UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Detección automática de carcinoma ductal invasivo.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Giménez Devis, Cristobal

Tutor/a: Casacuberta Nolla, Francisco

CURSO ACADÉMICO: 2021/2022

Resumen

El carcinoma ductal invasivo (IDC) es uno de los tipos de cáncer de mama más comunes (aproximadamente el 80 %), identificar y clasificar adecuadamente cada variante es un trabajo crucial en el campo de la medicina, por tanto, una automatización eficiente y precisa de esta tarea puede contribuir a reducir recursos, tiempo y errores.

En este trabajo, se realizará una aproximación a este problema aplicando diversas técnicas y métodos de aprendizaje automático a partir de un conjunto de datos compuesto por imágenes a color etiquetadas como benignos o malignos

Palabras clave: Redes neuronales, Aprendizaje profundo, Clasificación, Cancer de mama

Abstract

Invasive ductal carcinoma (IDC) is one of the most common types of breast cancer (aproximately 80%), identifying and classifying correctly each variant is a crucial task in the field of medicine, so, an efficient and precise automatization of this task can contribute to reduce resources, time and errors.

In this study, an approach to this problem will be made using several techniques and methods of machine learning from a dataset made of color images labeled as benign or malignant.

Key words: Neural networks, Deep learning, Clasification, Breast cancer

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

A Paco, por aceptar un proyecto tan ambicioso y por guiarme en las decisiones más ambiguas. Ahora, he descubierto mi vocación.

A mi familia, que ha superado innumerables injusticias y dificultades, que me ha enseñado nada más que amor y apoyo, sois los mejores guías que podría haber pedido jamás.

A mis amigos, por acompañarme durante este arduo y divertido viaje que nuestros agrestes corazones nunca olvidarán. Aunque ahora nuestros caminos se separen, el tiempo nos juntará de nuevo.

A todos los que me vieron empezar pero no pudieron verme acabar. Cada noche que miro al cielo puedo ver vuestra estrella, allá donde estéis, nunca dejéis de brillar.

Y también a vosotros, los que me despreciasteis sin un ápice de escrupulosidad, mil veces me habéis derrotado, pero no me vencisteis ni una vez, observad, pues hoy, me alzo sobre vosotros.

«*Represento aquello que nunca has podido matar, no importa cuánto lo hayas intentado. Yo soy la esperanza.*» - *Kelsier, El Imperio Final.*

CAPÍTULO 1
Introducción

El carcinoma ductal invasivo es un tipo de cáncer de mama que comienza en las células que residen en el tejido mamario, y que se extiende al resto de tejidos cercanos, a partir de ahí, puede propagarse al cuerpo entero a través del torrente sanguíneo y el sistema linfático.

En este documento presentaremos un estudio de distintos modelos y técnicas con el objetivo de aplicar el conocimiento del campo del aprendizaje automático (*Machine Learning*) a un problema delicado como es la detección automática del carcinoma ductal invasivo.

1.1 Motivación

El cáncer es una de las principales causas de muerte en todo el mundo. Según un informe publicado por el IARC en 2018[1], se estima que el cáncer de mama tiene una incidencia mundial del 11,6 %, y una tasa de mortalidad del 15 % en mujeres. En donde, la variante principal de esta enfermedad es el carcinoma ductal invasivo. En la lucha contra el cáncer, un factor clave consiste en su detección en sus fases iniciales, lo que incrementa mucho la posibilidad de que se pueda curar a tiempo.

Los últimos avances tecnológicos, y sobre todo, en la rama del aprendizaje automático han demostrado que son capaces de resolver una gran cantidad de problemas de todo tipo, incluso en medicina. Una aplicación de estos modelos de forma correcta y eficiente puede contribuir a agilizar el proceso de detección temprana y salvar vidas.

1.2 Objetivos

El principal objetivo de este estudio consiste en: clasificar correctamente una imagen cualquiera de tejido como benigno o maligno, para esto, añadiremos los siguientes subobjetivos:

- Explorar distintos modelos de aprendizaje automático para la clasificación de imágenes.
- Realizar un preprocesado adecuado al conjunto de datos que se va a utilizar para que sean aptos para su uso.

[1] Informe disponible en: https://www.iarc.who.int/wp-content/uploads/2018/09/pr263$_E$.*pdf*

- Comparar los resultados y rendimiento de los modelos y estudiar el comportamiento y efectividad del mejor.

1.3 Estructura de la memoria

El presente documento se divide en un total de 6 capítulos, los cuales se distribuyen como sigue:

En el **capítulo 2** describiremos resumidamente, los procedimientos y resultados ya existentes de algunos autores respecto al problema a resolver.

En el **capítulo 3** hablaremos de la metodología usada. Los modelos de aprendizaje automático utilizados, su funcionamiento, propiedades y arquitectura.

En el **capítulo 4** expondremos el marco experimental. Se explicarán las métricas de evaluación utilizadas, procederemos a describir el conjunto de datos y los hiperparámetros que se han usado en los experimentos.

En el **capítulo 5** mostraremos y compararemos los resultados obtenidos tras entrenar los modelos explicados anteriormente.

En el **capítulo 6** concluiremos el trabajo y relacionaremos los expuesto en este estudio con conocimiento adquiridos durante el grado.

CAPÍTULO 2
Estado del arte

Como prólogo a nuestro estudio, comencemos exponiendo los resultados y métodos de numerosos autores que han tratado previamente de resolver el mismo problema.

En primer lugar, en [1] presentan un conjunto de datos de imágenes histopatológicas de tejido diagnosticado con carcinoma ductal invasivo. A continuación, realizan una partición de los datos en conjuntos de entrenamiento y test, posteriormente, prueban distintos modelos y comparan los resultados resumidos en un tabla.

De los resultados que obtuvieron, el mejor viene de una red neuronal convolucional (modelo que se explica en apartado 3.1.2), que obtuvo un Valor-F (F1 - Score) de 0.718 y un acierto equilibrado (BAC) de 0.8423 (todas estas métricas son explicadas en el apartado 4.1).

Un año más tarde, Janowczyk y Madabhushi [2], retoman el estudio y proponen el uso de la red AlexNet (descrita en el apartado 3.1.3) junto con algunas alteraciones a las imágenes. En el que consiguen mejorar ligeramente los resultados de [1], con un Valor-F (F1 - Score) de 0.7648 y un acierto equilibrado (BAC) de 0.8468.

Posteriormente, surgen dos estudios independientes. En el que el primero [3], propone 3 arquitecturas diferentes de una red neuronal convolucional y mide los resultados de cada una. El mejor resultado, fue un Valor-F de 0.91 y un acierto equilibrado de 0.87.

Por último, Saini y Susan [4] construyen un modelo llamado VGGIN-Net que entrenan sobre un conjunto de datos de varios tipos de cáncer de mama llamado *BreakHis* [5]. Posteriormente, aprovechan su red pre-entrenada para obtener resultados mediante *fine tuning* sobre el conjunto de datos propuesto por [1], lo cual resultó en un Valor-F (F1) de 0.91 y un acierto equilibrado (BAC) de 0.8678.

En general, todos estos estudios obtienen buenos resultados, sin embargo, existe un amplio margen de mejora, ya que ningún autor ha probado arquitecturas ya existentes (véase apartado 3.1.3) ni otros modelos, como los recientes *Transformers* (explicados en el apartado 3.2). Además, de que el conjunto de datos tiene un desequilibrado claro entre clases y contiene imágenes corruptas o poco útiles cuyo impacto negativo en la fase de entrenamiento no ha sido considerado en ningún estudio (véase apartado 4.2).

CAPÍTULO 3
Metodología

En el campo del aprendizaje automático existen muchos modelos y arquitecturas para resolver una gran cantidad de problemas distintos. Desde los primeros perceptrones multicapa, hasta los sofisticados *Transformers* (y muchos más) que hoy en día se usan para tareas como el procesamiento del lenguaje natural, detección de objetos, clasificación de vídeo e imágenes e incluso el complicado problema del flujo óptico.

En este trabajo, exploraremos distintos modelos que su pueden aplicar a la clasificación de imágenes como son: Redes neuronales convolucionales, y los *Transformers* (concretamente el ViT).

3.1 Redes neuronales artificiales

Las redes neuronales son un modelo computacional que surgen de la línea de investigación de la inteligencia artificial que intenta crear procesos que realicen tareas de forma inteligente.

3.1.1. Introducción

Una red neuronal, consiste en un conjunto de unidades llamadas neuronas conectadas entre sí.

El primer modelo de neurona moderno y que ha servido de inspiración para el desarrollo de muchos modelos, es la neurona artificial (véase la figura 3.1).

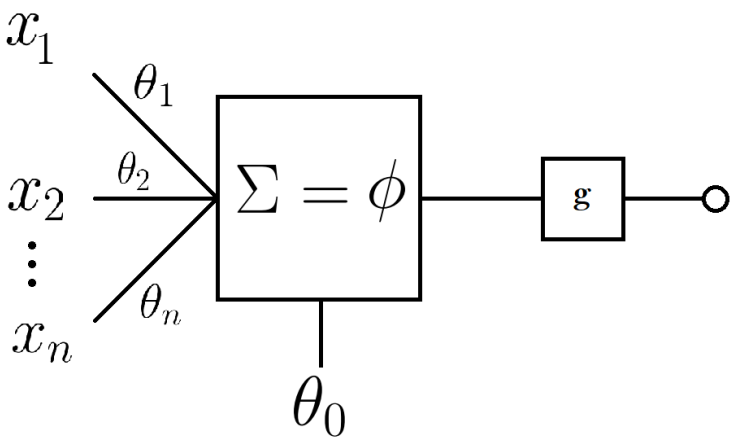


Figura 3.1: Modelo de una neurona artificial.

El modelo consiste en:

- Cada neurona tiene n entradas.
- Cada conexión con otra neurona tiene asociado un peso θ_i, $i > 0$.
- Cada neurona tiene un peso umbral θ_0.
- Una función de activación g.
- La salida de la neurona s.

De forma que la neurona realiza el cálculo:

$$s = g(\theta_0 + \sum_{i=1}^{n} \theta_i \cdot x_i)$$

Cuyo resultado será transmitido por la conexión de salida a la siguiente neurona. Este modelo (también llamado perceptrón simple), tiene la debilidad de que sólo puede resolver problemas linealmente separables, lo cual hace que no sea aplicable a la mayoría de problemas.

Perceptrón multicapa

Un perceptrón multicapa es una red neuronal artificial que consiste en un conjunto de neuronas (o perceptrones) distribuidos por capas conectadas entre sí (véase Figura 3.2):

- Capa de entrada: formada por la neuronas que introducen los datos de entrada a la red. No realizan ningún cálculo.
- Capa(s) oculta(s): constituidas por las neuronas que reciben sus datos de las capas directamente anteriores y cuyas salidas se propagan a neuronas de capas posteriores. En estas capas es donde se realiza la mayor parte del cálculo.
- Capa de salida: neuronas cuyos valores de salida representan la salida de toda la red.

