar la eficiencia computacional y la precisión de los modelos. Entre ellos se encuentra *MobileNet* [8], que se centra en la idea de desarrollar modelos precisos que no requieran un alto poder computacional y que puedan ser desplegados en dispositivos con recursos limitados.

Una de las principales características de *MobileNet* es el uso de una nueva forma de realizar convoluciones, conocida como *Depthwise Separable Convolution*. Esta técnica divide la convolución en dos etapas. Primero, se aplica una *depthwise convolution* que realiza una convolución por cada canal de entrada. Luego, se aplica una *pointwise convolution*, que es una convolución de 1x1, para combinar los resultados de la convolución anterior. Esta estrategia reduce significativamente el número de operaciones y parámetros requeridos, el cual produce modelos más livianos y eficientes.

Al igual que *MobileNet*, existen otras técnicas predecesoras pioneras e importantes en enfocarse en el desarrollo de modelos grandes, mas precisos, pero que aprovechen los recursos computacionales de manera eficiente. InceptionNetwork [20] busca mejorar la utilización de los recursos computacionales al combinar diferentes filtros de convolución (1x1, 3x3, 5x5 y pooling), permitiendo que la red aprenda automáticamente a utilizar cada uno de ellos sin aumentar excesivamente la carga computacional. Por otro lado, DenseNet [10] a diferencia de los trabajos anteriores donde se buscaban crear modelos mas profundos o mas anchos para ganar precisión, se centra en la reutilización de características y aborda el problema del desvanecimiento de gradiente al conectar cada capa con todas las capas subsiguientes. Este enfoque permite un flujo de información más denso y facilita el aprendizaje de características en diferentes niveles de abstracción.

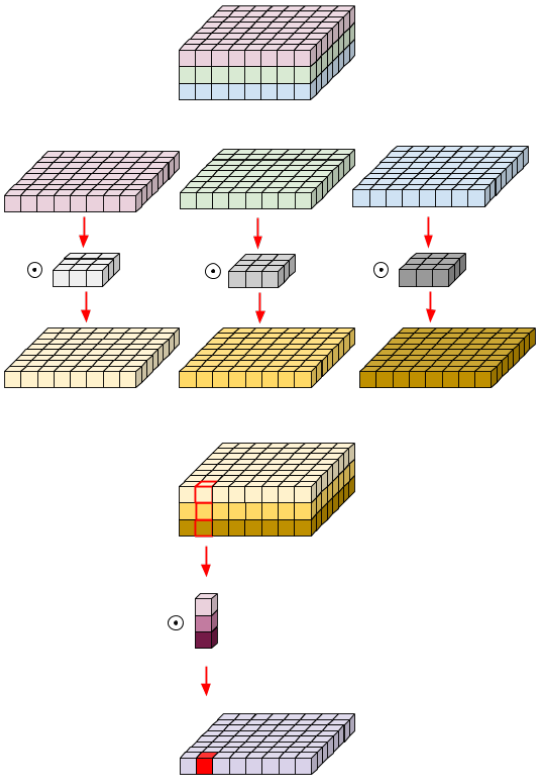


Figura 2.2.: Depthwise Separable Convolution
Fuente: [16]

2.1.5. Squeeze and Excitation Networks

En [9] se propone una arquitectura denominada *Squeeze-and-Excitation* (*SE*) para aprender de manera explícita la interdependencia entre los canales de salida de una convolución, lo que permite prestar mas atención a los canales relevantes y suprimir los menos importantes. El proceso de convolución se lleva en dos etapas. Primero se realiza *squeeze*, que tiene como objetivo reducir la dimensión espacial de los mapas de características. Esto se logra mediante una operación de pooling global (*Global average pooling*) aplicada a cada canal de características, lo cual produce un vector de características que representa la información global de cada canal.

Después se aplica *excitation*, que utiliza el vector de características obtenido en la etapa anterior para recalibrar adaptativamente las respuestas de cada canal. Para lograr esto, se toma el vector de características como entrada y alimenta a una pequeña red neuronal con dos capas. La primera capa comprime la información mediante un hiperparámetro de reducción r y la segunda capa restablece a la dimensión original. Además, actúa como una compuerta mediante una función de activación *sigmoid*, que produce un vector de pesos. Este último vector se multiplica por las características originales, ajustando así su importancia en el cálculo final. Ver figura 2.3.

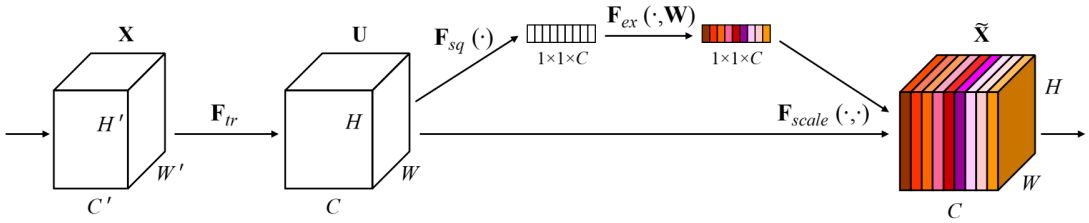


Figura 2.3.: Squeeze and Excitation
Fuente: [9]

2.2. Incorporación de Atención y Transformers

El campo del procesamiento de lenguaje natural (*NLP*) ha experimentado un rápido avance gracias a la introducción de modelos basados en atención y Transformers. Un trabajo influyente en este avance es *Attention Is All You Need* [24], donde se presenta el concepto de *self-attention*, un mecanismo de atención que difiere de otros al aprender a relacionar diferentes partes de una misma secuencia, en lugar de simplemente emparejar una entrada con una salida. Estos modelos han demostrado un rendimiento excepcional en diversas tareas de procesamiento de lenguaje, reemplazando en gran medida a los modelos recurrentes tradicionales. Además, han servido de inspiración para explorarlos en otros campos de investigación, tales como la clasificación de imágenes, detección y segmentación.

2.2.1. Vision Transformer

Inspirados por los avances en el (*NLP*), se ha experimentado con ideas y técnicas provenientes de este campo en la visión por computadora, logrando resultados muy prometedores. Un ejemplo destacado es el trabajo titulado *AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE* [5], donde se presenta *Vision Transformer* (*ViT*) para la clasificación de imágenes, arquitectura basada en Transformers. Este enfoque se basa en la idea de que las imágenes también pueden ser representadas como secuencias estructuradas, al igual que las secuencias de palabras en el *NLP*.

Para lograrlo, *ViT* divide la imagen de entrada en bloques no superpuestos denominados *patches*. Los bloques son convertido en una secuencia de características que se pasa a un modelo de autoatención (*Self attention*) [24] que aprende a relacionar la composición espacial de los píxeles. Esta técnica a resultado ser bastante efectiva en tareas de clasificación, superando a modelos convolucionales tradicionales del estado del arte en términos de rendimiento, además de requerir un poder computacional ligeramente menor.

2.2.2. Swin Transformer

A pesar de los avances logrados por *ViT* en el campo de la visión por computadora, existen ciertas limitaciones que dificultan su aplicabilidad como arquitectura

troncal en tareas de visión por computadora, especialmente para imágenes de alta resolución debido a su alto costo computacional, que aumenta cuadráticamente con el tamaño de la imagen. Además, los modelos de atención utilizados en *ViT* no se adaptan bien a las variaciones en la escala de las entidades visuales presentes en las imágenes, lo que produce un rendimiento limitado.

Para abordar estas limitaciones, se ha desarrollado *Swin Transformer* [13], una arquitectura jerárquica que emplea una estrategia de atención local en dos etapas (*W-MSA* y *SW-MSA*) 2.4. Primera etapa, *W-MSA* divide el mapa de características entrante en un conjunto de ventanas no superpuesta de tamaño fijo, luego se realiza la autoatención dentro de cada ventana. Segunda etapa, *SW-MSA* realiza una autoatención cruzada en función de la salida producida por *W-MSA*. Esta composiciones de bloques jerárquicos compuestos por *W-MSA* y *SW-MSA* supera a *ViT* y otros modelos del estado del arte en visión por computadora. Además, hace que *Swin Transformer* sea una arquitectura troncal adecuada para diversas tareas de visión por computadora, con un costo computacional lineal en función del tamaño de la imagen [13].

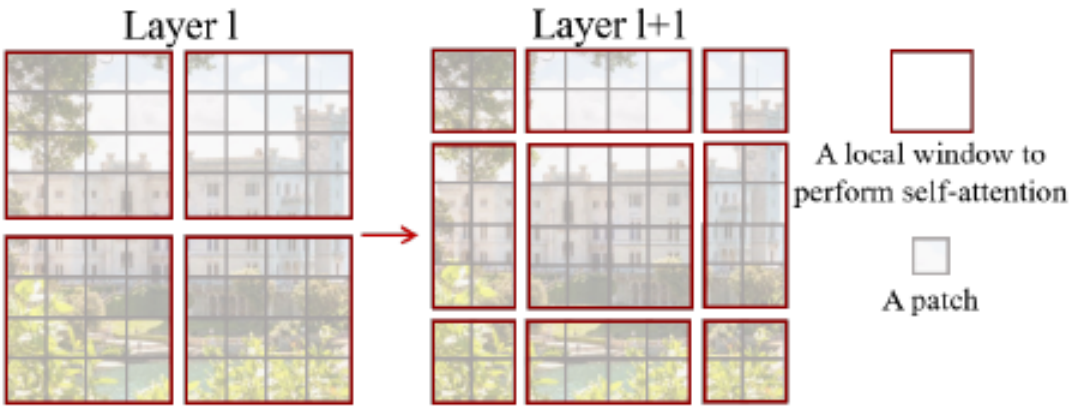


Figura 2.4.: Atención local en dos etapas, Swin Transformer
Fuente: [13]

2.2.3. Multi-Axis Vision Transformer

La arquitectura *Multi-Axis Vision Transformer* (*MaxViT*) [23] surge con el propósito de abordar los mismos desafíos que en *Swin Transformer*, es decir, mejorar la eficiencia computacional y servir como una arquitectura versátil para diversas tareas de visión por computadora.

Para lograr esto, se propone un nuevo mecanismo de atención denominado *Multi-axis attention*, que permite una interacción local y global en un tiempo computacional lineal en función del tamaño de la imagen. *Multi-axis attention* 2.5 opera en dos etapas. En la primera etapa, realiza una atención local, *Block attention*, similar a *W-MSA* de *Swin*, lo que permite captura la interacción local de los píxeles. Segunda etapa, se realiza una atención global denominada *Grid attention* que permite capturar la interacción global en un tiempo lineal.

Su arquitectura híbrida y jerárquica, de bloques de convolución y autoatención

(*MaxViT block*), hacen de su implementación muy sencilla y adecuado para funcionar como una arquitectura troncal en diferentes tareas de visión por computadora. Superando incluso a *Swin Transformer* en varias tareas de visión por computadora.

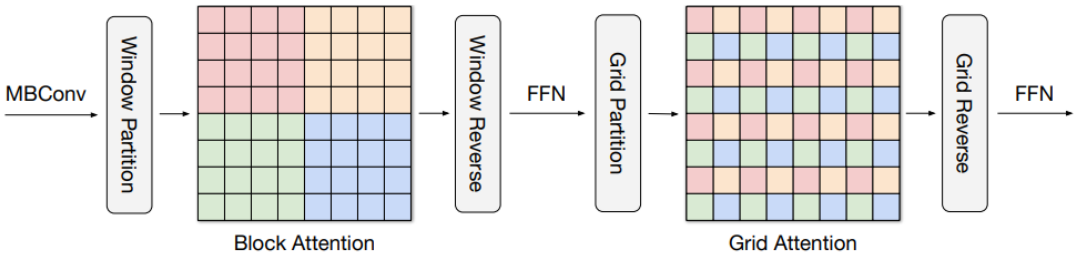


Figura 2.5.: Multi-axis attention, MaxViT Transformer
Fuente: [23]

El funcionamiento detallado de los modelos transformer descritos, incluyendo su arquitectura, algoritmos y otros temas relevantes necesarios para comprender y aplicarlos en tareas de visión por computadora, se explorará en el siguiente capítulo (Marco teórico 3).

Parte III.

Marco teórico

3. Marco conceptual

En este capítulo, se explicarán aspectos importantes como: métricas dentro de la visión por computadora, aumento de datos y arquitecturas *ViT, SWIN y MaxViT*; así como algunos puntos cruciales para entender los experimentos realizados, que se mostrarán en los capítulos siguientes.

3.1. Métricas en visión por computadora

Se define alguna de las principales terminologías utilizadas durante la evaluación de las prestaciones de un modelo de visión por computador.

- **Verdaderos positivos (TP)**: Muestras positivas correctamente identificadas por el modelo como positivo.
- **Verdaderos negativos (TN)**: Muestras negativas correctamente identificadas por el modelo como negativo.
- **Falsos positivos (FP)**: Muestra negativa incorrectamente identificada por el modelo como positivo.
- **Falsos negativos (FN)**: Muestra positiva incorrectamente identificada por el modelo como negativo.

3.1.1. Accuracy

Indica la proporción de muestras clasificadas correctamente.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

3.1.2. Precision

Mide la capacidad del modelo para clasificar muestras positivas que realmente son positivas. Es decir, es la cantidad de muestras positivas correctamente identificadas sobre el numero total de clasificaciones realizadas durante la prueba.

$$Precision = \frac{TP}{TP+FP} \tag{3.2}$$

3.1.3. Recall o Sensitividad

Indica la proporción de muestras correctamente clasificadas como positivo con respecto al numero total de muestras positivas del conjunto de entrenamiento.

$$Recall = \frac{TP}{TP+FN} \tag{3.3}$$

3.1.4. Especificidad

Indica la proporción de verdaderos negativos clasificados como negativos.

$$specificity = \frac{TN}{TN+FP} \tag{3.4}$$

3.1.5. Curva ROC y AUC

La curva *ROC* (del inglés *Receiver Operating Characteristic*) es una métrica que evalúa el rendimiento de un modelo de clasificación binaria en diferentes umbrales de clasificación. Esta curva se genera utilizando las ecuaciones de sensibilidad 3.3 en el eje "Y" y la especificidad 3.4 en el eje "X", ver Figura 3.1.

Aunque la curva *ROC* está diseñada para clasificadores binarios, también puede ser utilizada en clasificadores multiclase mediante el enfoque "uno contra todos", donde cada clase de interés se considera positiva y el resto de clases se consideran negativas.