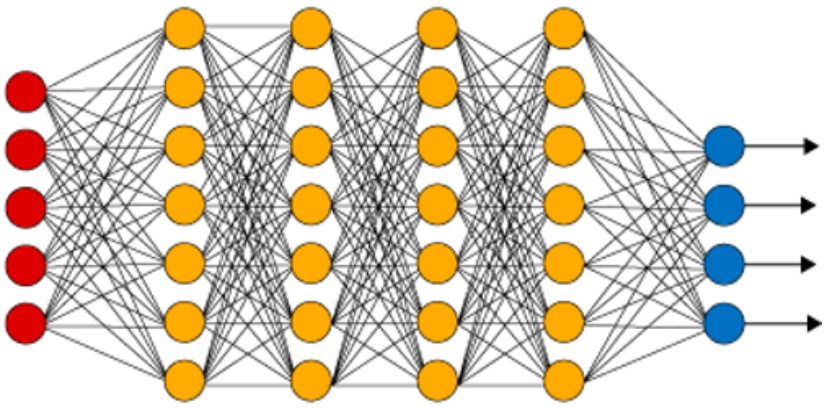


Figura 3.2: Esquema de un perceptrón multicapa. En rojo, la capa de entrada, en amarillo, las capas ocultas, y en azul, la capa de salida[1]

Este modelo, junto con las funciones de activación adecuadas, tiene la capacidad de resolver problemas que no son linealmente separables. Sin embargo estos modelos debían ser entrenados manualmente, ya que no existía un algoritmo que entrenara una red neuronal de forma automática. Hasta que finalmente, en 1986, Hinton, Rumelhart y Williams presentan el algoritmo de retropropagación (también llamado BackProp o propagación hasta atrás), descrito en la Figura 3.3, que permitió la automatización de la fase de entrenamiento de las redes neuronales.

Entrada: Una topología, datos de entrenamiento S, un factor de aprendizaje ρ, pesos iniciales θ_{ij}^l, $1 \leq l \leq L, 1 \leq i \leq M_l, 1 \leq j \leq M_{l-1}$, y condiciones de convergencia.
Salida: Pesos de las conexiones que minimizan el error cuadrático medio de S.

Mientras no se cumplan las condiciones de convergencia
Para $1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$, inicializar $\Delta\theta_{ij}^l = 0$
  Para toda muestra de aprendizaje (x,t) ϵ S
    Desde la capa de entrada hasta la capa de salida (l = 0,..., L)
      Para $1 \leq i \leq M_l$ si l = 0 entonces $s_i^0 = x_i$ si no calcular ϕ_i^l y $s_i^l = g(\phi_i^l)$
    Desde la salida hasta la entrada (l = L,..., 1)
      Para cada nodo $(1 \leq i \leq M_l)$

$$\text{Calcular } \delta_i^l = \begin{cases} \text{g'}(\phi_i^l)\ (t_{ni} - s_i^L) & \text{si } l == L \\ \text{g'}(\phi_i^l)\ (\sum_r \delta_r^{l+1}\theta_{ri}^{l+1}) & \text{en otro caso} \end{cases}$$

  Para cada peso $\theta_{ij}^l (0 \leq j \leq M_{l-1})$ calcular $\Delta\theta_{ij}^l = \Delta\theta_{ij}^l + \rho\ \delta_i^l s_j^{l-1}$
Para $1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$,
  Actualizar pesos $\theta_{ij}^l = \theta_{ij}^l + \frac{1}{N}\Delta\theta_{ij}^l$

Figura 3.3: Descripción del algoritmo de retropropagación (en pseudocódigo).

3.1.2. Redes neuronales convolucionales

Las redes neuronales convolucionales son una variante de las redes neuronales cuya creación se inspiró en el proceso que realiza el córtex visual del ojo para detectar y distinguir patrones [6].

Convoluciones

En el campo del aprendizaje automático, una convolución consiste en un producto escalar de una matriz de entrada (que puede ser una imagen, audio o texto) y una matriz llamada kernel o filtro.

Para obtener el resultado, el kernel actúa como una ventana deslizante que se mueve por la matriz de entrada de izquierda a derecha y de arriba a abajo en función de un parámetro llamado *stride*, que indica cuántas posiciones a la derecha debe desplazarse el kernel en cada momento.

A cada paso, el filtro realiza los productos escalares necesarios y suma los resultados en una matriz de salida (output).

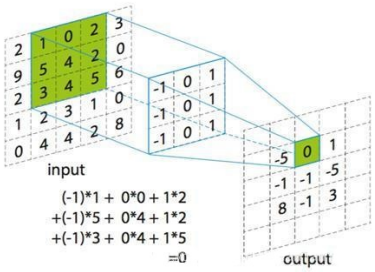


Figura 3.4: Ejemplo de una convolución de una matriz de 5x5 y un kernel de 3x3 (stride = 1).

Convoluciones en imágenes RGB

En este trabajo, usaremos imágenes a color, compuestas por 3 canales (rojo, verde y azul), lo que significa que para realizar correctamente una convolución debemos aplicar un kernel a cada canal de forma independiente, como se puede ver en la Figura 3.5

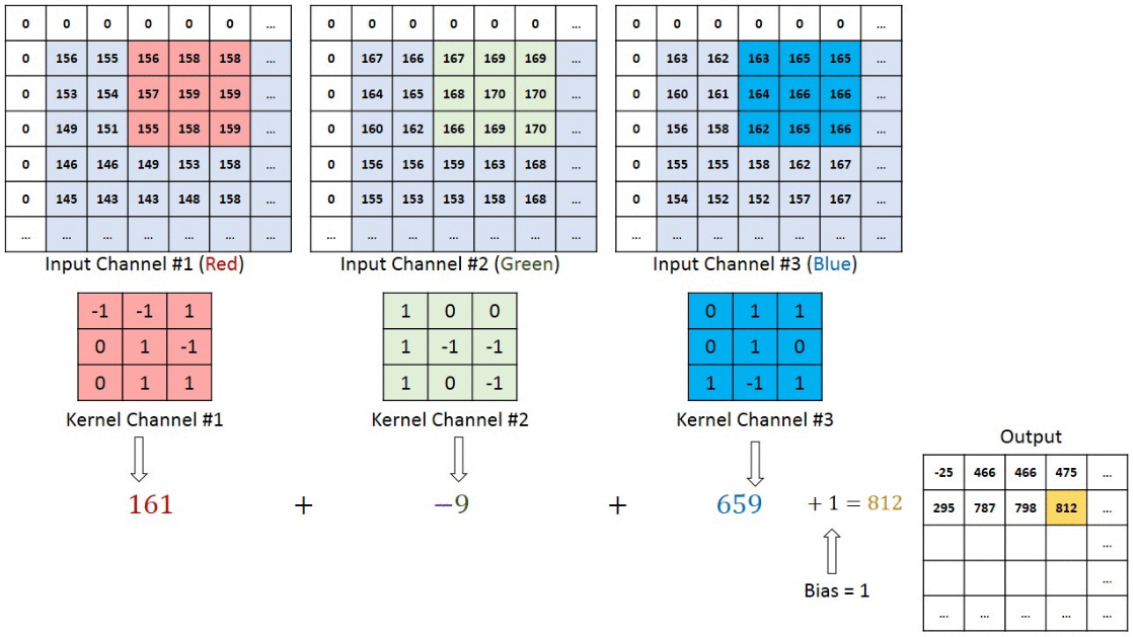


Figura 3.5: Ejemplo de una convolución de una imagen a color descompuesta en sus 3 canales de color (stride = 1).

Submuestreo o *sub-sampling*

Otro componente esencial en las redes convolucionales son las capas de submuestro (también llamadas *sub-sampling*), cuyo funcionamiento consiste en reducir el tamaño de los datos mientras avanzan por la capas de la red.

Estos componentes son muy útiles, ya que ignoran la posición original de las características que detectan los kernels, lo que permite que los modelos generalicen mejor sobre los datos, incluso los que han recibido alguna transformación afín.

Las capas de *sub-sampling* más conocidas son la Max-Pool y la Average-Pool. La primera consiste en tomar el valor máximo en la ventana, la segunda, calcula la media de los valores en la ventana deslizante (véase ejemplo en la figura 3.6).

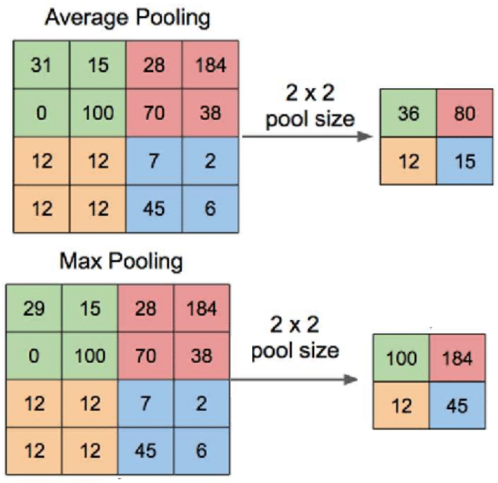


Figura 3.6: Ejemplo de funcionamiento de las capas Average-Pool y Max-Pool.

3.1.3. Modelos ya existentes

Durante las últimas décadas han surgido numerosas arquitecturas de redes convolucionales. Cada una de ellas ha sido propuesta para resolver un problema o para mejorar una arquitectura anterior a ella. Ahora, procederemos a explicar brevemente el contexto de cada modelo y su diseño.

LeNet5

La red LeNet5 fue una de las primeras redes convolucionales jamás creada. Este modelo fue propuesto y usado para identificar de forma automática dígitos manuscritos en blanco y negro [7].

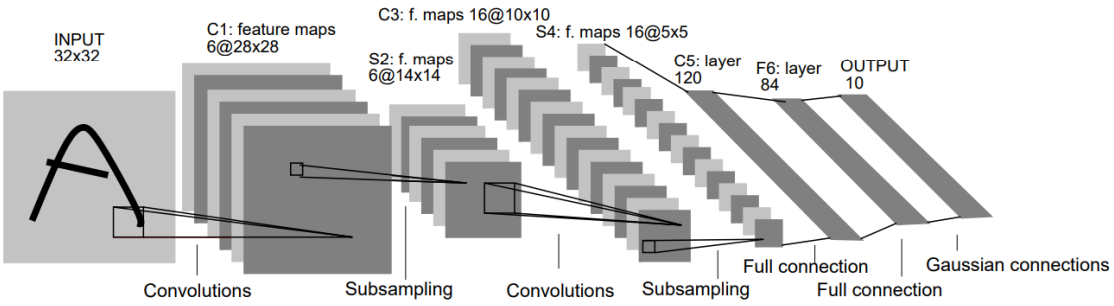


Figura 3.7: Arquitectura de la red LeNet5, extraida de [7].

La red tuvo un gran éxito, ya que se obtuvo una tasa de error de sólo el 1 %.

AlexNet

La red AlexNet [8] surgió como propuesta de solución en la competición llamada ILSVRC (*Image-Net Large Scale Visual Recognition Challenge*). El modelo alcanzó un tasa de error del 15,3 % y además demostró que la profundidad del modelo era necesaria para su correcto funcionamiento, que a pesar de que era computacionalmente costosa, era factible gracias a la aceleración por GPU.

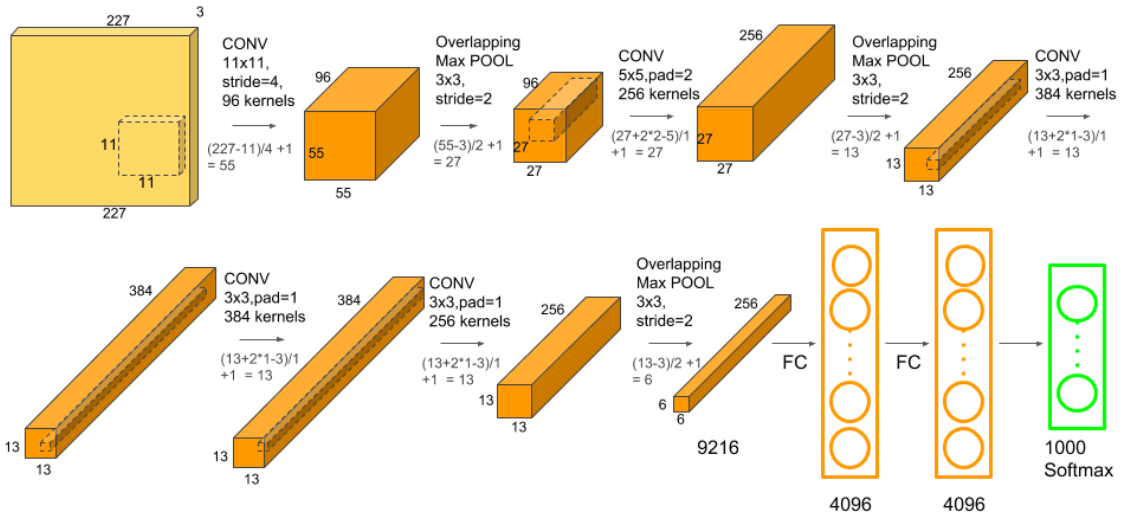


Figura 3.8: Ilustración de la red AlexNet.

VGG

Dos años más tarde, Simonyan y Zisserman [9] presentan un modelo de red convolucional en la ILSVRC del año 2014 con el objetivo de mejorar la tasa de acierto mediante el aumento del número de capas (profundidad) de la red, el modelo alcanzó el segundo puesto con una tasa de error del 7,3 %.

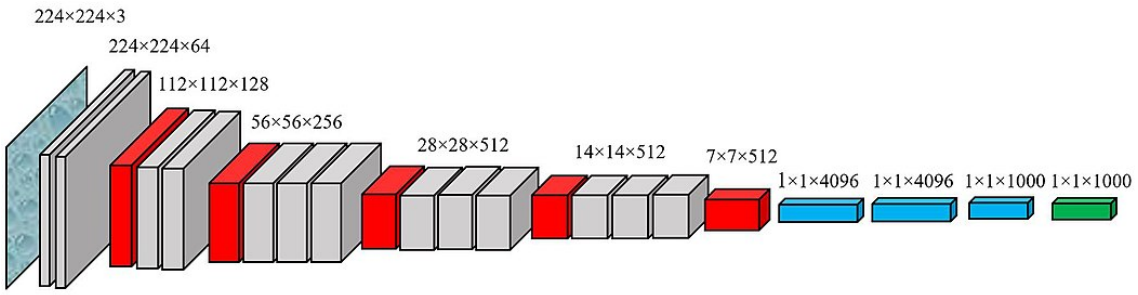


Figura 3.9: Ilustración de la red VGG.

Este modelo, pese a no vencer a la GoogLeNet (explicada en el apartado siguiente), ha sido aplicado éxitosamente a otros problemas en la medicina como la estimación de la frecuencia cardíaca en función del movimiento del cuerpo [10].

GoogLeNet

Al mismo tiempo, Google presenta su propio modelo: GoogLeNet [11] (también llamado InceptionNet), que utiliza resultados del artículo que introdujo el *Network in Network* [12].

Una de las características de este modelo es el uso de convoluciones con un kernel de 1x1. Lo cuál es extremadamente útil, ya que reduce la dimensionalidad de los datos de entrada, y por tanto, acelera la computación.

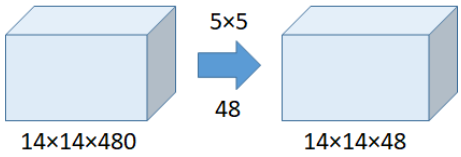


Figura 3.10: Ejemplo de una convolución de 5x5 sin reducción de dimensionalidad.

En este ejemplo (figura 3.10 [2]), el número de operaciones que se realizan es: $(14 \cdot 14 \cdot 48) \cdot (5 \cdot 5 \cdot 480) = 112{,}9$ millones de operaciones.

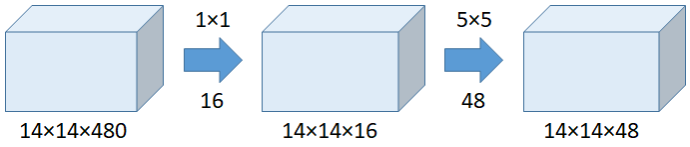
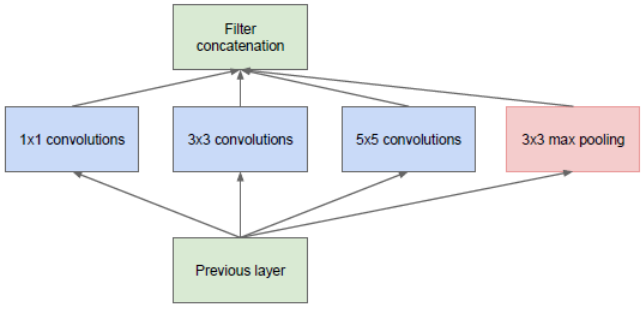


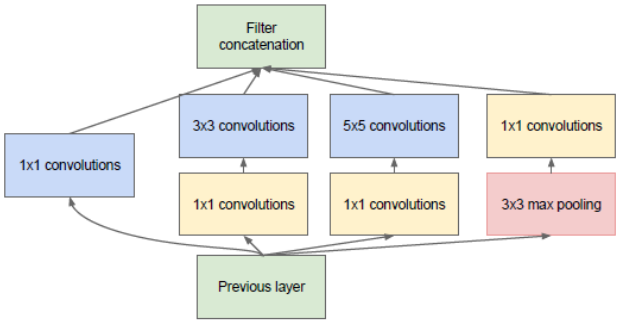
Figura 3.11: Ejemplo de una convolución de 5x5 con reducción de dimensionalidad mediante una convolución de 1x1.

Ahora, al aplicar una reducción a los datos, el número de operaciones es: $(14 \cdot 14 \cdot 16) \cdot (1 \cdot 1 \cdot 480) + (14 \cdot 14 \cdot 48) \cdot (5 \cdot 5 \cdot 16) = 1{,}5 + 3{,}8 = 5{,}3$ millones de operaciones.

Con esto, Google propone un modelo que se construye a base de *Inception modules* (véase figura 3.12), que consisten en la concatenación de varios kernels de tamaño distinto junto con una capa Max-Pool.



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

Figura 3.12: Ilustración de un *Inception module*, arriba, sin reducción de dimensionalidad [11]

[2]Imagen obtenida de: https://medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7

ResNet

Uno de los problemas de las redes neuronales profundas es que son difíciles de entrenar correctamente, ya que en la fase de entrenamiento, aparece un problema llamado desvanecimiento de gradiente. Este problema consiste en que mientras el gradiente se retropropaga a las capas anteriores, las sucesivas multiplicaciones que actualizan los pesos de la red hacen que el gradiente se reduzca, y que llegue a ser muy próximo a cero. Lo que hace las capas más cercanas a la entrada no se entrenen correctamente.

Han habido numerosos intentos de resolver este problema, pero finalmente, los autores de [13] introducen un modelo que resuelve esta dificultad mediante conexiones residuales (véase figura 3.13), que funcionan sumando la salida de la capa anterior con el resultado de la capa actual.

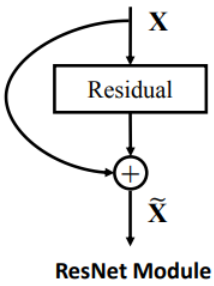


Figura 3.13: Funcionamiento de una conexión residual [13].

Con esta idea, se propone la arquitectura ResNet [13]. Que se compone por bloques que se comunican con los anteriores mediante conexiones residuales, que propagan el gradiente hacia atrás mejor.

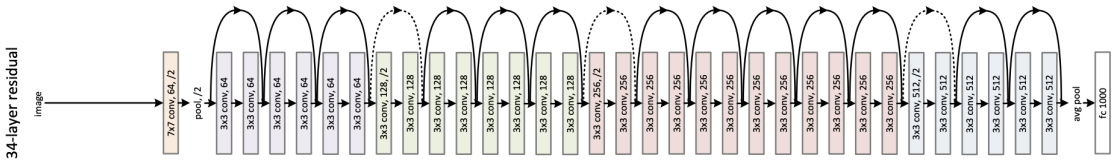


Figura 3.14: Arquitectura de la red ResNet [13]

.

EfficientNet

Otro de los desafíos a la hora de diseñar redes convolucionales es escalar el modelo. Es decir, cómo modificar o reajustar el modelo para que obtenga mejores resultados.

Probar manualmente configuraciones nuevas puede resultar tedioso e interminable, sin embargo, un artículo de 2019 [16] expone que un modelo cualquiera se puede escalar en:

- Profundidad (p): Que corresponde con el número de capas del modelo.
- Anchura (a): Que representa el número de neuronas en una sola capa.
- Resolución (r): Que hace referencia a el ancho y alto de la imagen de entrada.

Para realizar el escalado correctamente, se debe tomar una arquitectura base (cuyo diseño es crucial y no debe de ser un modelo general o poco desarrollado).

Seguidamente, ajustar tres parámetros: α, β y γ, que no pueden tener un valor cualquiera ya que están sujetos a dos restricciones.

$$p = \alpha^{\phi}$$
$$a = \beta^{\phi}$$
$$r = \gamma^{\phi}$$

Sujeto a:

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1,$$

Figura 3.15: Parámetros de escalado de un modelo y sus restricciones.