El área bajo la curva (AUC) mide la capacidad de discriminación de un modelo. Es decir, dado dos ejemplos, uno positivo y otro negativo, el AUC es la probabilidad de que un modelo los clasifique correctamente. Un valor de $AUC > 0{,}5$ indica que el clasificador es mejor que uno aleatorio, siendo un $AUC = 1{,}0$ indicativo de un clasificador perfecto. Por otro lado, un $AUC = 0{,}5$ indica un clasificador aleatorio, mientras que un $AUC < 0{,}5$ sugiere que el clasificador está invirtiendo la clasificación.

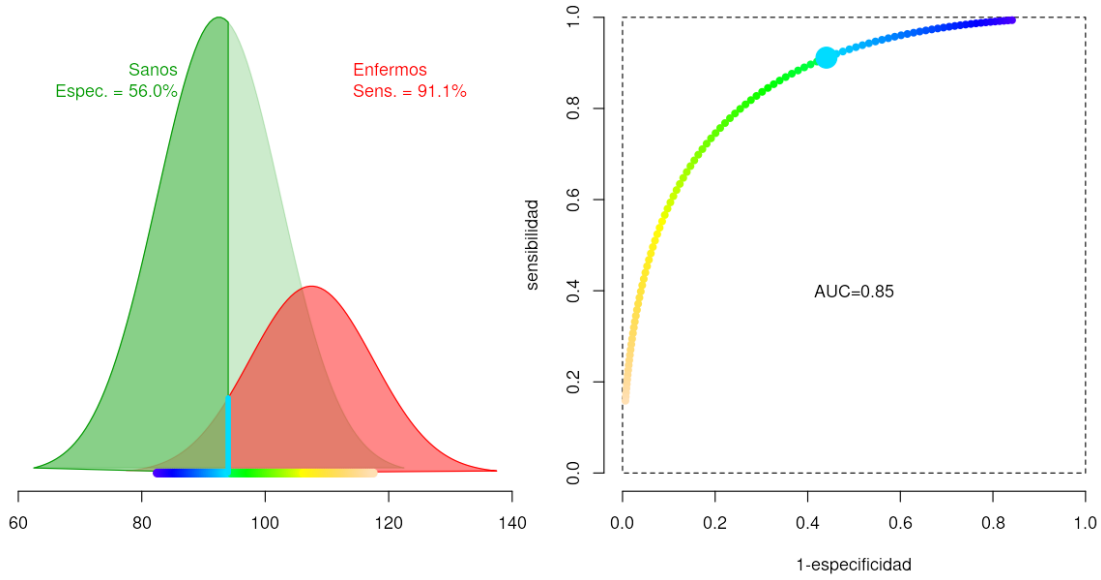


Figura 3.1.: AUC-ROC
Fuente

3.1.6. *FLOPs*

FLOPs u operaciones de punto flotante, es una métrica que mide la cantidad total de operaciones aritméticas de punto flotante realizadas por un algoritmo o modelo al procesar una única instancia de entrada. Esta métrica es fundamental para evaluar la complejidad computacional (tiempo, memoria y capacidad de procesamiento) y comprender la carga de trabajo que implica. Cuanto menor sea el valor de *FLOPs* para un modelo, menor será la carga computacional, por lo tanto, más eficiente será el modelo.

Por ejemplo, sea una imagen "x", RGB, de dimensión $(H \times W \times 3)$, el número de operaciones de punto flotante realizados por una convolución se determina mediante la fórmula: $Conv_{FLOPs}(x) = F \times H' \times W' \times k_h \times k_w \times k_3$.

Donde, $(H' \times W' \times F)$ son las dimensiones la imagen después de aplicar la convolución y $k = (k_h \times k_w \times k_c)$ es el tamaño del kernel usado.

3.1.7. *Throughput*

Throughput o rendimiento, hace referencia a la cantidad de trabajo que un sistema o dispositivo puede realizar en un período de tiempo específico. En el contexto de la clasificación de imágenes, se refiere a la cantidad de imágenes procesadas por segundo durante la inferencia [22]. Un alto *throughput* indica que el modelo puede procesar más datos de manera más rápida, lo que es fundamental en aplicaciones que requieren de un alto rendimiento.

3.2. Transferencia de aprendizaje

La transferencia de aprendizaje es una técnica que ha ganado mucha relevancia en el campo del aprendizaje automático. Es utilizada de forma habitual al entrenar modelos y resulta muy útil en diversas situaciones. Consiste en aprovechar un modelo ya entrenado en una tarea específica y aplicarlo en otra tarea distinta. Esto ofrece beneficios importantes, como la reducción del tiempo y los recursos computacionales necesarios para entrenar un nuevo modelo, además de proporcionar un desempeño muy prometedor en tareas donde el conjunto de entrenamiento es limitado.

3.3. Aumento de datos

El aumento de datos o *data augmentation* es una técnica que aumenta el tamaño del conjunto de entrenamiento mediante la aplicación de diversas transformaciones a las imágenes originales. Estas transformaciones pueden incluir cambios geométricos[1] y fotométricos [2]. El resultado es un conjunto de entrenamiento más diverso y variado con el cual se pueden entrenar modelos mas robustos.

El aumento de datos no solo incrementa la cantidad de datos disponibles para el entrenamiento, sino que también actúa como una forma de regularización. Al expandir la diversidad de los datos, se evita que el modelo se sobre ajuste (*overfitting*), es decir, que se ajuste demasiado a los datos de entrenamiento y pierda la capacidad de generalizar correctamente en datos nuevos o no vistos.

En esta sección se presentarán las transformaciones utilizadas durante el entrenamiento para aumentar los datos y mejorar el rendimiento de los modelos evaluados en los siguientes capítulos.

1. **Crop y resize:** Selecciona una área aleatorio de la imagen y luego la redimensiona a un tamaño deseado.
2. **Horizontal Flip:** Refleja la imagen en función del eje horizontal.
3. **Equalize:** Ecualiza el histograma de la imagen, lo que produce un realce de características con baja iluminación.
4. **Rotate:** Rota la imagen en función de un ángulo aleatorio dentro de un rango especificado.
5. **Solarize:** Invierte los valores de píxeles que se encuentran por encima de un umbral.

[1] La transformación geométrica de imágenes se refiere al proceso de alterar las propiedades geométricas de una imagen, como su forma, tamaño, orientación o posición. Implica aplicar operaciones matemáticas a los píxeles o coordenadas de la imagen para lograr la transformación deseada [17]

[2] La transformación fotométrica de imágenes se refiere al proceso de modificar las propiedades fotométricas de una imagen, como su brillo, contraste, color o tono. Estas transformaciones se aplican para cambiar la apariencia visual de una imagen conservando su estructura geométrica [17]

6. **Contrast:** Ajusta el contraste de una imagen, lo que produce un realce o una atenuación de algunas características de la imagen.
7. **Brightness:** Ajusta el brillo de la imagen. Puede producir imágenes mas claras o mas oscuras.
8. **Shear:** Realiza un recorte aleatorio en algún punto del eje horizontal o vertical respectivamente, produce una distorsión de la imagen.
9. **Translate:** Traslada la imagen en una fracción de su altura o anchura.
10. **Gaussian Blur:** Suaviza los bordes y reduce el ruido de la imagen.
11. **Gaussian Noise:** agrega ruido aleatorio a la imagen siguiendo una distribución normal.
12. **Erasing**: Elimina los píxeles dentro de una región elegida aleatoriamente.

3.4. Arquitecturas de clasificación

3.4.1. Visión transformer

Vision Transformer se compone de tres bloques principales: *patch embedding*, una serie de *transformer encoder* y un *MLP head*. Estos bloques trabajan en conjunto para llevar a cabo la tarea de clasificación. Ver figura 3.2.

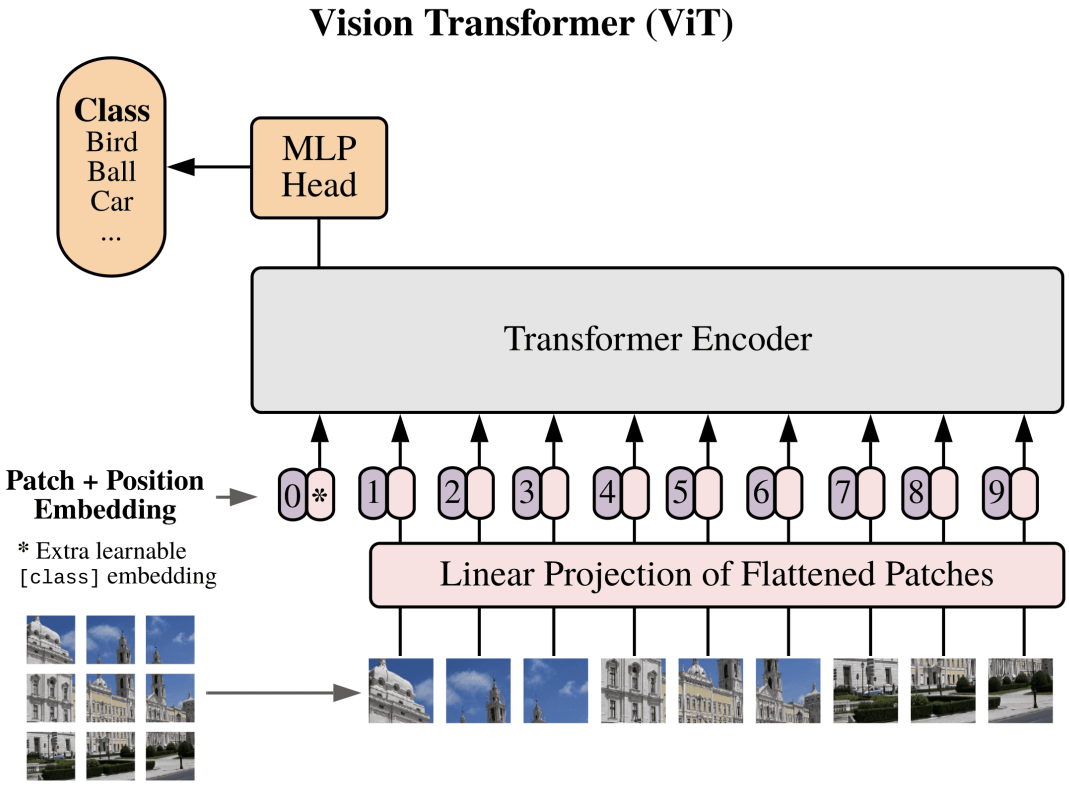


Figura 3.2.: Arquitectura Vision Transformer
Fuente: [5]

1. ***Patch Embedding*:** Consiste en proyectar una imagen $x \in \mathbb{R}^{H\times W\times C}$ a un nuevo espacio dimensional $\mathbb{R}^{N+1\times D}$, donde (H, W) es la resolución de la imagen original, C es el número de canales, N es el número de patches obtenidos de la imagen y D es la dimensión del vector latente al cual ha sido proyectado.

   a) Primero se divide la imagen x en N bloques de $(P \times P \times C)$ no superpuestos, donde $N = \frac{HW}{P^2}$.

   b) Se reestructura cada bloque $(P \times P \times C)$ en un vector, lo que resulta en un mapa de características $x_p \in \mathbb{R}^{N\times(P^2\times C)}$.

   c) x_p se multiplica mediante un *embedding* $\mathbf{E} \in \mathbb{R}^{(P^2\times C)\times D}$, que aprende a proyectar cada vector del mapa de características a un nuevo espacio dimensional $\mathbf{Z_0} \in \mathbb{R}^{N\times D}$. La dimensión D es constante en todas las capas posteriores.

   d) Se agrega un token $x_{cls} \in \mathbb{R}^{1\times D}$ como prefijo de $\mathbf{Z_0}$, lo que produce un nuevo $\mathbf{Z_0} \in \mathbb{R}^{N+1\times D}$. El token x_{cls} aprenderá a representar cada imagen de entrada y se usará posteriormente en la última capa de nuestro codificador para la clasificación.

   e) Para aprender a representar la información espacial de la imagen, se agrega un embedding posicional $\mathbf{E_{pos}} \in \mathbb{R}^{(N+1)\times D}$, el cual se suma a $\mathbf{Z_0}$.

   Lo anterior se resume en la figura 3.3, tal como se indica en [5].

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\, \cdots ;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D},\, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D}$$

Figura 3.3.: Patch embeding
Fuente: [5]

2. ***Transformer encoder*:** En la etapa de codificación se procesa la salida de la capa anterior $\mathbf{Z}_{l-1}$ mediante las siguientes ecuaciones 3.5.

$$\begin{aligned} z_l^{'} &= MSA(LN(z_{l-1})) + z_{l-1}, \quad l = 1 \dots L \\ z_l &= MLP(LN(z_l^{'})) + z_l^{'}, \quad l = 1 \dots L \end{aligned} \tag{3.5}$$

   Aquí, *LN* representa una capa de normalización que normaliza cada ejemplo de entrenamiento mediante su media y desviación estándar, y luego lo escala mediante dos parámetros aprendidos, lo que acelera el entrenamiento [1]. *MSA* es un *Multihead Self Attention*, que aprende a relacionar las características espaciales en diferentes subespacios de representación [24]. El símbolo $+$ denota una conexión residual, que previene el desvanecimiento del gradiente en redes profundas [7]. Finalmente, *MLP* se refiere a un perceptrón multicapa (2 capas) con *Gaussian Error Linear Units* (GELU) como función de activación.

   Es importante resaltar que la estructura más relevante dentro de un *transformer encoder* es el *MSA* por lo que es vital comprenderlo y analizarlo en mayor detalle.

- **Multihead selft attention:** El *MSA* es una extensión de la autoatención en la que se ejecuta "k" operaciones de autoatención, llamados *Heads*, en paralelo, y se proyecta sus salidas concatenadas [5].