Una vez hecho, elevando estos parámetros a un entero ϕ se puede calcular la profundidad, la anchura y la resolución que debería tener el modelo escalado.

En el caso de [16], proponen una arquitectura base (véase figura 3.19) que posteriormente, ajustando los parámetros de escalado, transforman el modelo en versiones más grandes y que ofrecen mejores resultados.

El modelo base está compuesto por bloques que agrupan varios *mobile inverted bottlenecks* (abreviado MBConv) pero, con algunas modificaciones. Originalmente propuestos en [14], estos módulos calculan las convoluciones concatenando los resultados de cada kernel que se aplica a cada canal de entrada. Este nuevo tipo de convoluciones han sido llamadas *depthwise convolutions* [14] (véase una ilustración de este proceso en la figura 3.16 [3]).

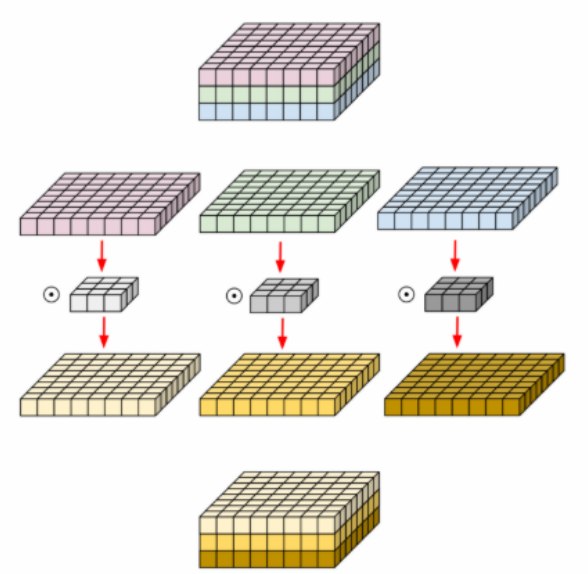


Figura 3.16: Ilustración del funcionamiento de una *depthwise convolution*

[3] Imagen extraída de: https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

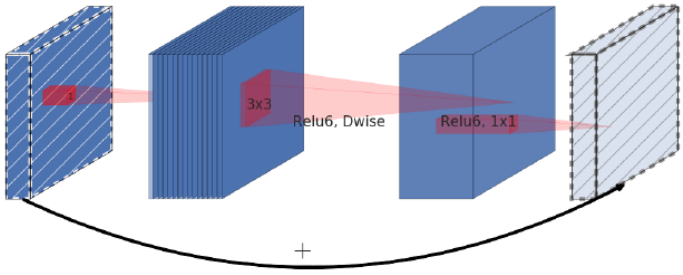


Figura 3.17: Esquema de un módulo MBConv (la flecha larga representa una conexión residual).

Además, a cada módulo MBConv, se le añade un mecanismo llamado *Squeeze and Excitation*, que recalibra cada canal en función de un parámetro que asigna un peso independiente a cada canal, para así, reforzar las características más importantes de la imagen y a su vez, ignorar las menos importantes. Obteniendo así una representación interna más robusta (véase un esquema en la figura 3.18). Propuesto originalmente en [15].

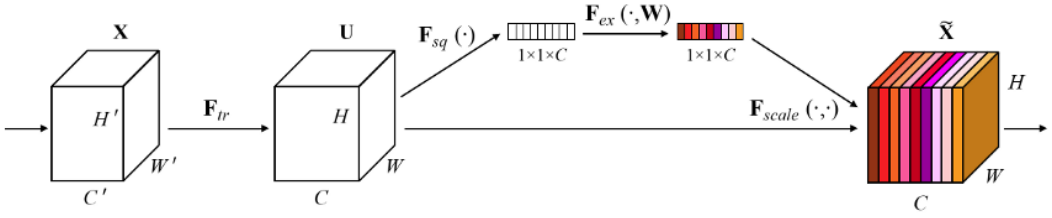


Figura 3.18: Ilustración del funcionamiento del *Squeeze and Excitation*, imagen extraída de [15].

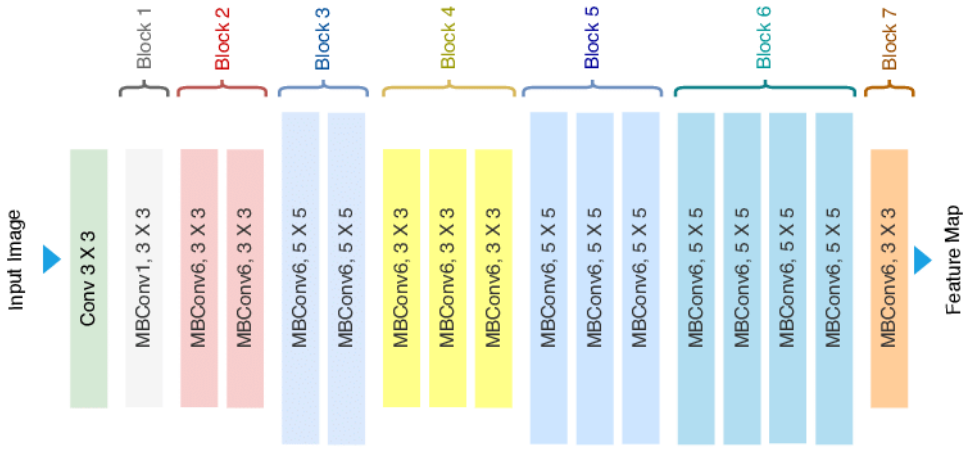


Figura 3.19: Arquitectura EfficientNetB0.

ConvNeXt

Finalmente, un artículo [17] en el que se realizó un estudio profundo sobre todas las redes convolucionales y sus diseños, identificó y eliminó los cuellos de botella en los modelos, y construyó una red que ofrece mejores resultados que todas las anteriores.

Los autores comienzan con la arquitecura ResNet50 (figura 3.14), cuyo diseño van modificando gradualmente según toman distintas decisiones sobre el modelo (figura 3.20).

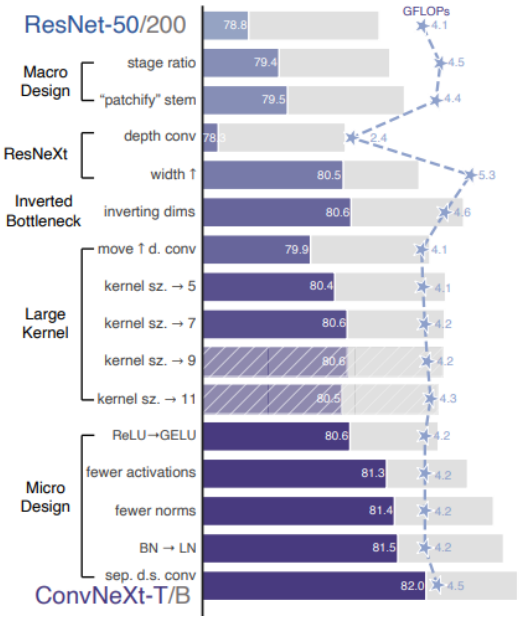


Figura 3.20: Cambios realizados a la arquitectura ResNet para diseñar la ConvNeXt. Imagen obtenida de [17].

Una de las primeras modificaciones fue reemplazar el optimizador del factor de aprendizaje Adam [23] por el optimizador AdamW [24], una versión más precisa y estable que su precursora.

De todos las modificaciones, el más importante consistió en rediseñar cada bloque de la ResNet, cuyos cambios consisten en convertir las funciones de activación ReLU [18] a la función GELU [19] y en cambiar el tamaño de los kernels (véase la figura 3.21 [4] para una comparación más gráfica).

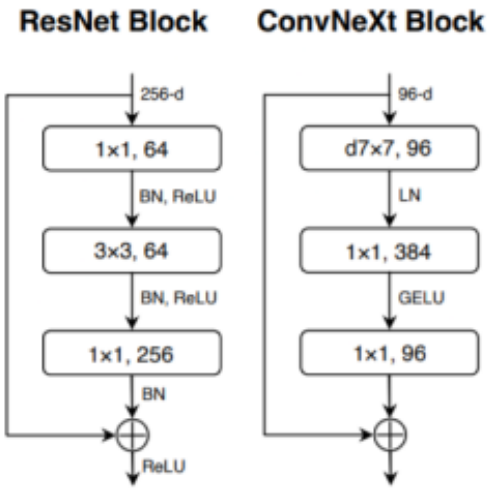


Figura 3.21: Comparación de un bloque ResNet respecto a uno ConvNeXt.

[4]Imagen obtenida de: https://sh-tsang.medium.com/review-convnext-a-convnet-for-the-2020s-53b9ada30ab9

3.2 Modelos de atención

En el campo del aprendizaje automático, y más concretamente, en la rama del procesamiento del lenguaje natural, las redes neuronales recurrentes (RNN), LSTM [25] y GRU [26] han sido modelos diseñados con el fin de resolver problemas como: el modelado de secuencia, es decir, predecir qué palabra viene después de otra. El modelado del lenguaje, que consiste en determinar la probabilidad de una secuencia de palabras y la traducción automática.

Todos estos modelos se basan en las posiciones de los símbolos de las secuencias de entrada y salida, por otro lado, en 2017, se propone una arquitectura completamente distinta, cuyo funcionamiento consiste en transformar una secuencia de entrada a una secuencia de salida (de ahí el nombre *Transformer* [27]).

3.2.1. Arquitectura *Transformer*

Un *Transformer* está compuesto por una pila de codificadores (*Encoders*) y una pila de decodificadores (*Decoders*).

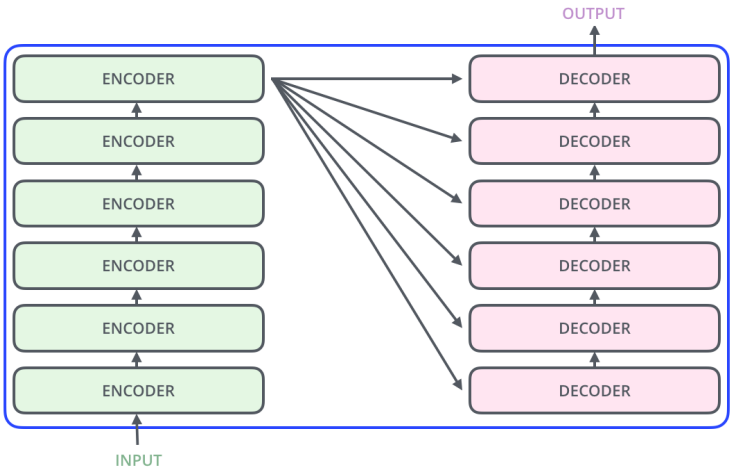


Figura 3.22: Esquema de la arquitectura *Transformer*

A su vez, cada codificador y decodificador tienen su propio diseño: el codificador consiste en una capa de atención por la que fluyen las secuencias de entrada, cuyas salidas son luego propagadas por un perceptrón multicapa (cada codificador tiene el mismo perceptrón colocado a la salida). Por otro lado, el decodificador sigue el mismo esquema, pero tiene en el centro una capa de atención adicional, que permite al decodificador centrarse en las partes importantes de la secuencia.