  Dado una entrada $\mathbf{Z_{l-1}} \in \mathbb{R}^{(N+1)\times D}$ proveniente de la capa anterior del modelo transformer, la segunda dimensión D se divide por el número de *heads*, y cada una de estas divisiones son proyectadas hacia tres matrices:*query*, *key* y *value* $(\mathbf{q}, \mathbf{k}, \mathbf{v})$ respectivamente, mediante el producto escalar entre $U_{qkv} \in \mathbb{R}^{D\times 3D_h}$, las matrices resultantes representan la proyección de la entrada en tres subespacios diferentes, $\mathbf{qkv} \in \mathbb{R}^{(N+1)\times D_h}$, donde $D_h = \frac{D}{k}$. Las tres submatrices se utilizan en la operación de autoatención 2, los resultados de cada *head* se concatenan y se multiplican por la matriz $U_{msa} \in \mathbb{R}^{k.D_h\times D}$, obteniendo así la salida $MSA(\mathbf{Z_{l-1}}) \in \mathbb{R}^{(N+1)\times D}$, que aprende la información de todo los *heads* ("global attention").

$$MSA(Z) = [SA_1(z); SA_2(z); \ldots; SA_k(z),]U_{msa}, \quad U_{msa} \in \mathbb{R}^{k.D_h\times D} \tag{3.6}$$

- ***Self Attention (SA)*:** La autoatención o *Self Attention (SA)* 3.4 es un mecanismo de atención que permite relacionar diferentes partes de una secuencia entre sí [24].

$$\begin{gathered} A = softmax(qk^T/\sqrt{D_h}); \quad A \in \mathbb{R}^{N\times N} \\ SA(z) = Av; \quad SA \in \mathbb{R}^{N\times D_h} \end{gathered} \tag{3.7}$$

  Para lograrlo, primero se realiza un *softmax* del producto escalar entre las matrices $\mathbf{q}$ y $\mathbf{k}$, normalizado por $\sqrt{D_h}$, para obtener la matriz de atención A. Este paso busca identificar las secciones más relevantes dentro de la imagen en función de sus similitudes.

  Finalmente, se realiza la operación de *Self Attention (SA)* multiplicando la matriz de atención A por la matriz $\mathbf{v}$, obteniendo así la salida SA. Esta salida representa la relación aprendida entre los diferentes elementos de la imagen.

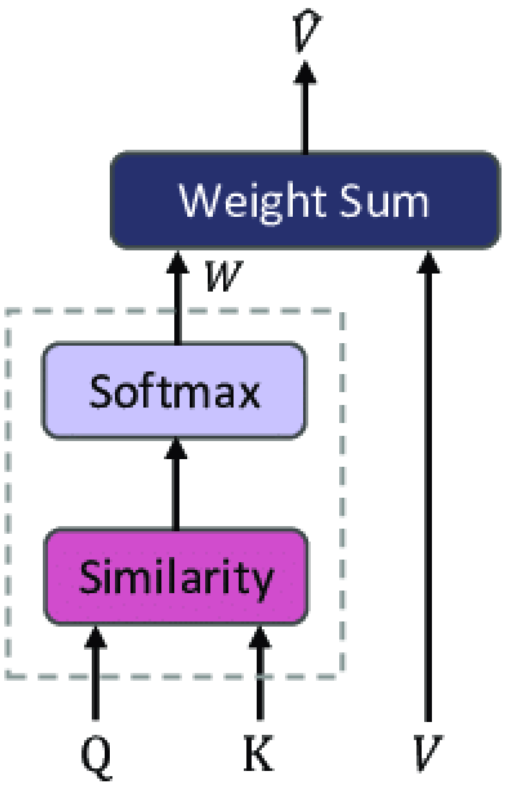


Figura 3.4.: Representación de la autoatención
Fuente

3. ***MLP head*:** La clasificación se realiza sobre el token agregado $x_{cls} \in \mathbb{R}^{1\times D}$ mediante un perceptrón multicapa de dos capas.

   Primero, el token x_{cls} se proyecta hacia una nueva dimensión D_{mlp} mediante una multiplicación con la matriz de pesos $W_0 \in \mathbb{R}^{D\times D_{mlp}}$. Luego, se realiza una segunda proyección hacia la dimensión del número de clases a clasificar, D_{n_cls}, mediante otra multiplicación con la matriz de pesos $W_1 \in \mathbb{R}^{D_{mlp}\times D_{n\_cls}}$. La salida es un vector de probabilidades $y \in \mathbb{R}^{1\times D_{n\_cls}}$, que indica el nivel de pertenecía hacia cada clase.

Una vez entendido las componentes principales, se describe el funcionamiento de la arquitectura. *ViT* toma una imagen y la divide en un conjunto de bloques no superpuestos de tamaño fijo que recubren toda la imagen de entrada. Estos bloques se reorganizan en vectores unidimensional, y cada vector unidimensional se proyecta en un espacio dimensional aprendido, luego, se agrega un token de clasificación al inicio de estas proyecciones.

Una vez que se ha agregado el token de clasificación, se añade información posicional unidimensional a cada proyección para mantener su referencia espacial. A todo este proceso se le denomina como *patch embeddings*.

Los *patch embeddings* resultantes se alimentan a un *transformer enconder* [24] donde se aprende a relacionarlos mediante un mecanismo de auto atención global.

Finalmente, para realizar la clasificación, se utiliza un perceptrón multicapa (*MLP*

head) sobre la salida del transformer, específicamente sobre el token de clasificación. El perceptron multicapa produce las probabilidades de pertenencia de la imagen de entrada a una clases de interés.

3.4.2. SWIN transformer

La arquitectura *SWIN Transformer* consta de 3 elementos principales: *Patch Embedding*, compuesto por la capa *patch partition* y *linear embeding*; *Swin Transformer Block*, *Patch Merging* y *Image Classification*. Estos elementos trabajan en conjunto para lograr un rendimiento superior a modelos previos como *ViT*, y además, presentan un costo computacional lineal en el modelo de atención en función del tamaño de la imagen. La interacción de estos bloques se muestra en la figura 3.5.

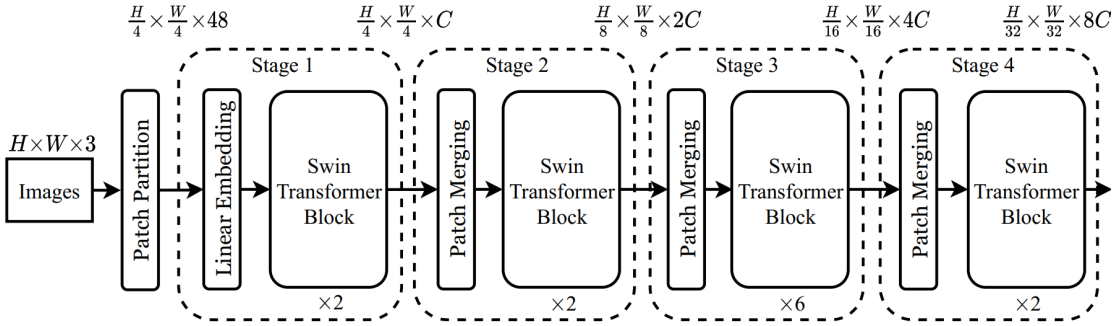


Figura 3.5.: Arquitectura SWIN transformer
Fuente: [13]

1. ***Patch Embedding*:** Al igual que en *VIT* (ver sección 3.4.1), esta etapa consiste en proyectar una imagen $x \in \mathbb{R}^{H\times W\times 3}$ en un nuevo espacio dimensional $x \in \mathbb{R}^{\frac{H}{4}\times\frac{W}{4}\times C}$.

   Se agrupa la imagen de entrada en bloques 4×4, no superpuesto, que recubren la imagen de entrada. Cada bloque es denominado como *patch*. Seguidamente, cada *patch* es reordenado en vectores de dimensión $1\times(4\times4\times3)$, lo que produce un mapa de características de tamaño $\frac{H}{4} \times \frac{W}{4} \times 48$. Finalmente, cada *patch* es proyectado a un nuevo espacio dimensional C (*linear embedding*). Al final, el mapa de características tiene una dimensión $\frac{H}{4} \times \frac{W}{4} \times C$.

   Para realizar ello, se utiliza una convolución con un filtro y stride $k = S = 4$; y un número de filtros C. Lo que se traduce como la proyección de cada $patch = 4 \times 4 \times 3 = 48$ a C canales. Ver figura 3.6.

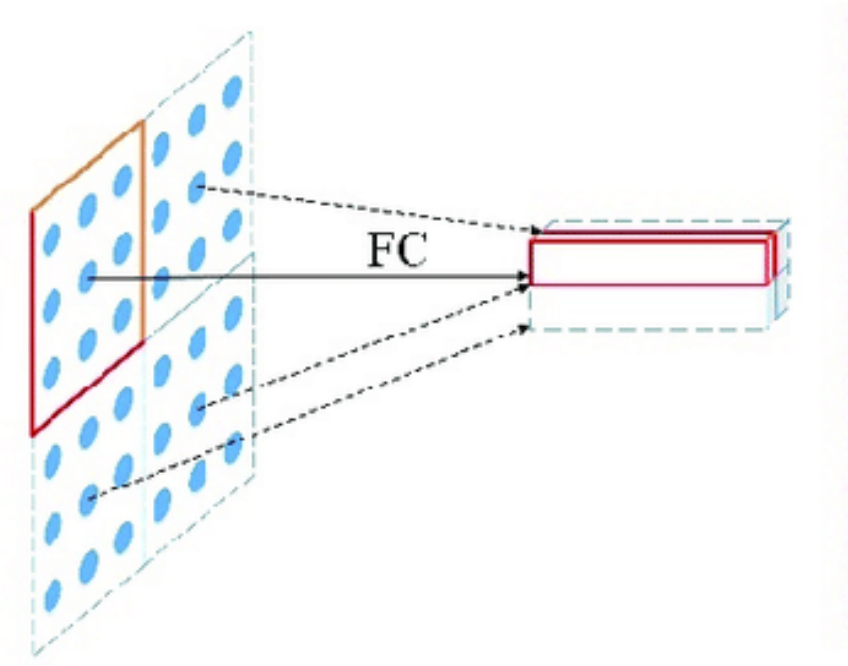


Figura 3.6.: Patch Embedding
Fuente

Esta serie de pasos correspondería a la etapa *patch Partition* y *Linear Embedding* (división y proyección) de la arquitectura *SWIN* .

***Swin transformer block*:** Un *Swin block* está formado por capas de normalización (*LN*), perceptrón multicapa (*MLP*), conexiones residuales ($\oplus$) y dos elementos importantes denominados *windows-based multi self-attention* (*W-MSA*) y *shifted window multi self-attention* (*SW-MSA*). Estos elementos se organizan en dos subbloques que se ejecutan de manera secuencial, como se muestra en el siguiente gráfico:

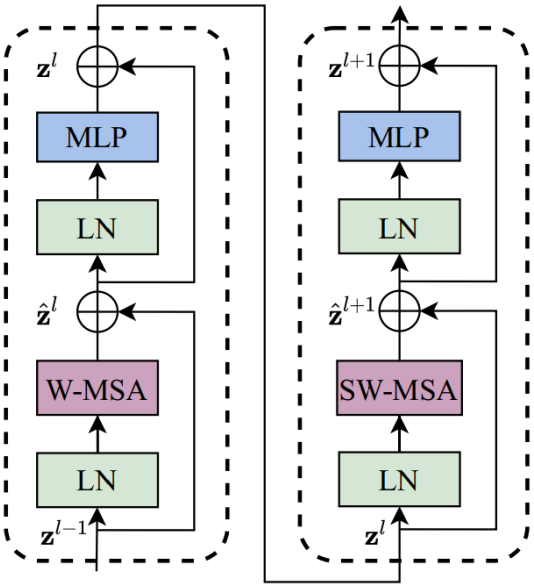


Figura 3.7.: Bloque *SWIN*
Fuente: [13]

En este apartado se detallaran únicamente el funcionamiento de los dos mecanismos importantes: *W-MSA* y *SW-MSA*. El funcionamiento del resto de elementos ya se ha explorado en *ViT* (ver sección 3.4.1).

La representación gráfica de cada ítem que se describe a continuación, se encuentra en la imagen 3.8.

a) ***W-MSA*:** Los *patches* Z_{l-1} de la capa anterior se agrupan mediante ventanas de tamaño fijo $M \times M$, no superpuestas, que recubren Z_{l-1}. Dentro de cada ventana, se aplica el mecanismo de auto atención, el cual se conoce como atención local (*local attention*). Esto permite que el modelo aprenda a relacionar la información espacial entre *patches*, dentro de cada ventana. Ver figura 3.8

***SW-MSA*:** Al aplicar *W-MSA*, se observa una carencia de interacción entre características de las ventanas adyacentes, lo que dificulta al modelo relacionar patrones entre ellas. Para abordar este problema, se introduce *SW-MSA*, que toma la salida de *W-MSA* y aplica ventanas de tamaño $M \times M$, pero desplazadas en un factor de $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$. Este desplazamiento genera nuevas ventanas de diferentes dimensiones, lo que resulta en un mayor cómputo e incremento en el número de ventanas.

Para solucionar este incremento, los autores aplican un enfoque denominado *cyclic-shifting*. Ahora, consiste en desplazar los parches hacia el lado superior izquierdo, logrando una reordenación de los parches para que puedan ser procesados de manera eficiente y con un costo lineal, al igual que en *W-MSA*.

b) ***Masked MSA*:** *SW-MSA* tiene inconvenientes al momento de aplicar la atención debido a que, al realizar *cyclic-shifting*, existen ventanas

conformadas por regiones diferentes que no tienen relación alguna. Para solucionar esto, se crea una máscara que indica al modelo de atención qué píxeles están relacionados y cuáles no. Esta máscara garantiza que solo se consideren las regiones adyacentes y se descarten las conexiones incorrectas.

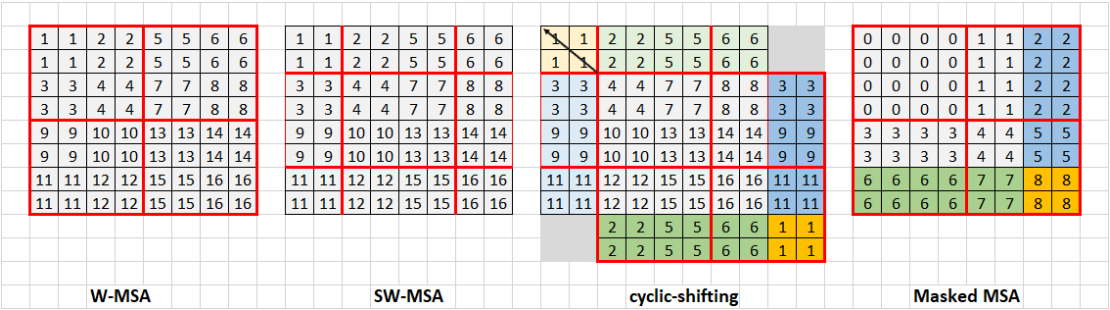


Figura 3.8.: Representación gráfica de *W-MSA*, *SW-MSA* y subcomponentes

2. ***Patch Merging*:** La capa *Patch Merging* tiene la función de reducir la dimensión y expandir el número de canales del mapa de características entrante. En esta etapa, se aprende la información jerárquica de los objetos presentes en la imagen. Para lograrlo, se concatenan las características de cada grupo de *patches* vecinos pertenecientes a una ventana de tamaño 2×2, lo que produce que el mapa de características decremente su tamaño en un factor de dos e incremente el número de canales en un factor de cuatro. Posteriormente, se normaliza y se proyecta a un espacio que reduce el numero de canales en un factor de dos. Este proceso se resume en la imagen 3.9.

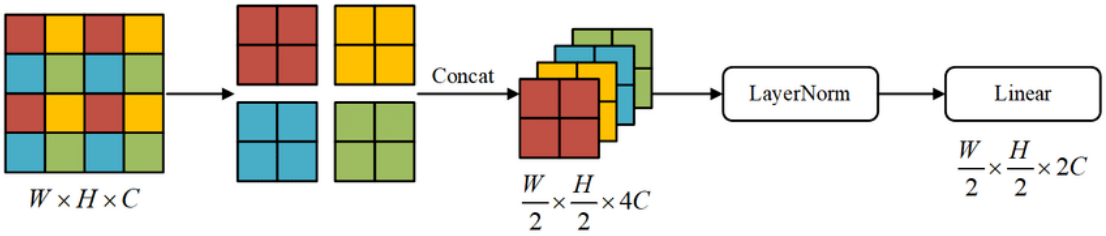


Figura 3.9.: *Patch Merging*
Fuente

3. ***Image classification*:** Para llevar a cabo la clasificación, se realiza una transformación a la salida de la arquitectura *SWIN transformer*, cuya dimensión es $\frac{H}{32} \times \frac{W}{4} \times 8C$, para convertirla en un vector. Este vector es procesado mediante un perceptrón multicapa que se encarga de realizar la tarea de clasificación.

La arquitectura SWIN está compuesta por 4 etapas, organizadas secuencialmente, que se repiten n veces independientemente según el tamaño del modelo. Cada etapa está precedida por un *patch merging*, seguido de un *Swin block*, con excepción de la primera etapa, donde se elimina el *patch merging*. En cada etapa en la que existe un *patch merging*, se reduce y se expande en un factor de dos la dimensión y los canales del mapa de características entrante, respectivamente.

Al igual que en *ViT*, la imagen de entrada se convierte en *patch embeddings*. A diferencia de *ViT*, *SWIN* no agrega ningún token de clasificación ni un token posicional. Sin embargo, agregar un token posicional puede mejorar el rendimiento, según indican los autores. Luego, los *patch embeddings* pasan a través de todas las etapas de la arquitectura, lo que resulta en una salida similar a la de otros modelos, como *ResNet* o *VGG*, que son troncales de muchas arquitecturas. Por lo tanto, para realizar una clasificación, es necesario transformarlos mediante un *average pooling* u otro tipo de transformación que nos devuelva un vector que posteriormente será usado por la capa de clasificación.

3.4.3. Multi-axis vision transformer

MaxViT es una arquitectura híbrida que combina redes convolucionales con el bloque *Multi-axis self attention*. Este bloque descompone la atención en dos etapas (atención global y local), lo que reduce significativamente el costo computacional cuadrático observado en *ViT*. La descomposición se logra mediante transformaciones en el eje espacial, lo que permite un costo computacional lineal en función del tamaño del mapa de características de entrada en el modelo de atención.

La arquitectura *MaxViT* esta compuesto por el apilamiento jerárquico de bloques MBConv y *Multi-axis self attention* (*Max-SA*). Esta composición hace a *MaxViT* una arquitectura simple y versátil, adecuado para funcionar como arquitectura troncal en muchas tareas de visión por computadora. Una representación gráfica de la arquitectura se observa en la figura 3.10.

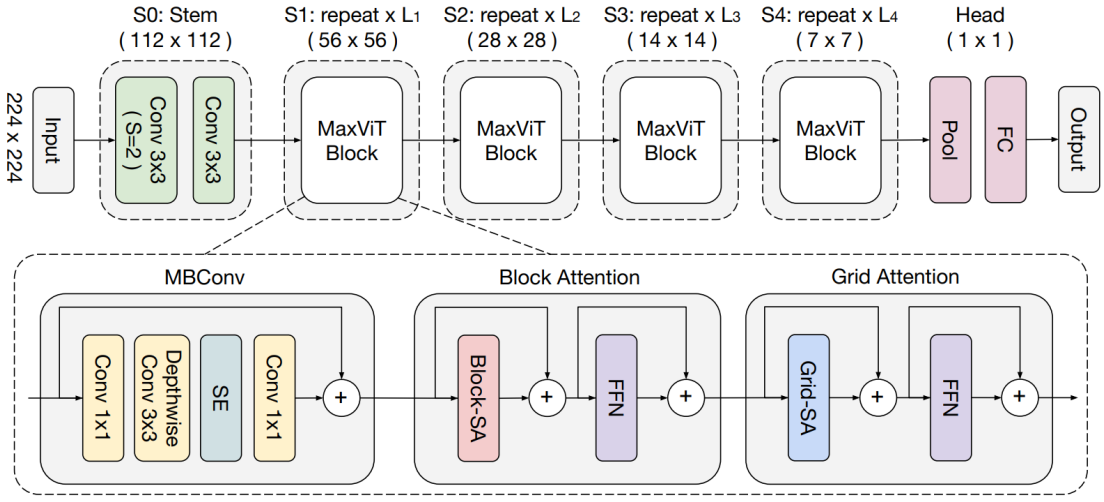


Figura 3.10.: Arquitectura *MaxVIT*
Fuente: [23]

1. ***MBConv*:** La composición de capas en el bloque *MBConv* se define mediante la siguiente ecuación:

$$MBConvA(x) = x + Proj(SE(DWConv(Conv(Norm(x))))) \tag{3.8}$$

Donde, $x \in \mathbb{R}^{H \times W \times C}$ es un mapa de características. $Norm$ es una capa de

normalización por lote de entrada [11]. *Conv* una capa de convolución 1×1 encargado de expandir el numero de canales en un factor de $4C$. *DWConv* es una capa de convolución *Depth-wise* 2.1.4 con un filtro de tamaño $k = 3$. *SE* es una capa *Squeeze and excitation* 2.1.5 que resalta los canales mas relevantes de la característica de entrada, el factor de reducción usado es $r = 0{,}25$. *Proj* es otra convolución 1×1 que proyecta el numero de canales resultante a una dimensión inferior C. Al final, el resultado se une con el mapa de características de entrada mediante una conexión residual.

MaxViT, al ser una arquitectura jerárquica, necesita reducir la dimensión y expandir el número de canales de la característica de entrada a medida que la red se vuelve más profunda. Por tanto, se realiza una modificación a la ecuación anterior que nos permitirá realizar lo descrito, la nueva variante se muestra a continuación:

$$MBConvB(x) = Proj(Pool2D(x)) + Proj(SE(DWConv \downarrow (Conv(Norm(x))))) \tag{3.9}$$

Estas modificaciones implican que *DWConv* ahora reduce el tamaño de las características de entrada en un factor de dos, mediante un $stride = 2$. Por otro lado, la dimensión de la característica de entrada se reducen en un factor de dos mediante un *pooling 2D* y posteriormente se proyecta el canal a una dimensión superior $2C$. Ambas salidas se combinan mediante una conexión residual para formar el resultado final del bloque *MBConv*.

2. ***Multi-axis self attention*:** La descomposición de la auto atención se realizada en dos subbloques secuenciales denominados *Block-attention* y *Grid-attention*.

   Block-attention realiza el mismo proceso que *W-MSA*, visto en *SWIN transformer*. Divide un mapa de características $X \in \mathbb{R}^{H \times W \times C}$ en bloques no superpuestos de tamaño $P \times P$, donde en cada bloque se realiza la auto atención local. Esto se resume en la siguiente secuencia de transformaciones:

$$\begin{gathered} Block : (H, W, C) \rightarrow (\tfrac{H}{P} \times P, \tfrac{W}{P} \times P, C) \rightarrow (\tfrac{HW}{P^2}, P^2, C) \\ X = X + Unblock(Attention(Block(LN(X)))) \\ X = X + MLP(LN(X)) \end{gathered}$$

   En la primera línea se observa la forma en cómo se descompone el mapa de características entrante en bloques de $P \times P$. La segunda y tercera línea muestran la sucesión de transformaciones utilizadas. LN es una normalización. *Block* descompone el mapa de características en bloques de $P \times P$. *Attention* es la auto atención aplicada en cada bloque. *Unblock* restablece el conjunto de bloques a su dimensión original ($H \times W \times C$). MLP es un perceptrón multicapa con dos capas ocultas.

   Grid-attention realiza cortes en cuadrículas distribuidos uniformemente por todo el espacio del mapa de características. Para ello, se define el tamaño de la cuadrícula $G \times G$, y cada elemento de la rejilla está distribuido

uniformemente por todo el eje espacial. La atención se realiza entre las celdas de la cuadrícula que están distribuidas uniformemente.

Por ejemplo, dado un mapa de características de tamaño 8×8 y una cuadrícula $G = 2 \times 2$, ver figura 3.11, la atención se realizaría entre todas las celdas del mismo color. A este enfoque se le denomina como una atención global dispersa.

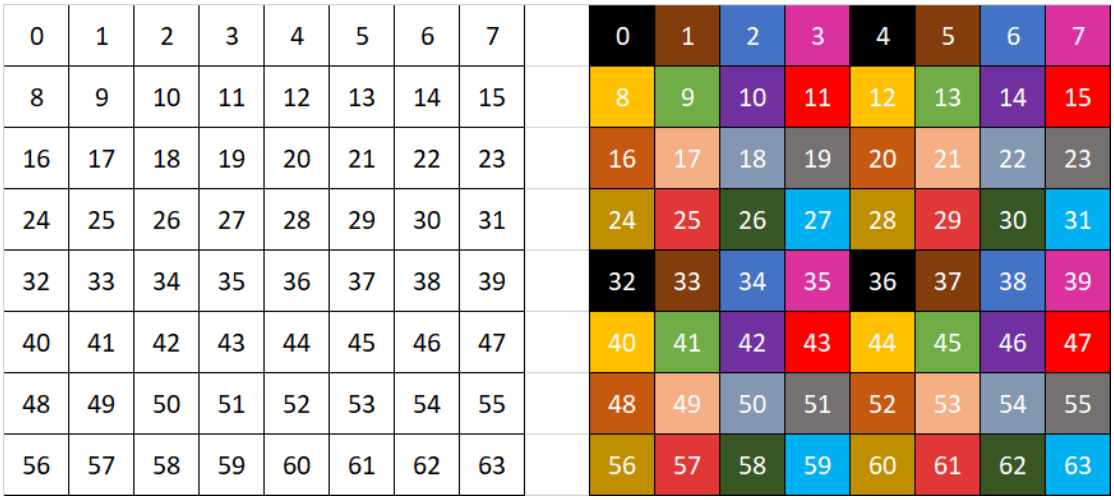


Figura 3.11.: *Grid attention*

Para lograrlo, se sigue la siguiente secuencia de transformaciones que se observan en la ecuación 2. Donde, *Grid* es la transformación que nos permite redistribuir el eje espacial en cuadriculas y *Ungrid* devuelve las transformaciones a su dimensión original.

$$Grid : (H, W, C) \rightarrow (G \times \tfrac{H}{G}, G \times \tfrac{W}{G}, C) \rightarrow \underbrace{(G^2, \frac{HW}{G^2}, C) \rightarrow (\frac{HW}{G^2}, G^2, C)}_{Permutacin}$$

$$X = X + Ungrid(Attention(Grid(LN(X))))$$
$$X = X + MLP(LN(X))$$

MaxVit está compuesto por un tallo (*Stem*), una serie de bloques *MaxViT block* ($MBConv \rightarrow MaxSA$) agrupados en cuatro etapas que se repiten "n" veces, independientemente uno de los otros; y una capa de clasificación. Todos ellos están apilados jerárquicamente en ese orden para realizar las tareas de clasificación.

El tallo toma una imagen $x \in \mathbb{R}^{H \times W \times C}$ y la proyecta a un nuevo espacio dimensional $x_p \in \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times k.C}$ mediante dos convoluciones. Luego, x_p es pasado a través de la serie de etapas conformadas por *MaxViT block*. Al inicio de cada etapa, el mapa de características es reducido en un factor de dos mediante *MBConvB*. Al final, el resultado es pasado hacia la capa de clasificación.

Parte IV.

Desarrollo del proyecto

4. Dataset y arquitecturas de clasificación

4.1. NIH Chest X-rays dataset

El conjunto de datos *NIH Chest X-rays* (*ChestX-ray14*) es uno de los datasets más grandes, de accesos abierto y ampliamente utilizados en el ámbito de las imágenes de rayos X. Contiene radiografías frontales del tórax y está compuesto por un total de 112,120 imágenes, cada una con una dimensión de 1024×1024. Estas imágenes provienen de 30,805 pacientes. El dataset abarca 14 tipos diferentes de enfermedades, como *Atelectasis*, *Consolidation*, *Infiltration*, *Pneumothorax*, *Edema*, *Emphysema*, *Fibrosis*, *Effusion*, *Pneumonia*, *Pleural-thickening*, *Cardiomegaly*, *Nodule Mass*, *Hernia*, además de una etiqueta adicional llamada *no finding*. La etiqueta *no finding* indica que no se ha detectado ninguna de las enfermedades mencionadas. Sin embargo, podría existir la posibilidad de que exista otras enfermedades no contempladas o en el etiquetado se haya presentado un grado de incertidumbre [25].

Dentro del conjunto, 51,708 imágenes están etiquetadas con una o más enfermedades, mientras que las restantes 60,412 imágenes están etiquetadas como "no finding". La distribución del número de imágenes por cada enfermedad se puede observar en la Tabla 4.1.