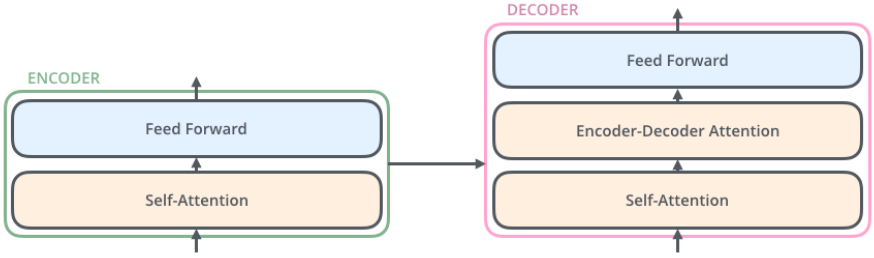


Figura 3.23: Esquema de un codificador y un decodificador.

3.2.2. Mecanismo de atención

La atención es un mecanismo que consiste en crear tres vectores a partir de los datos de entrada: un vector de consulta (*Query*, abreviado Q), un vector clave (*Key*, K) y un vector de valores (*Values*, V) y realizar la siguiente operación (fórmula extraída de [27]):

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Para calcular los vectores, internamente, existen tres matrices cuyos valores son parámetros entrenables con las que se realiza un producto matricial con las palabras de entrada (que suelen ser *embeddings*) para obtener dichos tres vectores (ilustración en la figura 3.24).

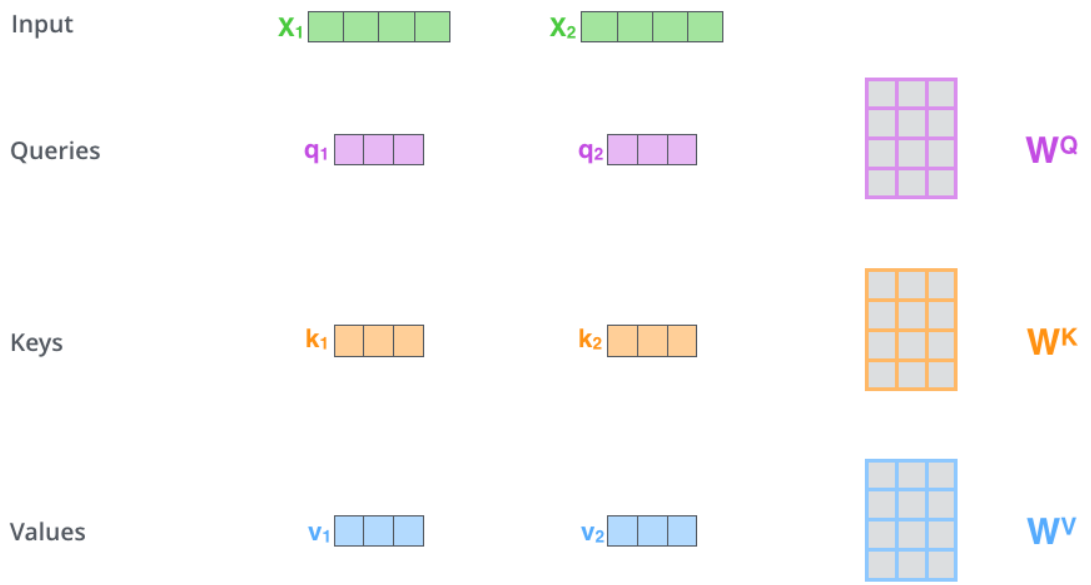


Figura 3.24: Mecanismo de atención (esquema).

De forma que se opera como sigue:

- $q_i = X_i \cdot W^Q$ para i ≥ 1
- $k_i = X_i \cdot W^K$ para i ≥ 1
- $v_i = X_i \cdot W^V$ para i ≥ 1

Posteriormente, se calcula una puntuación para cada palabra. Para hacer esto, se debe tomar el vector consulta de la palabra y realizar un producto escalar con los vectores claves de todas las palabras (incluso ella misma). Es decir, si tenemos p palabras, se debe hacer:

$$q_i \cdot k_j \text{ para } 1 \geq j \geq p.$$

A continuación se debe dividir los resultados de los productos escalares por la raíz cuadrada de la dimensión de los vectores (esto se hace para tener gradientes más estables). En el artículo original [27] la dimensión de los vectores de consulta,

clave y valores es 64, por tanto, se debe dividir todo por 8 y después, aplicar una función de activación softmax.

Por último, para calcular los vectores de salida, se realiza una suma de ponderada de los vectores de valor con los pesos que se han obtenido de la función softmax (puede verse un ejemplo en la figura 3.25 [5]).

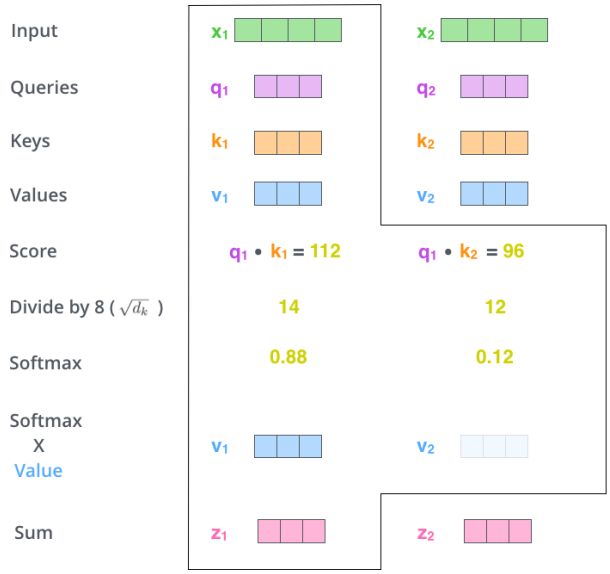


Figura 3.25: Ejemplo de funcionamiento de un mecanismo de atención. En este ejemplo, la salida sería $Z_1 = 0,88 \cdot V_1 + 0,12 \cdot V_2$

Con esto, la arquitectura codificador - decodificador quedaría como puede verse en la figura 3.26 [6]

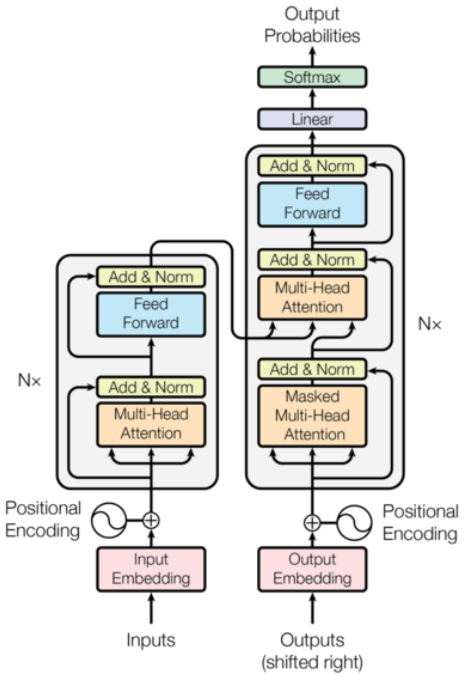


Figura 3.26: Arquitectura codificador - decodificador en detalle.

[5] Imagen extraída de: https://jalammar.github.io/illustrated-transformer/
[6] Imagen extraída de: https://nlp.seas.harvard.edu/2018/04/03/attention.html

3.2.3. *Vision Transformer* (ViT)

En contraposición a las redes convolucionales, en 2020, un artículo propone un modelo que usa *Transformers* para la clasificación de imágenes, el *Vision Transformer* (abreviado ViT) [28].

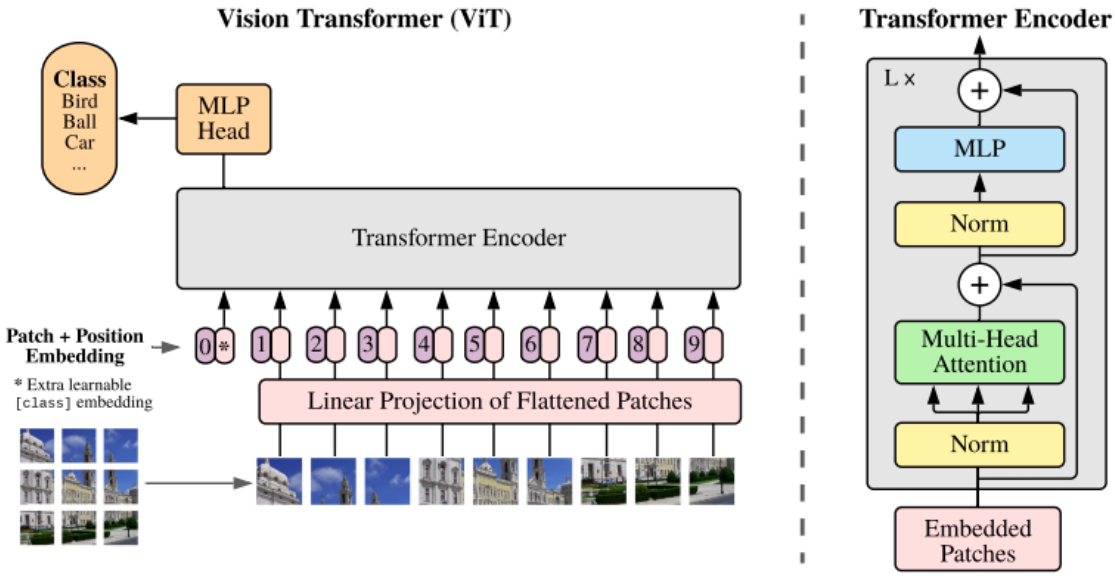


Figura 3.27: Modelo *Vision Transformer*. Imagen extraída de [28]

Este modelo, divide la imagen de entrada en parches más pequeños (sin regiones que solapen), que luego, se aplanan y se transforman en vectores, que sirven de entrada para el codificador.

El codificador difiere del diseño original en que la normalización se realiza antes de la atención y de la entrada al perceptrón multicapa.

CAPÍTULO 4
Marco Experimental

En este capítulo, introduciremos las métricas de evaluación usadas en nuestro proyecto. Posteriormente, explicaremos el conjunto de datos y expondremos los hiperparámetros usados en los experimentos que se han realizado.

4.1 Descripción de las métricas de evaluación

Una de las partes más importante en el proceso de diseñar un modelo de aprendizaje automático es evaluar el modelo, es decir, medir su rendimiento para comprobar lo bien que generaliza el modelo y la fiabilidad de las predicciones que realiza. Para ello, existen distintas métricas que miden estas características.

Recordemos que este trabajo consiste en clasificar una imagen de tejido en benigno (IDC negativo) o maligno (IDC positivo). Por lo tanto, una predicción puede ser cualquiera de estos tipos:

- Verdadero positivo (TP): Una predicción de tejido maligno, que realmente, es tejido maligno.
- Falso positivo (FP): Una predicción de tejido maligno que, en realidad no es tejido maligno.
- Verdadero negativo (TN): Una predicción de tejido benigno que, realmente, es tejido benigno.
- Falso negativo (FN): Una predicción de tejido maligno que en realidad no es tejido benigno.

Precisión

La precisión (Pr) es una métrica que representa la calidad de la predicción. En nuestro caso, se define como el porcentaje de las predicciones de tejido maligno que son, efectivamente, IDC positivo.

$$Pr = \frac{TP}{TP + FP}$$

Sensibilidad

La sensibilidad (también llamado *Recall*, y abreviado Rc), es una métrica que representa la cantidad de muestras que el modelo puede identificar correctamente. En nuestro estudio, se podría definir como el porcentaje de muestras de tejido maligno que ha detectado el modelo.

$$Rc = \frac{TP}{TP + FN}$$

Valor-F1

El Valor-F1 (también *F1-Score*) es un número que combina ambas precisión y exhausitividad, lo que lo hace útil para comparar el rendimiento combinado de ambas métricas.

$$F1 = \frac{2 \cdot Pr \cdot Rc}{Pr + Rc}$$

Especificidad

La especificidad (o *Specificity*, abreviado Spc), es una métrica similar a la precisión, pero para la otra clase, es decir, en nuestro caso se define como el porcentaje de las predicciones de tejido benigno que son, realmente, IDC negativo.

$$Spc = \frac{TN}{TN + FP}$$

Matriz de confusión

Una matriz de confusión se puede ver como una representación matricial de todas las predicciones que realiza un modelo de clasificación binario. Usaremos estas matrices para calcular las métricas explicadas anteriormente

		Predicción	
		Positivos	**Negativos**
Realidad	**Positivos**	Verdaderos Positivos (TP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (TN)

Figura 4.1: Ejemplo de cómo se construye una matriz de confusion.

Curva ROC-AUC

Una curva ROC (Receiver Operating Characteristic) es forma de representar gráficamente la sensibilidad respecto de la especificidad de un clasificador binario en función de un umbral de discriminación.

Esta curva es muy útil ya que mide la efectividad del modelo al tratar de distinguir entre ambas clases. Suele ir acompañada de un número llamado AUC (*Area Under the ROC Curve*), que se calcula como el área por debajo de la curva ROC en el intervalo [0,1], y se puede interpretar como la probabilidad de que una predicción realizada por el modelo sea correcta.

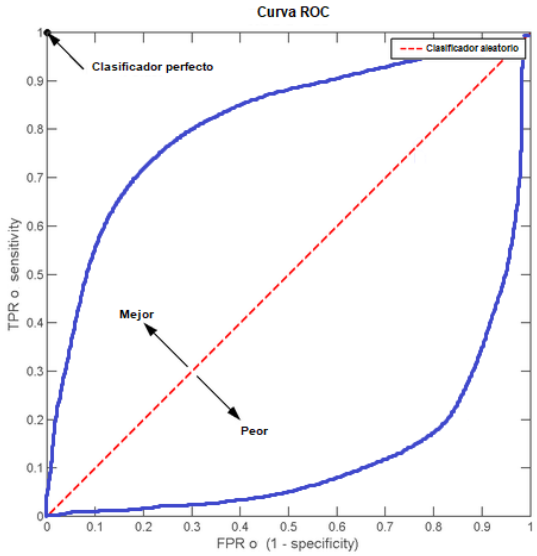


Figura 4.2: Ejemplo de el aspecto de dos curvas ROC. Cuanto más alto esté la curva de la linea diagonal del centro, mejor es el modelo.

4.2 Descripción de los datos

4.2.1. Orígen de las imágenes

El conjunto de datos proviene de la digitalización de imágenes histopatológicas tomadas mediante un microscopio a 40 aumentos de 162 mujeres diagnosticadas con carcinoma ductal invasivo en el Hospital de la Universidad de Pensilvania y en el Instituto de Cancer de Nueva Jersey [1]

Posteriormente, un patólogo experto delimitó manualmente en cada imagen la región cancerígena, y finalmente, cada imagen histopatológica fue dividida en parches de 50 x 50 píxeles de forma que aquellos que estuvieran dentro de la región delimitada fueron etiquetados como tejido maligno, y el resto, como benignos (tal y como se puede apreciar en la figura 4.3).

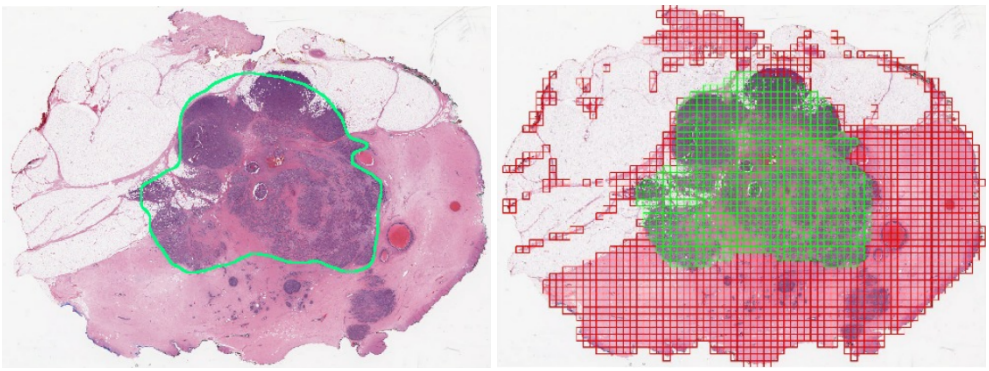


Figura 4.3: A la izquierda, imagen original con la región cancerígena delimitada. A la derecha, división y extracción de los parches de la imagen (en verde, malignos, y en rojo, benignos) [1].

Tras este proceso, el conjunto de datos resulta en 277.524 parches, donde, del total, 198.738 están etiquetados como benignos y 78.786 como malignos.[1]

4.2.2. Limpieza de los datos

En el conjunto de imágenes, se puede observar que muchas de ellas no están en condiciones de ser usadas, es decir, algunas han sufrido alguna corrupción, otras no tienen la resolución adecuada, y varias no muestran suficiente tejido y mayoritariamente están en blanco. Por tanto, es crucial eliminar de los datos estas imágenes, ya que pueden perjudicar en el proceso de aprendizaje de los modelos, y sólo aportan confusión entre clases y ruido indeseado.

Para limpiar el conjunto de datos, primero, se han descartado y eliminado directamente aquellas imágenes que no tuvieran la resolución adecuada. Cabe mencionar, que además, se han anonimizado todas las muestras y se ha eliminado cualquier referencia a las pacientes de las que se han obtenido las imágenes originales.

A continuación, mostramos algunas imágenes defectuosas, enmarcadas en rojo las pertenecientes a la clase 0, y en verde, aquellas de clase 1.

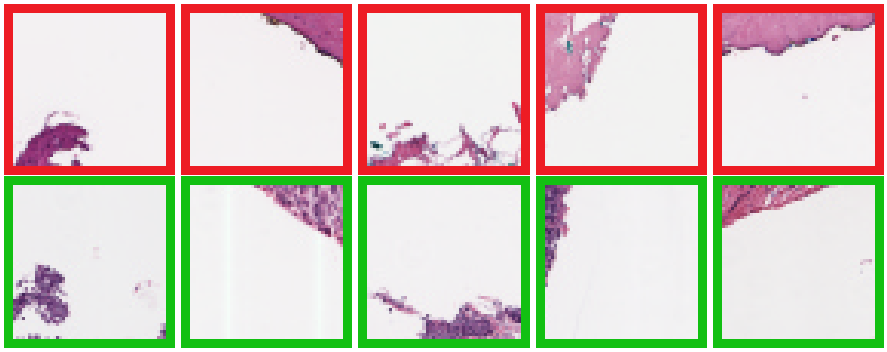


Figura 4.4: Imágenes etiquetadas de clases distintas, que muestran poco tejido y son relativamente parecidas. En rojo, benignas y en verde, malignas.

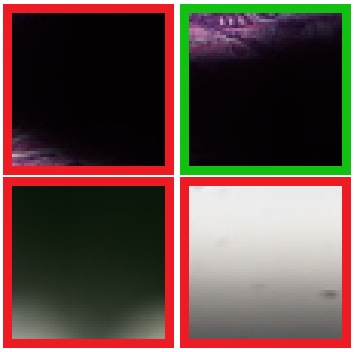


Figura 4.5: Imágenes que han sufrido alguna corrupción

Posteriormente, procedemos a eliminar manualmente aquellas imágenes no aptas para ser usadas.

[1] De ahora en adelante, nos referiremos indistintamente a las imágenes de tejido benigno como clase 0 y aquellas de tejido maligno como clase 1 y viceversa.

4.2.3. Equilibrado de clases y particionado

Una vez completado el proceso de limpieza, el número de imágenes de clase 1 queda reducida a 75.555 imágenes en total, por otro lado, las muestras de clase 0 superan con creces a las de la clase contraria.

Esto provoca que los datos aún no puedan ser usados como entrenamiento para modelos, ya que hay un claro desequilibro entre clases. Un conjunto de datos desequilibrado, por lo general, afecta a los algoritmos de entrenamiento en su proceso de generalización de la información y perjudicando a aquellas clases con menos muestras, lo que puede provocar que un modelo tienda a clasificar como perteneciente a la clase mayoritaria cualquier muestra que se le proporcione.

Para resolver este problema, se ha optado por reducir el número de muestras de la clase mayoritaria (*undersampling* o *subsampling* de la clase 0). Para ello, hemos eliminado muestras del conjunto de datos aleatoriamente hasta que permanecieran exactamente 75.555 imágenes. Tras esto, las muestras se han distribuido en particiones tal y como se muestra en la siguiente lista:

- **Conjunto de entrenamiento:** 122400 imágenes, 61200 de cada clase.
- **Conjunto de validación:** 13600 imágenes, 6800 de cada clase.
- **Conjunto de test:** 15110 imágenes, 7555 de cada clase.