Enfermedad	# Imagenes
Atelectasis	11,535
Cardiomegaly	2,772
Effusion	13,307
Infiltration	19,871
Mass	5,746
Nodule	6,323
Pneumonia	1,353
Pneumothorax	5,298
Consolidation	4,667
Edema	2,303
Emphysema	2516
Fibrosis	1,686
Pleural_thickening	3,385
Hernia	227
No findings	60,412
Total	**112,120**

Tabla 4.1.: Numero de imágenes en el conjunto de datos:*NIH Chest X-rays*

4.1.1. Distribución del conjunto de datos

La distribución del conjunto de datos se compone de un conjunto oficial [25], el cual ha sido dividido en un conjunto de entrenamiento y otro de prueba. Las imágenes de los 30,805 pacientes han sido distribuidas de manera independiente en cada uno de estos conjuntos, asegurando que un paciente no aparezca en ambos.

Sin embargo, se ha decidido adoptar una estrategia similar a la utilizada en [21], en la cual se emplea el conjunto oficial pero se subdivide el conjunto de entrenamiento en dos partes: aproximadamente un 80 % se asigna al conjunto de entrenamiento y el restante 20 % se destina a la validación. En este enfoque, también se garantiza que los datos de un mismo paciente no se encuentren presentes en ambos subconjuntos.

4.2. Preprocesamiento y aumento de datos

En esta etapa, se adapta el tamaño de las imágenes mediante una interpolación bicubica, el cual reduce las imágenes a diferentes tamaños (224×224, 384×384 y 512×512), según las características de cada modelo.

Además de reducir el tamaño, se aplica *data augmentation* al conjunto de entrenamiento, el objetivo es incrementar la diversidad de los datos mediante transformaciones. Lo que produce, modelos mas robustos con una gran capacidad de generalización. Sin embargo,la elección de las transformaciones adecuadas junto con sus hiperparámetros respectivos, así como la búsqueda de la combinación óptima de tales transformaciones, resulta en un espacio de búsqueda muy amplio y costoso en términos computacionales.

En ese contexto, se introduce *RandAugment* [2], que automatiza este proceso y reduce el espacio de búsqueda drásticamente. *RandAugment* selecciona aleatoriamente un conjunto de transformaciones de una lista predefinida k y las aplica a una imagen con una determinada intensidad. El numero de transformaciones elegidas que se aplican a cada imagen esta determinado por el hiperparámetro N, mientras que la intensidad de cada transformación esta determinado por el hiperparametro M. De esta forma, se garantiza que cada transformación sea ocupada con una probabilidad uniforme de $\frac{1}{k}$.

La lista k, esta compuesto por las transformaciones descritas en la sección 3.3: *equalize*, *rotate*, *solarize*, *contrast*, *brightness*, *shear*, *translate* y *gaussian blur*. Luego, se configura $N = 2$ y $M = 5$. El algoritmo usando *Numpy* se muestra a continuación:

```
k = [
Equalize, Rotate, Solarize, Contrast, Brightness,
Shear, Translate, Gaussian blur]

def randaugment(N, M):
    """Generate a set of distortions.
    Args:
    N: Number of augmentation transformations to
    apply sequentially.
    M: Magnitude for all the transformations.
    """

    sampled_ops = np.random.choice(k, N)
    return [(op, M) for op in sampled_ops]
```

El flujo completo seguido para aumentar los datos de entrenamiento se visualiza en la Figura 4.1.

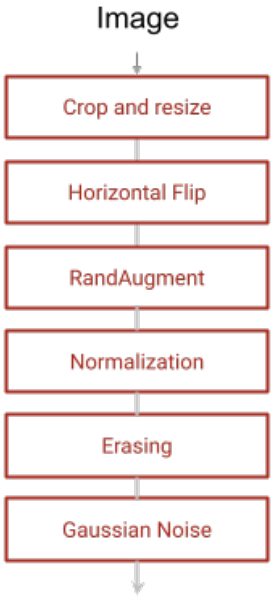


Figura 4.1.: Flujo seguido para el aumento de datos en el conjunto de entrenamiento

Las probabilidades de ocurrencia de cada transformación es de 0.5, con excepción de *RandAugment* y la normalización basada en la media y la desviación estándar de *ImageNet* [3], que siempre se realizan. Además, se incorporan las transformaciones *gaussian noise* y *erasing*. Estas últimas, a pesar de no preservar la distribución original del conjunto de datos, contribuyen a mejorar la capacidad de generalización de los modelos [2, 27].

El resultado de aplicar esta secuencia de transformaciones se observa en la Figura 4.2.

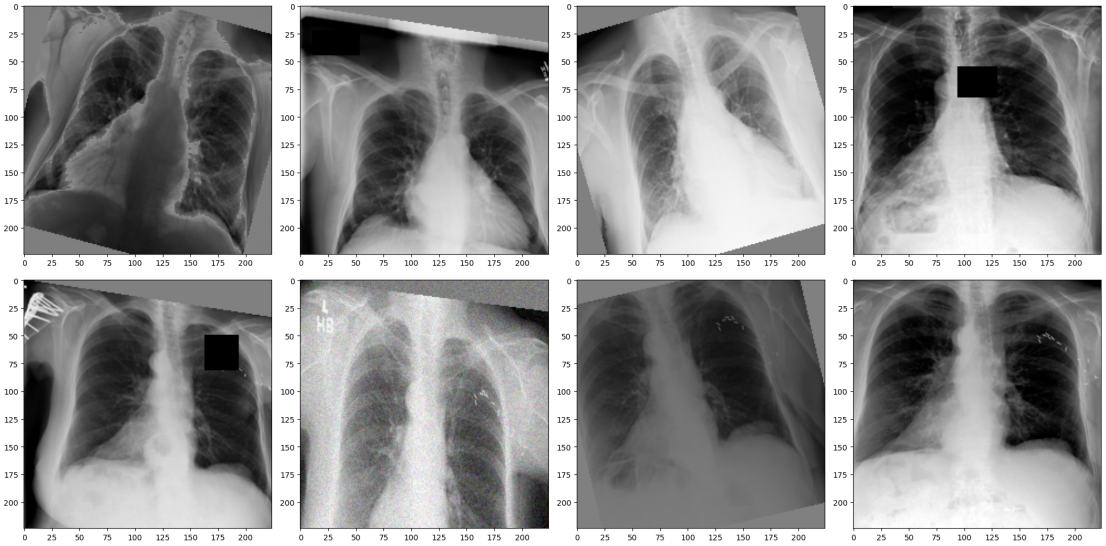


Figura 4.2.: Resultados sintéticos generados para el entrenamiento

4.3. Arquitecturas transformer y adaptación

En esta sección, se detallan las tres variantes de la arquitectura Transformer que se emplearán en la etapa de experimentación: *Vision Transformer*, *Swin Transformer* y *Multi-axis Vision Transformer*. Cada una de estas arquitecturas se divide en cuatro escalas, determinadas por su profundidad: *tiny*, *small*, *base* y *large*. Además, cada escala acepta imágenes de distintos tamaños: 224×224, 384×384 y 512×512 píxeles.

Para construir estas arquitecturas, se ha hecho uso de modelos preentrenados disponibles en la biblioteca de aprendizaje profundo *Timm* [26], los cuales fueron previamente entrenados en el conjunto de datos *ImageNet*. Para hacer una comparación justa entre las arquitecturas, se ha procurado que los pesos utilizados para inicializar las arquitecturas provengan de un mismo conjunto de datos de preentrenamiento (*ImageNet* 1k, 22k o una combinación de ambos). La relación de arquitecturas y su conjunto de preentrenamiento correspondiente se presenta en la Tabla 4.2.

Es importante señalar que, lamentablemente, no se pudieron encontrar modelos preentrenados para las arquitecturas *ViT* y *Swin* que sean compatibles con imágenes de dimensiones 512×512. Asimismo, para la arquitectura *Swin*, no se hallaron modelos preentrenados en las escalas *tiny* y *small* que admitan imágenes de tamaño 384×384. No obstante, para superar estas limitaciones, se ha adoptado una estrategia secuencial. Se toma un modelo de la misma escala con dimensión de entrada 224×224, que previamente había sido entrenado en el conjunto de datos *ChestX-ray14*, y se utilizá para el entrenamiento de modelos con entrada 384×384. Posteriormente, estos modelos adaptados con entrada de 384×384 se emplearan como punto de partida para el entrenamiento de los modelos con dimensiones de entrada de 512×512. Esta estrategia sigue el enfoque presentado en [13].

			ImageNet		
Model		Size	21k	1k	$21k \rightarrow 1k$
ViT	Tiny	224			x
		384		x	
		512		*	
	Small	224		x	
		384		x	
		512		*	
	Base	224			x
		384			x
		512			*
	Large	224			x
		384			x
		512			*
Swin	Tiny	224		x	
		384		*	
		512		*	
	Small	224		x	
		384		*	
		512		*	
	Base	224			x
		384			x
		512			*
	Large	224			x
		384			x
		512			*
MaxViT	Tiny	224		x	
		384		x	
		512		x	
	Small	224		x	
		384		x	
		512		x	
	Base	224	x		
		384			x
		512			x
	Large	224			x
		384			x
		512			x

Tabla 4.2.: Relación de arquitecturas evaluadas y el dataset utilizado para inicializar los pesos
$21k \rightarrow 1k$, indica que la arquitectura fue entrenado en Imagenet 21k y ajustado en Imagenet 1k. **x** indica la existencia de un modelo preentrenado en el dataset correspondiente. ***** indica que la arquitectura ha sido adaptado a partir de otra arquitectura con dimensión de entrada, inmediatamente inferior perteneciente a la misma escala.

En la Tabla 4.3, se presentan los hiperparámetros empleados en la configuración

de la arquitectura *ViT*, en sus diversas escalas.

Model	# Layers	Width	MLP	Heads	Patch-size
ViT-T	12	192	768	3	16
ViT-S	12	384	1536	6	16
ViT-B	12	768	3072	12	16
ViT-L	24	1024	4096	16	16

Tabla 4.3.: Hiperparámetros de configuración para Vision transformer

De igual forma, las Figuras 4.3 y 4.4, muestran los hiperparámetros de configuración para las arquitecturas *Swin* y *MaxViT*, respectivamente.

	downsp. rate (output size)	Swin-T	Swin-S	Swin-B	Swin-L
stage 1	4× (56×56)	concat 4×4, 96-d, LN	concat 4×4, 96-d, LN	concat 4×4, 128-d, LN	concat 4×4, 192-d, LN
		[win. sz. 7×7, dim 96, head 3] × 2	[win. sz. 7×7, dim 96, head 3] × 2	[win. sz. 7×7, dim 128, head 4] × 2	[win. sz. 7×7, dim 192, head 6] × 2
stage 2	8× (28×28)	concat 2×2, 192-d , LN	concat 2×2, 192-d , LN	concat 2×2, 256-d , LN	concat 2×2, 384-d , LN
		[win. sz. 7×7, dim 192, head 6] × 2	[win. sz. 7×7, dim 192, head 6] × 2	[win. sz. 7×7, dim 256, head 8] × 2	[win. sz. 7×7, dim 384, head 12] × 2
stage 3	16× (14×14)	concat 2×2, 384-d , LN	concat 2×2, 384-d , LN	concat 2×2, 512-d , LN	concat 2×2, 768-d , LN
		[win. sz. 7×7, dim 384, head 12] × 6	[win. sz. 7×7, dim 384, head 12] × 18	[win. sz. 7×7, dim 512, head 16] × 18	[win. sz. 7×7, dim 768, head 24] × 18
stage 4	32× (7×7)	concat 2×2, 768-d , LN	concat 2×2, 768-d , LN	concat 2×2, 1024-d , LN	concat 2×2, 1536-d , LN
		[win. sz. 7×7, dim 768, head 24] × 2	[win. sz. 7×7, dim 768, head 24] × 2	[win. sz. 7×7, dim 1024, head 32] × 2	[win. sz. 7×7, dim 1536, head 48] × 2

Figura 4.3.: Hiperparámetros de configuración para Swin transformer
Fuente: [13]

Stage	Size	MaxViT-T	MaxViT-S	MaxViT-B	MaxViT-L
S0: Conv-stem	1/2	B=2 C=64	B=2 C=64	B=2 C=64	B=2 C=128
S1: MaxViT-Block	1/4	B=2 C=64	B=2 C=96	B=2 C=96	B=2 C=128
S2: MaxViT-Block	1/8	B=2 C=128	B=2 C=192	B=6 C=192	B=6 C=256
S3: MaxViT-Block	1/16	B=5 C=256	B=5 C=384	B=14 C=384	B=14 C=512
S4: MaxViT-Block	1/32	B=2 C=512	B=2 C=768	B=2 C=768	B=2 C=1024

Figura 4.4.: Hiperparámetros de configuración para MaxViT transformer
Fuente: [23]

4.3.1. Adaptación

Para abordar la tarea de clasificación en el conjunto de datos *ChestX-ray14*, se realizó una adaptación a los modelos preentrenados. Primero, se elimino la capa de clasificación. Luego, siguiendo la metodología propuesta por [21], se incorporo un conjunto de perceptrones multicapa independientes para cada patología, compuestos por tres capas ocultas con 384, 48 y 48 neuronas respectivamente, todas con una función de activación *ReLU*.

La idea de este enfoque es que las arquitecturas base asuman el rol de arquitecturas troncales, que aprendan características comunes entre todas las patologías. Por

otro lado, el perceptrón multicapa asuma el rol de aprender las características especificas correspondientes a cada afección. Al final de cada perceptron multicapa se agrega una neurona de salida, con una función de activación sigmoide, lo que da como resultado un vector $t \in \mathbb{R}^{1\times 14}$.

4.4. Entrenamiento y validación

Para realizar el entrenamiento de los modelos se ha ocupado el código en pytorch, proporcionado por trabajos anteriores [21, 13] y luego se ha adaptado para nuestra necesidades especificas [1].

Dado que los modelos utilizados varían en escalas, dimensión de entrada y requieren considerables recursos computacionales, se ha aplicado diferentes técnicas que nos han permitido sobre llevar los problemas identificados como: falta de memoria en GPU, tiempo de entrenamiento y desbordamiento de gradiente.

Primero se explica las técnicas para abordar los problemas mencionados y posteriormente se detallara los hiperparámetros usados en el entrenamiento.

1. ***Gradient accumulation*:** Esta técnica se emplea para superar limitaciones de memoria y recursos computacionales al entrenar modelos con un lote de entrenamiento (*mini-batch*) que no caben completamente en la memoria. Esta estrategia implica dividir un lote de entrenamiento en varios sublotes más pequeños, que se procesan secuencialmente para calcular los gradientes. Estos gradientes se acumulan a lo largo del proceso y esta acumulación culmina cuando se procesa el último sublote. Luego, los valores acumulados de gradientes se utilizan para actualizar los pesos del modelo en un solo paso de actualización.