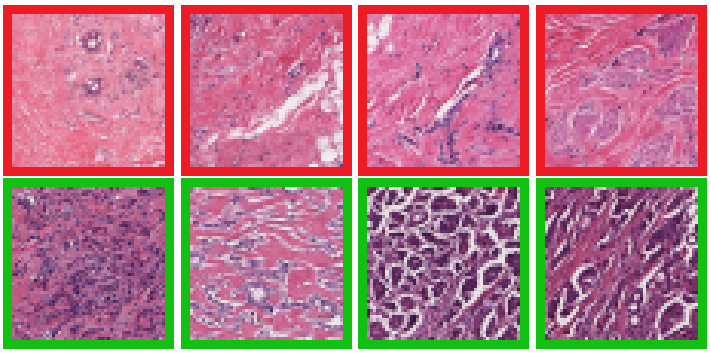


Figura 4.6: Muestras usables del conjunto de datos final.

4.2.4. Aumento de datos

Uno de los grandes problemas en el campo del aprendizaje automático es la escasez de muestras de entrenamiento, y en especial cuando se tratan de muestras médicas.

Existen diversas técnicas que permiten aumentar el volumen de los datos de entrenamiento que se disponen (*data augmentation*), que consisten en añadir copias de datos ya existentes, pero con ligeras modificaciones, o agregar datos sintéticos creados a partir de los datos disponibles.

En nuestro caso, optaremos por realizar modificaciones a los datos ya existentes. Para ello, usaremos las transformaciones ya implementadas que ofrece la librería torchvision.

Sin embargo, un reciente estudio [21] concluye que los efectos de la regularización y el aumento de datos, dependen de las clases, y no todas las modificaciones

deberían ser aplicadas a un conjunto de datos, pues pueden ser incluso perjudiciales en la fase de entrenamiento.

Esto se debe a las características intrínsecas de los datos, es decir, aquellos atributos que permiten distinguir una clase de otra. Como puede ser la forma (por ejemplo, si quisiéramos distinguir entre: manzanas, plátanos y lechugas, usar transformaciones que realicen translaciones, rotaciones o recortados puede ser perjudicial, ya que estos tres objetos se distinguen por forma y no por color, ya que, recordemos, que existen manzanas y plátanos verdes, como la lechuga), o el color, como es el caso de nuestro conjunto de datos.

Como se puede observar en la figura 4.6, las imágenes no tienen ninguna forma definida ni precisa, pero se puede distinguir la clase 0 de la 1 por el color. Por tanto, no usaremos transformaciones que afecten al color (como el ruido gaussiano o fluctuaciones de brillo y contraste), sino que utilizaremos las siguientes transformaciones:

- **RandomRotation:** Una posible modificación a los datos es permitir rotaciones aleatorias en cada imagen. Cada vez que se use la imagen, se rotará aleatoriamente un número concreto de grados En este estudio, permitiremos únicamente rotaciones en el intervalo [-90,90] con saltos de quince, es decir[-90,-75,-60...90].

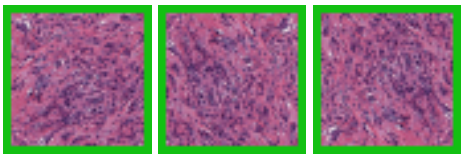


Figura 4.7: Ejemplo de una imagen original, rotada -90º y 90º.

- **RandomHorizontalFlip:** Otra posibilidad consiste en girar horizontalmente la imagen por completo, sin embargo, para evitar que todas las imágenes sean rotadas, esta modificación tendrá una asociada una probabilidad para realizar el cambio o no (ajustada manualmente al 25 %).

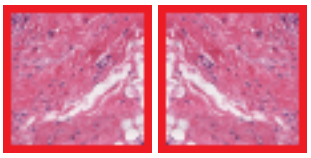


Figura 4.8: Ejemplo de una imagen original y la misma pero rotada horizontalmente.

- **RandomVerticalFlip:** Al igual que la anterior, esta modificación consiste en girar la imagen verticalmente. La cual también tiene una probabilidad asociada para rotar la imagen o dejarla igual (ajustada manualmente al 5 %).

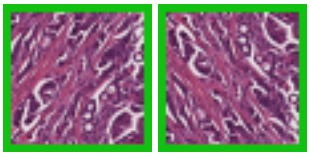


Figura 4.9: Ejemplo de una imagen original y la misma pero rotada verticalmente.

- **Normalize:** Normalizar un conjunto de datos puede parecer un esfuerzo innecesario. Pero, hay que tener en cuenta que un conjunto de muestras sin normalizar puede tener características cuyos rangos numéricos varíen mucho más que otros, lo que puede provocar que el modelo converja muy lentamente debido a que el optimizador ajusta correctamente el factor de aprendizaje para una característica pero muy erróneamente para otras.

  Para normalizar, se debe restar a cada muestra la media del conjunto y posteriormente dividir por la desviación típica. En nuestro caso, cada imagen tiene tres canales de color (RGB), por tanto, este proceso se debe realizar a cada canal individualmente.

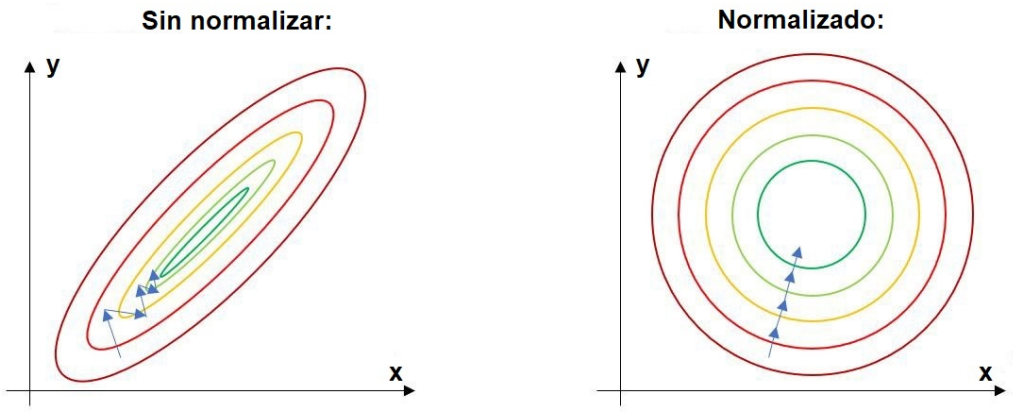


Figura 4.10: Un conjunto de datos no normalizado (izquierda) converge más lentamente que uno normalizado (derecha)

4.3 Hiperparámetros

Durante la fase de entrenamiento de un modelo, existen configuraciones de los algoritmos utilizados que afectan mucho al rendimiento y efectividad del modelo. Estos parámetros tienen un valor óptimo desconocido, y que no se pueden obtener de forma directa, por tanto, son ajustados manualmente a criterio del programador.

A continuación describiremos los hiperparámetros que hemos elegido y ajustado en este trabajo.

4.3.1. Función de pérdida

Una función de pérdida es una función que evalúa la diferencia entre las predicciones del modelo y los valores reales (etiquetas) de las muestras usadas en la fase de entrenamiento.

Esta es la función encargada de generar el gradiente, por tanto, elegir correctamente esta función es crucial para que el algoritmo de entrenamiento converja correctamente.

En nuestro estudio, hemos optado por utilizar como función de pérdida la entropía cruzada, concretamente, una de sus variantes para clasificación binaria, que se define así:

$$L_{BCE}(x,t) = -\frac{1}{N}\sum_{i=1}^{N}(t_i \cdot log(x_i) + (1-t_i)\cdot log(1-x_i)$$

Donde x representa la predicción y t el valor real.

Cabe mencionar que en la implementación, se ha usado un módulo que incorpora a la salida de la función de perdida, una función sigmoide, que aporta estabilidad numérica. [2]

4.3.2. Optimizador del factor de aprendizaje

Para entrenar un modelo, se deben modificar los parámetros del modelo de tal forma que minimicen la función de pérdida, sin embargo, esta tarea no se realiza manualmente, ya que pueden haber millones y millones de parámetros entrenables en una sola red.

De esto, se encarga el optimizador, que es una función o algoritmo que modifica los parámetros de un modelo (como los pesos de la red) en función de un factor de aprendizaje (también llamado *learning rate*), que regula la velocidad a la que se cambian los valores de los parámetros.

En nuestro estudio usaremos el optimizador AdamW [24] con un factor de aprendizaje inicial de 10^{-3} (véase figura 4.11)[3], un tipo de optimizador adaptable más estable y robusto que su predecesor, Adam [23], que calcula un factor de aprendizaje a cada parámetro individualmente, a diferencia de otros como el SGD [22], que usa un único factor con todos los parámetros.

for $t = 1$ **to** ... **do**
  if *maximize* :
    $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$
  else
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  $\theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1)g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)g_t^2$
  $\widehat{m_t} \leftarrow m_t/(1-\beta_1^t)$
  $\widehat{v_t} \leftarrow v_t/(1-\beta_2^t)$
  if *amsgrad*
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_t - \gamma\widehat{m_t}/(\sqrt{\widehat{v_t}^{max}} + \epsilon)$
  else
    $\theta_t \leftarrow \theta_t - \gamma\widehat{m_t}/(\sqrt{\widehat{v_t}} + \epsilon)$

return θ_t

Figura 4.11: Algoritmo AdamW

[2] La función de pérdida se llama BCEWithLogitsLoss, cuya documentación puede verse en: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

[3] Imagen extraida de la documentación de PyTorch: https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

4.3.3. Planificador del optimizador

Elegir el factor de aprendizaje es una tarea complicada, ya que, a priori, no se puede conocer con exactitud cual es valor óptimo de este hiperparámetro.

Un factor de aprendizaje muy alto puede provocar que las sucesivas actualizaciones de los parámetros de la red, no converjan, y no se alcance nunca un mínimo de la función de pérdida. Por otro lado, un factor de aprendizaje muy bajo haría que costara mucho tiempo entrenar un modelo, ya que tomaría mucho tiempo alcanzar un mínimo (véase ilustración de este problema en la figura 4.12[4]).

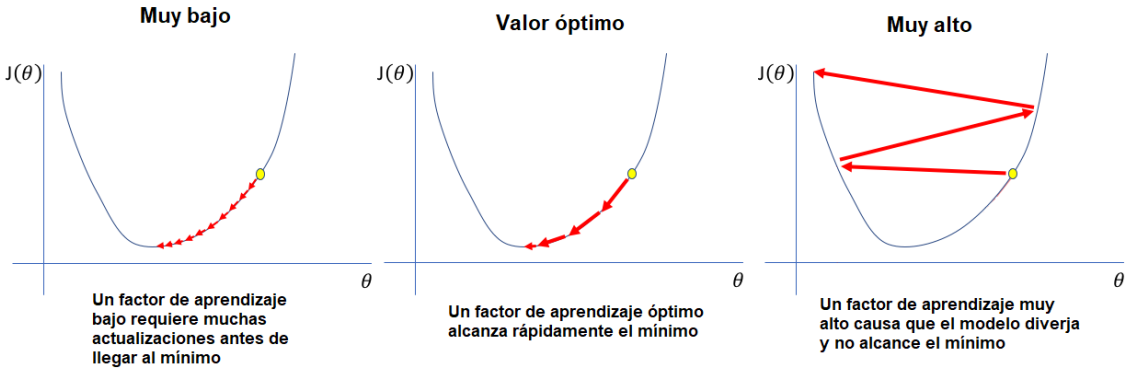


Figura 4.12: Impacto del factor de aprendizaje en la fase de entrenamiento.