2. ***Precisión mixta*:** Esta técnica permite entrenar modelos empleando diferentes precisiones numéricas en cálculos específicos, sin comprometer los resultados o el desempeño de los modelos. Por ejemplo, se hace uso de una precisión numérica en 16 bits en operaciones realizadas por las funciones de activación, calculo de los gradientes y su almacenamiento, mientras que se emplean 32 bits para la actualización de los pesos del modelo [15]. Para implementar esta técnica, se utilizó la funcionalidad de *AUTOMATIC MIXED PRECISION* (*AMP*) proporcionado por PyTorch, la cual se encarga de gestionar de manera automática dichas operaciones.

3. ***Gradient clipping*:** Esta técnica evita que los gradientes se vuelvan demasiado grandes y provoquen problemas de convergencia. Para ello, si la norma de los gradientes supera un umbral definido, estas se normalizan para conseguir gradientes con un valor máximo al umbral.

Los hiperparámetros de entrenamiento han sido cuidadosamente seleccionados tras una exhaustiva revisión de la literatura [19, 13, 23, 21]. Estos valores óptimos se han empleado de manera uniforme en todas las arquitecturas entrenadas, asegurando de esta forma una comparación justa y objetiva entre ellas.

[1] Codigo: `https://github.com/vladiH/TransformerModelsChestXray`

El proceso de entrenamiento se llevó a cabo mediante el uso de AMP (*Automatic Mixed Precision*), empleando un *learning rate* de 3.1e-5 y un tamaño de lote (*batch size*) de 32. Para controlar la tasa de aprendizaje a lo largo del entrenamiento, se implementó un *cosine learning scheduler* con un período de calentamiento de 20 *epoch*. La duración total del entrenamiento se fijó en 300 *epoch*. El optimizador utilizado fue *AdamW*, con un coeficiente de decaimiento de pesos (*weight decay*) de 0.05. Además, se aplicó un límite (*gradient clipping*) de 5 para evitar problemas de explosión de gradientes. En los casos en que los modelos más grandes no admitían un lote (*batch*) de tamaño 32, se implementó la técnica de acumulación de gradientes (*gradient accumulation*). Dado que se trata de un problema de clasificación multietiqueta, la función de pérdida utilizada fue la *Binary Cross Entropy*. Con el fin de mitigar el riesgo de sobreajuste, se aplico *early stopping* en función del AUC del conjunto de validación. Finalmente, los modelos elegidos son aquellos con el AUC más altos dentro del conjunto de validación.

Parte V.

Experimentos y Resultados

5. Experimentación y Resultados

En este capítulo, se detallan los experimentos realizados para determinar los hiperparámetros previamente mencionados en el capítulo anterior (Desarrollo del proyecto 4). Posteriormente, se emplearán estos hiperparámetros obtenidos para entrenar las tres arquitecturas y conseguir las siguientes métricas: el área bajo la curva ROC *AUC*, el número de operaciones de punto flotante (*FLOPs*), la velocidad de procesamiento (*throughput*), el número total de parámetros en cada modelo y el tiempo estimado para realizar una epoca de entrenamiento.

El uso de estas métricas nos proporcionarán una comprensión integral de cómo los modelos se desempeñan en términos de eficiencia y velocidad de cálculo y cómo se comparan con respecto al *AUC* obtenido.

5.1. Determinación de hiperparámetros de entrenamiento

Como ya se ha mencionado en las secciones 4.2 y 4.4 del capitulo anterior, un gran numero de hiperparámetros fue obtenido mediante una revisión exhaustiva de la literatura. No obstante, se realizaron experimentos empleando la arquitectura *MaxViT-tiny*, cuya dimensión de entrada es de 224×224, para determinar los siguientes hiperparámetros: tamaño de lote (*batch size*), nivel de intensidad de transformación M en el aumento de datos y el valor umbral para el proceso de *clipping*. Este ultimo se ha considerado ya que en los códigos proporcionados se estable como 5 y en los documentos expuestos se describen como 1.

Model	**Learning rate**	**Epoch**	**M**	**Batch**	**Clipping**	**Validation AUC**
MaxViT Tiny	3.1e-5	30	5	16	1	82.1
			5	32	5	**82.3**
			7	16	1	82.0
			7	32	5	82.1
			9	16	1	81.9
			9	32	5	81.7

Tabla 5.1.: Determinación de hiperparámetros

En la Tabla 5.1, se presentan los resultados de los experimentos realizados para determinar los valores óptimos de los hiperparámetros: *M*, *batch size* y umbral de *clipping*, utilizando el conjunto de validación, lo cual ha conseguido un promedio de 82.3 en *AUC*. Es necesario mencionar, que no se observa demasiada diferencia

entre cada experimento realizado. Esto implicaría lanzar mas pruebas de cada experimento para determinar el mas óptimo, pero esto escapa fuera de los objetivos de este trabajo.

A partir de este punto, estos valores, en conjunto con los hiperparámetros previamente mencionados en la sección de entrenamiento 4.4 del capítulo anterior, serán empleados en el proceso de entrenamiento de todas las arquitecturas.

5.2. Experimentos en clasificación y resultados

En esta sección, se presentan los experimentos realizados utilizando las arquitecturas *Vision Transformer*, *Swin Transformer* y *MaxViT Transformer* para llevar a cabo la clasificación de patologías en el conjunto de datos *ChestX-ray14*. Cada una de estas arquitecturas ha sido entrenada utilizando los hiperparámetros detallados en las sección 4.4 del capitulo anterior.

Es relevante señalar que los resultados obtenidos en el conjunto de test provienen de los modelos que lograron el mejor AUC promedio durante una época específica de entrenamiento, determinado en el conjunto de validación. A continuación, se presentan los resultados de cada experimento.

5.2.1. Resultados para Vision transformer

Los resultados de las pruebas realizadas con la arquitectura *Vision Transformer* se presentan en la Tabla 5.2. Esta tabla muestra las estadísticas obtenidas en el conjunto de test para la arquitectura *Vision Transformer*, evaluado en diferentes escalas y configuraciones.

Se observa que se logran resultados óptimos al utilizar imágenes de entrada con una resolución de 384×384. Por otro lado, las imágenes con una resolución de 512×512 arrojan resultados incluso inferiores a las de 224×224. Además, se observa que los modelos de escalas superiores para la misma dimensión de entrada tienden a mostrar un AUC promedio superior.

Test Mean AUC vision transformer												
Patologias	**tiny**			**small**			**base**			**large**		
	224	**384**	**512**	**224**	**384**	**512**	**224**	**384**	**512**	**224**	**384**	**512**
Atelectasis	0.755	0.765	0.745	0.749	0.774	0.747	0.767	0.781	0.743	0.765	0.768	0.744
Cardiomegaly	0.842	0.843	0.827	0.859	0.875	0.830	0.864	0.875	0.826	0.869	0.876	0.815
Consolidation	0.741	0.746	0.726	0.737	0.749	0.721	0.734	0.748	0.714	0.749	0.757	0.720
Edema	0.828	0.835	0.839	0.827	0.830	0.839	0.829	0.831	0.845	0.825	0.844	0.847
Effusion	0.815	0.818	0.803	0.814	0.822	0.808	0.817	0.829	0.803	0.816	0.829	0.803
Emphysema	0.849	0.891	0.870	0.857	0.872	0.875	0.875	0.894	0.877	0.860	0.903	0.859
Fibrosis	0.760	0.821	0.778	0.769	0.809	0.801	0.768	0.809	0.786	0.780	0.815	0.811
Hernia	0.801	0.751	0.764	0.753	0.774	0.735	0.790	0.872	0.725	0.753	0.811	0.815
Infiltration	0.702	0.691	0.682	0.690	0.708	0.696	0.685	0.687	0.693	0.695	0.698	0.695
Mass	0.777	0.785	0.708	0.782	0.800	0.740	0.797	0.803	0.717	0.802	0.807	0.717
Nodule	0.720	0.737	0.710	0.737	0.737	0.697	0.723	0.770	0.694	0.738	0.748	0.709
Pleural-Thickening	0.751	0.760	0.737	0.745	0.756	0.742	0.742	0.765	0.729	0.751	0.767	0.746
Pneumonia	0.702	0.702	0.696	0.677	0.711	0.700	0.695	0.700	0.710	0.679	0.723	0.699
Pneumothorax	0.825	0.838	0.854	0.837	0.867	0.862	0.843	0.853	0.861	0.856	0.867	0.848
Total	0.776	**0.785**	0.767	0.774	**0.792**	0.771	0.781	**0.801**	0.766	0.781	**0.801**	0.773

Tabla 5.2.: Estadísticas obtenidas mediante Vision transformer en el conjunto de test

5.2.2. Resultados para Swin transformer

De igual forma, en la Tabla 5.3 se observa las estadísticas obtenidas en el conjunto de test evaluados en diferentes escalas y configuraciones con la arquitectura *Swin transformer*.

En este caso, se nota una evolución jerárquica en el AUC, lo que significa que generalmente hay un aumento en el AUC cuando tanto la resolución como la escala de la arquitectura se incrementan.

Test Mean AUC swin transformer												
Patologias	**tiny**			**small**			**base**			**large**		
	224	**384**	**512**	**224**	**384**	**512**	**224**	**384**	**512**	**224**	**384**	**512**
Atelectasis	0.777	0.774	0.771	0.776	0.782	0.784	0.776	0.790	0.794	0.787	0.786	0.790
Cardiomegaly	0.840	0.860	0.866	0.867	0.876	0.879	0.856	0.882	0.877	0.872	0.881	0.876
Consolidation	0.736	0.749	0.752	0.754	0.755	0.752	0.754	0.751	0.758	0.760	0.755	0.758
Edema	0.829	0.837	0.839	0.850	0.854	0.850	0.816	0.833	0.850	0.841	0.854	0.853
Effusion	0.821	0.826	0.828	0.829	0.833	0.833	0.828	0.836	0.838	0.829	0.835	0.836
Emphysema	0.903	0.894	0.898	0.903	0.919	0.918	0.907	0.926	0.934	0.901	0.920	0.922
Fibrosis	0.802	0.795	0.792	0.809	0.808	0.812	0.809	0.824	0.830	0.797	0.817	0.834
Hernia	0.828	0.828	0.814	0.760	0.791	0.782	0.792	0.688	0.778	0.850	0.815	0.852
Infiltration	0.694	0.698	0.696	0.708	0.701	0.701	0.703	0.709	0.709	0.713	0.713	0.702
Mass	0.804	0.796	0.791	0.818	0.801	0.811	0.815	0.837	0.843	0.815	0.825	0.834
Nodule	0.768	0.754	0.753	0.768	0.752	0.758	0.756	0.776	0.792	0.767	0.784	0.789
Pleural-Thickening	0.752	0.752	0.756	0.771	0.775	0.776	0.774	0.789	0.789	0.776	0.774	0.781
Pneumonia	0.691	0.701	0.685	0.717	0.728	0.721	0.701	0.718	0.729	0.713	0.715	0.716
Pneumothorax	0.852	0.849	0.845	0.871	0.860	0.860	0.866	0.880	0.883	0.860	0.881	0.877
Total	0.793	**0.794**	0.792	0.800	**0.803**	**0.803**	0.797	0.803	**0.815**	0.806	0.811	**0.816**

Tabla 5.3.: Estadísticas obtenidas mediante Swin transformer en el conjunto de test

5.2.3. Resultados para MaxViT transformer

En la Tabla 5.4, se presentan los resultados obtenidos al evaluar la arquitectura *MaxViT* en diferentes escalas y configuraciones de entrada. Al igual que en *Swin*, se aprecia un patrón jerárquico consistente, a excepción de la escala *base*, donde se registra un descenso en el AUC en las diferentes resoluciones.

Test Mean AUC maxvit transformer												
Patologias	**tiny**			**small**			**base**			**large**		
	224	**384**	**512**	**224**	**384**	**512**	**224**	**384**	**512**	**224**	**384**	**512**
Atelectasis	0.777	0.775	0.782	0.769	0.785	0.783	0.764	0.777	0.783	0.770	0.775	0.790
Cardiomegaly	0.857	0.853	0.843	0.862	0.848	0.854	0.854	0.859	0.841	0.872	0.858	0.863
Consolidation	0.742	0.748	0.746	0.738	0.748	0.747	0.744	0.747	0.748	0.745	0.749	0.758
Edema	0.833	0.814	0.839	0.848	0.854	0.854	0.829	0.844	0.848	0.842	0.842	0.844
Effusion	0.824	0.827	0.833	0.826	0.827	0.833	0.827	0.833	0.826	0.825	0.831	0.836
Emphysema	0.873	0.911	0.921	0.890	0.908	0.925	0.890	0.896	0.929	0.888	0.914	0.922
Fibrosis	0.814	0.808	0.821	0.798	0.789	0.819	0.787	0.812	0.823	0.816	0.816	0.834
Hernia	0.779	0.754	0.762	0.797	0.778	0.692	0.744	0.667	0.720	0.811	0.756	0.766
Infiltration	0.704	0.705	0.703	0.699	0.714	0.709	0.698	0.707	0.698	0.701	0.708	0.713
Mass	0.794	0.807	0.820	0.803	0.809	0.825	0.799	0.809	0.815	0.802	0.819	0.834
Nodule	0.750	0.767	0.793	0.769	0.784	0.787	0.756	0.784	0.793	0.758	0.782	0.789
Pleural-Thickening	0.758	0.774	0.776	0.763	0.776	0.785	0.764	0.765	0.785	0.751	0.778	0.781
Pneumonia	0.708	0.715	0.727	0.720	0.721	0.739	0.706	0.723	0.715	0.715	0.722	0.730
Pneumothorax	0.849	0.863	0.872	0.858	0.875	0.878	0.856	0.866	0.880	0.851	0.865	0.879
Total	0.790	0.794	**0.803**	0.796	0.801	**0.802**	0.787	0.792	**0.800**	0.796	0.801	**0.810**

Tabla 5.4.: Estadísticas obtenidas mediante MaxViT transformer en el conjunto de test

5.2.4. Resumen de resultados obtenidos en el conjunto de test

La tabla 5.5 muestra el resumen y una comparación de los resultados obtenidos en términos de AUC en la clasificación de 14 patologías por las diferentes arquitecturas evaluadas en este estudio. Primero, se observa que en *ViT*, la resolución óptima es 384, llegando a ser superior incluso a *MaxViT* en las escalas *base* y *large*. En *Swin* y *MaxViT*, la resolución óptima es 512. Segundo, se observa que *Swin* es superior al resto de arquitecturas en toda sus variantes y resoluciones de entrada, seguido por *MaxViT*, alcanzando un *AUC* promedio máximo de 81.6 % en la escala *large*, con una resolución de 512. Finalmente, en general la resolución de 384×384 podría ser considerado como un óptimo dentro de toda las arquitecturas evaluadas.

Resumen Test Mean AUC												
Model	**Tiny**			**Small**			**Base**			**Large**		
	224	**384**	**512**	**224**	**384**	**512**	**224**	**384**	**512**	**224**	**384**	**512**
ViT	77.6	**78.5**	76.7	77.4	**79.2**	77.1	78.1	**80.1**	76.6	78.1	**80.1**	77.3
Swin	79.3	**79.4**	79.2	80.0	**80.3**	**80.3**	79.7	80.3	**81.5**	80.6	81.1	**81.6**
MaxViT	79.0	79.4	**80.3**	79.6	80.1	**80.2**	78.7	79.2	**80.0**	79.6	80.1	**81.0**
Total	78.6	**79.1**	78.7	79.0	**79.9**	79.2	78.8	**79.9**	79.4	79.4	**80.4**	80.0

Tabla 5.5.: Resumen de AUC promedio obtenidos en el conjunto de test
Los valores en negrita representan el AUC promedio mas alto dentro de una escala agrupados por tipo de arquitectura. La linea bajo un numero indica el maximo AUC promedio obtenido mediante una resolución comparados entre cada arquitectura (mayor valor en el eje vertical).

5.3. Eficiencia y rendimiento de las arquitecturas entrenadas

En esta sección, se llevará a cabo un análisis comparativo de las arquitecturas entrenadas, centrándonos en la eficiencia y la velocidad de cálculo para evaluar su rendimiento.

Primero, mediremos cómo se desempeña cada arquitectura en relación con la cantidad de parámetros que utiliza. Esto nos dará una idea de cuánta capacidad de aprendizaje tiene cada modelo. Luego, analizaremos el rendimiento computacional de las arquitecturas, medido en términos del número de operaciones de punto flotante que realizan. Esto nos brindará información sobre la intensidad computacional de cada arquitectura en función de su rendimiento. Finalmente, examinaremos su rendimiento en función del número de imágenes procesadas por segundo. Esto nos brindara información de la cantidad de trabajo realizado por el modelo en términos de imágenes por segundo, lo cual permitirá determinar las capacidades de la arquitectura en entornos de alto rendimiento.

Los resultados de estos análisis nos permitirán obtener una comprensión mas profunda de cada arquitectura y al mismo tiempo permitirá tomar decisiones mas acertadas para su aplicabilidad en la clasificación de imágenes médicas del mundo

real.

5.3.1. Rendimiento en función de los Parámetros

La Figura 5.1 presenta una comparativa entre el rendimiento y el numero de parámetros de las arquitecturas. Los puntos más a la izquierda representan las arquitecturas de escala *tiny*, mientras que aquellos más hacia la derecha reflejan las escalas *large*, en diversas configuraciones según el tamaño de imagen.

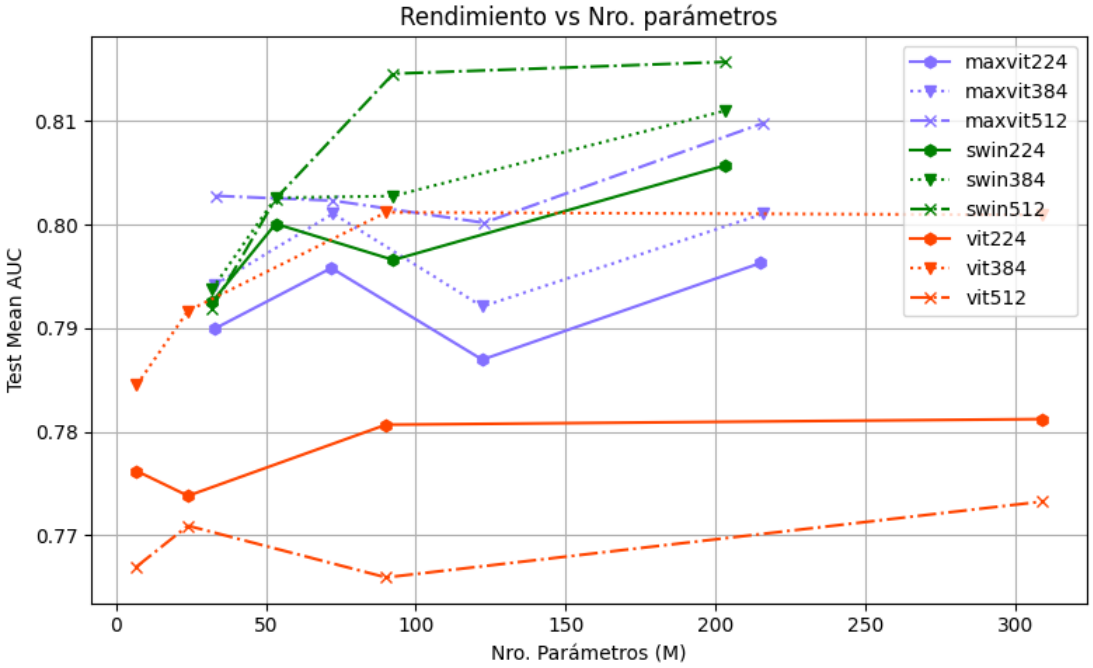


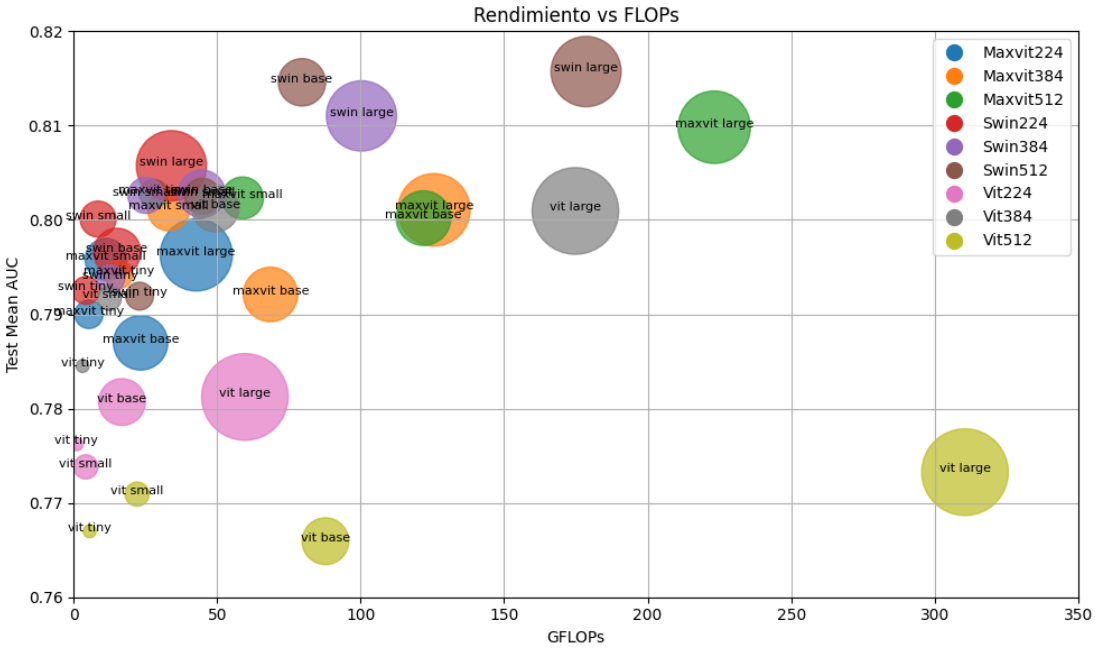
Figura 5.1.: Rendimiento en el conjunto de Test en función al número de parámetros

La gráfica detalla que, un aumento en la cantidad de parámetros no siempre se traduce en una mejora proporcional en el rendimiento. Además, se observa que las arquitecturas *ViT* en las escalas *tiny* y *small* tienen un menor número de parámetros, pero también presentan un rendimiento inferior. Sin embargo, a partir de la escala *base*, *ViT* logra superar al resto en términos de parámetros. Por otro lado, *Swin* generalmente requiere menos parámetros que *MaxViT* y al mismo tiempo, ofrece un mejor rendimiento.

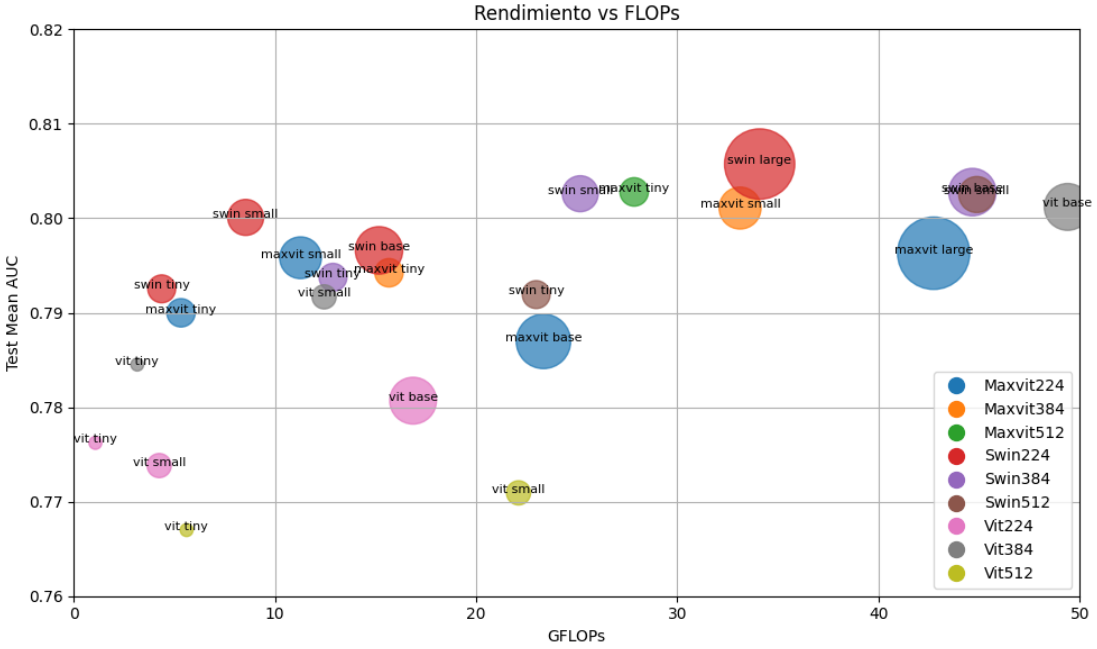
En general, *ViT* presenta un incremento mas abrupto en el numero de parámetros al variar de escala, seguido por *MaxVit* y *Swin*.

5.3.2. Rendimiento en función de los FLOPs

La Figura 5.2 presenta una comparativa del rendimiento en relación con el número de operaciones de punto flotante realizadas por cada arquitectura en sus diferentes configuraciones. El tamaño de cada circulo representa una proporción con respecto al numero de parámetros que ocupa cada modelo.



(a)



(b)

Figura 5.2.: Rendimiento en el conjunto de Test en función al número de operaciones aritméticas realizadas
(a) Rendimiento vs FLOPs desde una perspectiva global. (b) Rendimiento vs FLOPs en el rango de 0 a 50 GFLOPs.

Se puede observar que la arquitectura más compleja es *ViT*, seguida por *MaxViT* y *Swin*, todas configuradas en la escala *large* y con una imagen de resolución de 512×512. Entre ellas, *Swin* muestra el mejor rendimiento, como se puede apreciar en la Figura 5.2a.

Posteriormente, se nota que la arquitectura *ViT* presenta una menor complejidad en comparación con las otras arquitecturas cuando se consideran las escalas *tiny* y *small*. Sin embargo, en la escala *base*, *ViT* se vuelve más compleja que *Swin*, aunque no tanto como *MaxViT*, como se ilustra en la Figura 5.2b.

A medida que avanzamos hacia la escala *large* con una resolución de entrada de 512×512, *ViT* se convierte en la arquitectura más compleja en comparación con todas las demás.

Finalmente, el aspecto mas resaltante que se evidencia en la Figura 5.2, es que la complejidad depende en mayor medida de la resolución entrada, mas no del numero de parámetros necesarios para ajustar el modelo a un conjunto de datos determinado.

En la Figura 5.3 se plasma la complejidad de las arquitecturas evaluadas en toda sus configuraciones, agrupadas por tipo de modelo y dimensión de entrada, desde otra perspectiva.

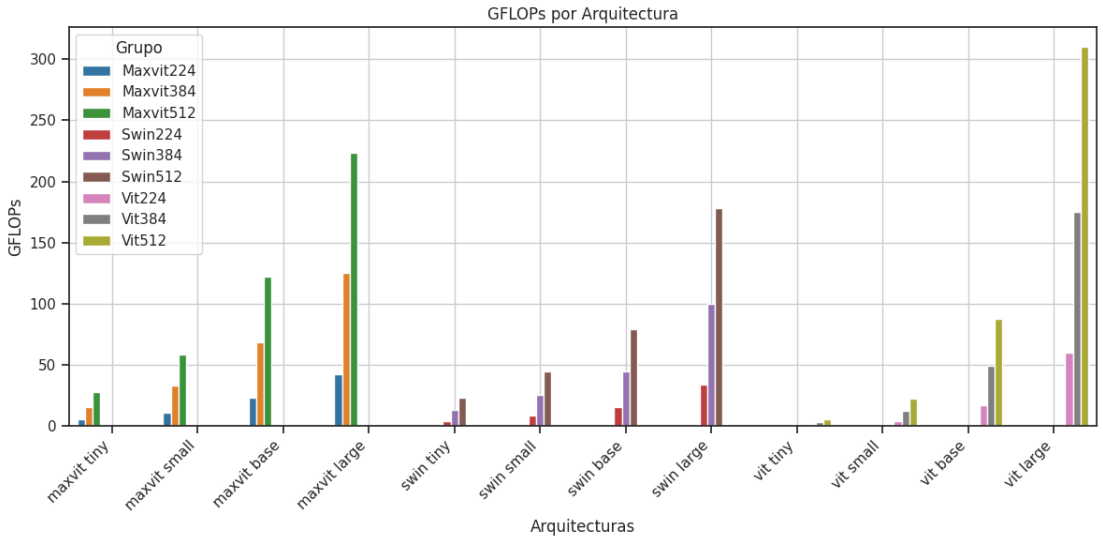


Figura 5.3.: Evolución de la complejidad computacional de las arquitecturas evaluadas

5.3.3. Rendimiento en Relación al Throughput

La Figura 5.4 exhibe una comparación del rendimiento en función de la cantidad de imágenes inferidas por segundo (*throughput*). El tamaño del circulo representa la complejidad del modelo.

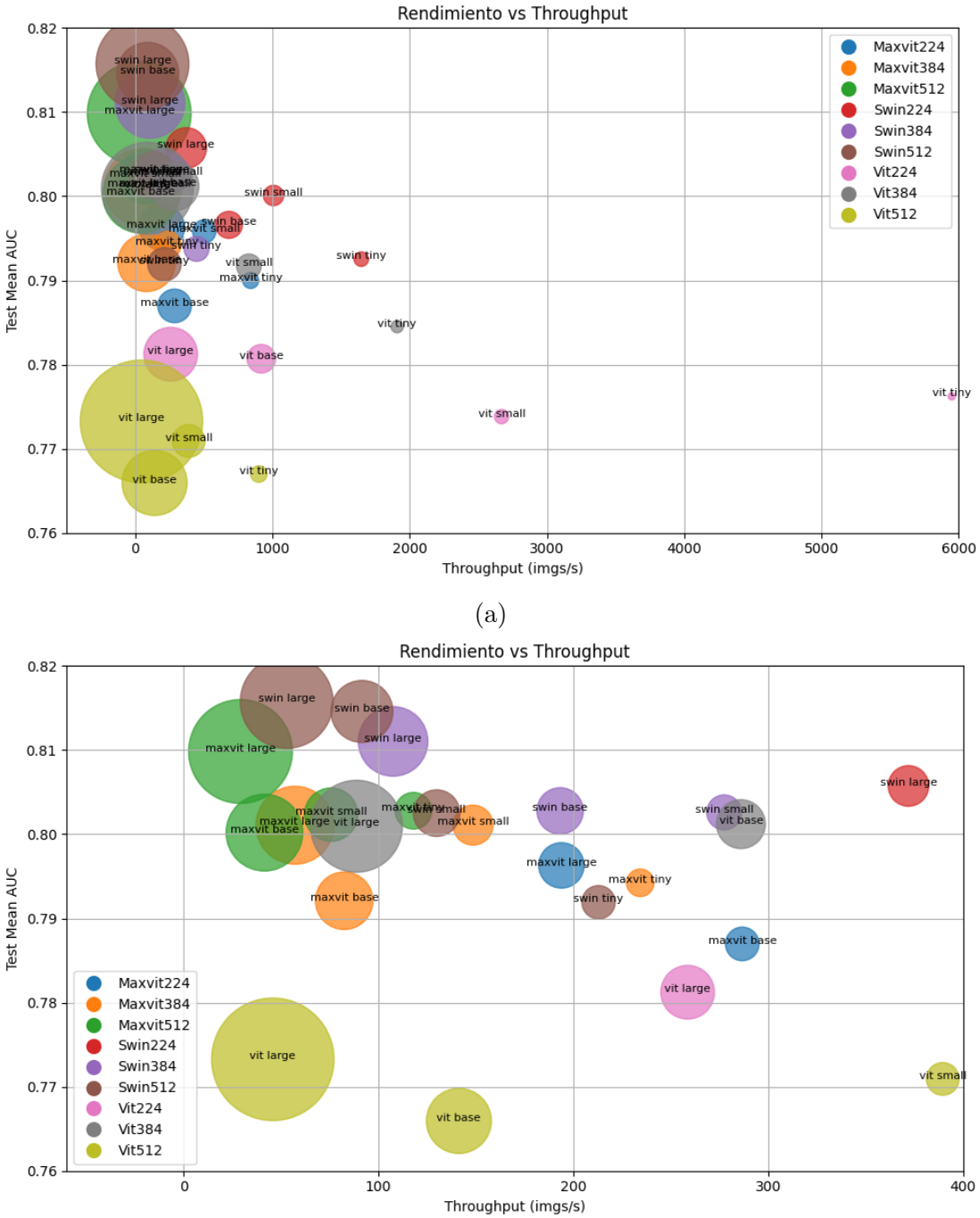


Figura 5.4.: Rendimiento en el conjunto de Test en función a la velocidad de inferencia
(a) Rendimiento vs Throughput desde una perspectiva global. (b) Rendimiento vs Throughput en el rango de 0 a 400 imgs/s.

El análisis muestra que, para imágenes de 224×224, la arquitectura *ViT* logra una inferencia significativamente más rápida en comparación con las demás, alcanzando aproximadamente 6000 imágenes por segundo para la escala *tiny*, llegando a ser superior en rendimiento a unos mas complejos de su misma familia. No obstante, esta velocidad tiende a disminuir con dimensiones de entrada más grandes

y escalas mayores.

Por otro lado, la arquitectura que presenta menor *trhoughput* es *MaxViT* en toda sus configuraciones.

Finalmente, la Figura 5.4b muestra casos particulares en los que algunas arquitecturas, a pesar de tener una mayor complejidad en términos de *FLOPs*, presentan una mayor capacidad de procesamiento en términos de imágenes por segundo y al mismo tiempo, un rendimiento superior. Esto sugiere que la eficiencia del modelo no está directamente relacionada con su complejidad computacional, ya que esta puede estar influenciada por diversos factores, como la existencia de hardware especializado para determinadas configuraciones o software altamente optimizado.

En este contexto, la arquitectura *Swin* destaca con el rendimiento más alto en todas sus variantes, llegando a ofrecer un equilibrio favorable entre velocidad, rendimiento y complejidad.

5.4. Resumen de resultados

La Tabla 5.6 resume los resultados de los experimentos realizados en todas las arquitecturas y sus diversas configuraciones. En particular, se presenta el *AUC* promedio (*Val AUC* y *Test AUC*) en los conjuntos de validación y pruebas, respectivamente. Además, se indican los parámetros en millones, los *FLOPs* en gigaflops y el tiempo necesario para completar una época de entrenamiento en minutos (*min*).

Las arquitecturas evaluadas demuestran un rendimiento notable en el conjunto de validación, alcanzando aproximadamente un *AUC* promedio del 85 % en el mejor de los casos. Sin embargo, su desempeño se reduce ligeramente en el conjunto de pruebas, donde el *AUC* promedio de la mejor arquitectura alcanza alrededor del 82 %

Resumen General								
Model		Size	VAUC	TAUC	Params(M)	GFLOPs	Img/s	min
ViT	Tiny	224	0.81	0.776	6.8	1.1	5948.8	1.6
		384	0.819	0.785	6.8	3.2	1908.4	2.1
		512	0.797	0.767	6.8	5.6	900.5	3.2
	Small	224	0.814	0.774	24.0	4.3	2669.3	1.6
		384	0.826	0.792	24.0	12.4	828.1	2.4
		512	0.801	0.771	24.0	22.1	389.5	3.9
	Base	224	0.817	0.781	90.1	16.9	918.7	2.0
		384	0.828	0.801	90.1	49.4	286.0	5.2
		512	0.801	0.766	90.1	87.7	141.3	9.4
	Large	224	0.818	0.781	308.9	59.7	258.6	5.9
		384	0.822	0.801	308.9	174.8	88.7	15.6
		512	0.804	0.773	308.9	310.6	45.8	29.2
Swin	Tiny	224	0.826	0.793	31.9	4.4	1647.1	1.9
		384	0.821	0.794	31.9	12.9	446.7	4.9
		512	0.822	0.792	31.9	23.0	212.9	11.2
	Small	224	0.833	0.8	53.2	8.5	1007.8	2.3
		384	0.83	0.803	53.2	25.2	277.3	7.6
		512	0.833	0.803	53.2	44.9	129.8	17.5
	Base	224	0.831	0.797	92.5	15.2	682.8	2.9
		384	0.837	0.803	92.5	44.7	193.2	10.5
		512	**0.848**	0.815	92.5	79.6	91.5	23.8
	Large	224	0.834	0.806	203.5	34.1	371.9	5.0
		384	0.844	0.811	203.5	100.3	107.5	17.6
		512	0.844	**0.816**	203.5	178.5	52.9	38.5
MaxViT	Tiny	224	0.823	0.79	33.1	5.3	841.2	2.8
		384	0.836	0.794	33.1	15.7	234.3	8.3
		512	0.836	0.803	33.3	27.9	118.0	15.8
	Small	224	0.827	0.796	71.8	11.3	502.8	4.1
		384	0.832	0.801	72.4	33.1	148.5	12.2
		512	0.837	0.802	72.4	58.9	75.7	23.7
	Base	224	0.82	0.787	122.2	23.3	286.6	7.0
		384	0.832	0.792	122.2	68.6	82.4	20.9
		512	0.832	0.8	122.8	121.9	41.5	40.7
	Large	224	0.83	0.796	215.0	42.7	193.9	9.7
		384	0.836	0.801	216.1	125.6	57.1	29.0
		512	0.841	0.81	216.1	223.2	29.2	55.5

Tabla 5.6.: Resumen de estadísticas obtenidas mediante *ViT*, *Swin* y *MaxViT* en el dataset *ChestX-ray14*

Conclusiones

En este trabajo de fin de máster, se ha llevado a cabo un análisis exhaustivo del rendimiento de tres arquitecturas Transformer: *ViT*, *Swin* y *MaxViT*, en la tarea de clasificación de 14 patologías en radiografías de tórax utilizando el conjunto de datos médicos *NIH Chest X-rays*. A través de una serie de experimentos y evaluaciones, se han obtenido resultados significativos que permiten conocer las capacidades y limitaciones de estas arquitecturas en el contexto de la imagen médica.

Se pudo observar que, en general, todas las arquitecturas demostraron un rendimiento prometedor en la tarea de clasificación de patologías en radiografías de tórax, con *AUC* promedio de 84.8 % en el conjunto de validación obtenida por *Swin base*. Sin embargo, es importante destacar que el rendimiento en el conjunto de prueba fue ligeramente inferior, con un 81.6 %, logrado por la arquitectura *Swin large*, con una dimensión de entrada de 512. Esto sugiere la necesidad de realizar ajustes adicionales y mejoras en los modelos para lograr una generalización más sólida.

Al evaluar el rendimiento en función de diversos aspectos, como el número de parámetros, los *FLOPs* y el *throughput*, se revelaron patrones interesantes. Se observó que la arquitectura *ViT* tenía un rendimiento ligeramente inferior en comparación con las otras arquitecturas cuando se utilizaba la misma dimensión de entrada. Sin embargo, esta limitación podía superarse mediante su eficiencia para imágenes de resolución igual o menor a 384, lo que generaba resultados similares a los de *MaxViT*. Por otro lado, *MaxViT* en general demostró ser menos eficiente y obtener un rendimiento inferior en comparación con *Swin*. También se observó que la complejidad de las arquitecturas y su rendimiento estaban relacionados con las dimensiones de las imágenes de entrada. Finalmente, se destacó que un modelo más complejo no necesariamente era menos eficiente, ya que estas pueden ser superados mediante algoritmos y hardware mas optimizados.

En términos de escalabilidad y eficiencia, la arquitectura *Swin* demostró un rendimiento consistente en diferentes escalas y configuraciones, superando a las arquitecturas evaluadas en términos de *AUC* y manteniendo un equilibrio entre *FLOPs* y *throughput*. Sin embargo, es importante destacar que la elección de una depende en gran medida del problema al cual se quiere dar una solución.

En conclusión, este trabajo proporciona una visión integral de las capacidades y desafíos de las arquitecturas Transformer en la clasificación de imágenes médicas. Los resultados obtenidos ofrecen orientación valiosa para la elección de la arquitectura más adecuada en función de los requisitos de rendimiento y eficiencia en aplicaciones del mundo real.

Bibliografía

[1] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization.

[2] Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2019). Randaugment: Practical data augmentation with no separate search. *CoRR*, abs/1909.13719.

[3] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

[4] Desternes, J. (2021). Data augmentation for medical image analysis in deep learning. `https://www.imaios.com/en/resources/blog/ai-for-medical-imaging-data-augmentation`.

[5] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929.

[6] García Fenoll, I. (2010). Aportaciones a la Segmentación y Caracterización de Imágenes Médicas 3D.

[7] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

[8] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.

[9] Hu, J., Shen, L., and Sun, G. (2017). Squeeze-and-excitation networks. *CoRR*, abs/1709.01507.

[10] Huang, G., Liu, Z., and Weinberger, K. Q. (2016). Densely connected convolutional networks. *CoRR*, abs/1608.06993.

[11] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

[12] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

[13] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030.

[14] López., X. A. L., Soledispa., S. N. Z., Contreras., J. B. Y., Franco., E. H. Z., Armijos., R. B. O., Poveda., K. A. F., Zambrano., M. F. C., Subia., D. L. F., Gavilánez., W. E. V., and Mendoza., J. C. P. (2020). *Introducción al diagnóstico por imagen*. Médicina. Mawil Publicaciones de Ecuador.

[15] Micikevicius, P., Narang, S., Alben, J., Diamos, G. F., Elsen, E., García, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. (2017). Mixed precision training. *CoRR*, abs/1710.03740.

[16] Pandey, A. (2018). Depth-wise convolution and depth-wise separable convolution. Medium, Towards Data Science.

[17] Pytorch (2023). Illustration of transforms. Pytorch.

[18] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

[19] Steiner, A., Kolesnikov, A., , Zhai, X., Wightman, R., Uszkoreit, J., and Beyer, L. (2021). How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*.

[20] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. *CoRR*, abs/1409.4842.

[21] Taslimi, S., Taslimi, S., Fathi, N., Salehi, M., and Rohban, M. H. (2022). Swinchex: Multi-label classification on chest x-ray images with transformers.

[22] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2020). Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877.

[23] Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., and Li, Y. (2022). Maxvit: Multi-axis vision transformer.

[24] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

[25] Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. M. (2017). Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *CoRR*, abs/1705.02315.

[26] Wightman, R. (2019). Pytorch image models. `https://github.com/huggingface/pytorch-image-models`.

[27] Wightman, R., Touvron, H., and Jégou, H. (2021). Resnet strikes back: An improved training procedure in timm. *CoRR*, abs/2110.00476.