Para resolver este problema en este estudio, usaremos un planificador del factor de aprendizaje (tambien llamado *learning rate scheduler*), que consiste en una función que modifica el factor de aprendizaje durante la ejecución de la fase de entrenamiento.

En nuestro caso, hemos optado por utilizar un planificador de caida exponencial, que funciona según la fórmula:

$$\theta_t = \gamma^e \cdot \theta_0$$

Donde e simboliza el número de iteración actual y siendo $\gamma = 0{,}95$ y $\theta_0 = 10^{-3}$

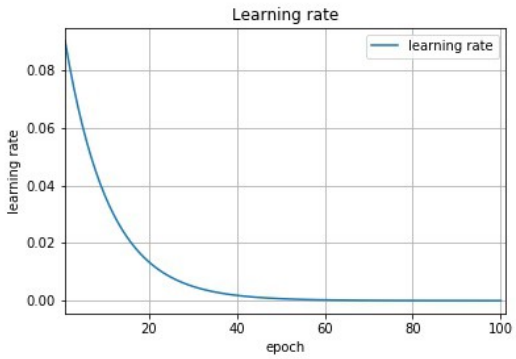


Figura 4.13: Ejemplo de cómo el planificado modifica el valor del factor de aprendizaje con el tiempo.

[4]Imagen extraída de: https://www.jeremyjordan.me/nn-learning-rate/

CAPÍTULO 5
Resultados

En este capítulo expondremos los resultados obtenidos de la aplicación de los modelos y técnicas descritas anteriormente.

Antes de comenzar, cabe mencionar que no se ha podido acceder a las particiones de test utilizadas por [1, 2, 3, 4] y por tanto los resultados no se han obtenido sobre los mismos datos. 2

Para obtener los resultados, los modelos han sido entrenados con las imágenes del conjunto de entrenamiento y validación conjuntamente y las métricas han sido calculadas sobre la partición de test (véase apartado 4.2.3)

5.1 Resultados cuantitativos

A continuación exponemos un tabla con los resultados obtenidos, en orden descendente según el Valor-F1 y el acierto equilibrado (acompañados de intervalos de confianza al 95 %).

Modelo	Precision (Pr)	Recall (Rc)	Especificidad (Spc)	Valor-F (F1-Score)	Acierto equlibrado (BAC)
EfficientNetB6	0,928	0,924	0,929	0,926	92,7% ± 0,42%
EfficientNetB3	0,922	0.93	0.92	0,926	92,6% ± 0,42%
EfficientNetB5	0,926	0,925	0,926	0,925	92,5 % ± 0,42%
EfficientNetB2	0,924	0,926	0,924	0,925	92,5% ± 0,42%
EfficientNetB1	0,921	0,927	0,92	0,924	92,4% ± 0,42%
EfficientNetB7	0,917	0,929	0,916	0,923	92,3% ± 0,43%
EfficientNetB0	0,921	0,923	0,921	0,922	92,2% ± 0,43%
EfficientNetB4	0,918	0,924	0,917	0,921	92% ± 0,43%
VGG13	0,915	0,92	0,914	0,917	91,7% ± 0,44%
VGG11	0,915	0,913	0,916	0,914	91,4% ± 0,45%
VGG16	0,916	0,911	0,917	0,913	91,4% ± 0,45%
ConvNeXt Large	0,906	0,922	0,905	0,914	91,3% ± 0,45%
ResNet50	0,908	0,916	0,908	0,912	91,2% ± 0,45%
ResNet152	0,908	0,913	0,907	0,91	91% ± 0,46%
ConvNeXt Tiny	0,914	0,904	0,915	0,909	90,9% ± 0,46%
ConvNeXt Base	0,909	0,908	0,909	0,908	90,8% ± 0,46%
ConvNeXt Small	0,918	0,893	0,92	0,905	90,7% ± 0,46%
GoogleNet	0,904	0,907	0,904	0,905	90,5% ± 0,47%
VGG19	0,898	0,914	0,896	0,905	90,5% ± 0,47%
ViT-B-16	0,909	0,895	0,91	0,901	90,3% ± 0,47%
ResNet101	0,894	0,912	0,892	0,903	90,2% ± 0,47%
AlexNet	0,842	0,859	0,839	0,85	84,9% ± 0,57%
LeNet5	0,834	0,867	0,827	0,85	84,7% ± 0,57%

Figura 5.1: Resultados ordenados según el BAC y el Valor-F.

Se puede observar que el mejor resultado obtenido corresponde al modelo EfficientNetB6, que supera los obtenidos por [1, 2, 3, 4] (véase también el capítulo 2).

La curva ROC-AUC obtenida es la siguiente:

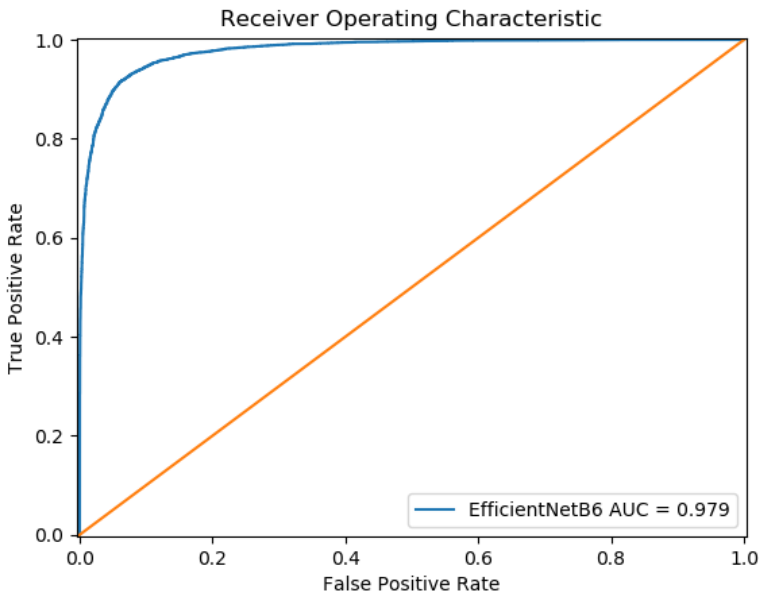


Figura 5.2: Curva ROC-AUC del modelo EfficientNetB6.

Junto con la matriz de confusión del modelo.

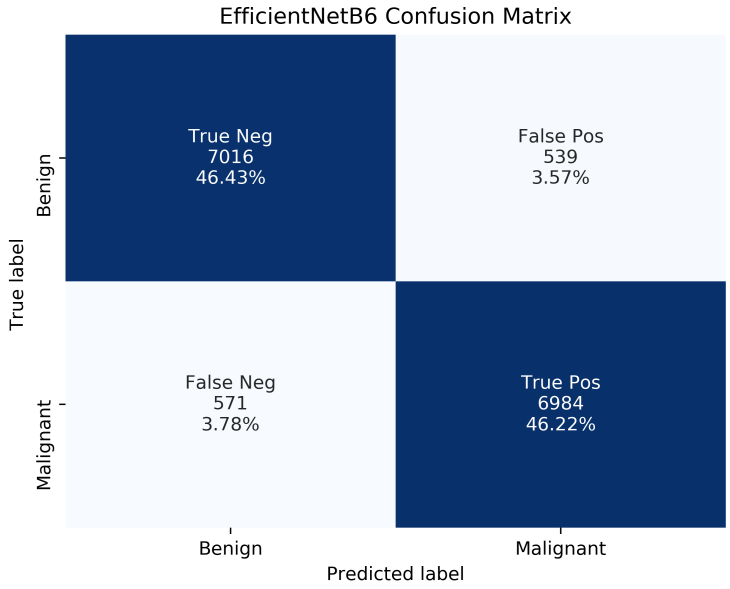


Figura 5.3: Matriz de confusión del modelo EfficientNetB6.

5.2 Resultados cualitativos

El objetivo principal del aprendizaje automático es crear un modelo que, sobre un conjunto de datos de entrenamiento sepa generalizar y predecir (o clasificar) datos del mundo real.

Sin embargo, este proceso no es perfecto y los modelos cometen errores en sus predicciones, en nuestro caso, errores de clasificación. Como ya hemos explicado en el apartado 4.1, algunas predicciones pueden ser falsos negativos, o falsos positivos. A continuación, expondremos algunos casos en los que modelo ha predicho erróneamente la clase de una imagen.

Ejemplos de falsos positivos:

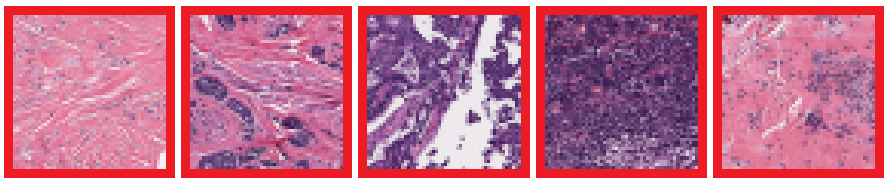


Figura 5.4: Imágenes de tejido benigno clasificadas erróneamente como tejido maligno (falsos positivos).

Ejemplos de falsos negativos:

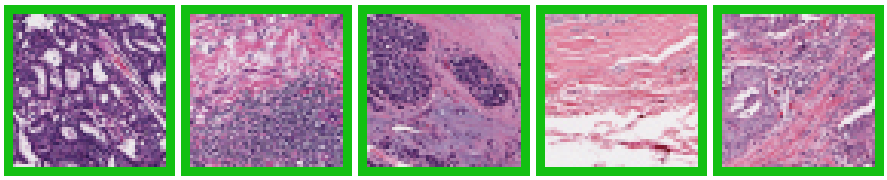


Figura 5.5: Imágenes de tejido maligno clasificadas erróneamente como tejido benigno (falsos negativos).

En contraposición a los resultados basados en métricas de evaluación, también es interesante evaluar manualmente el modelo. Para esto, hemos tomado todas los imágenes del conjunto de datos original y hemos realizado el siguiente proceso:

- Reconstruir la imagen original: A partir de los parches individuales, hemos recreado la imagen original a color (estas imágenes están en la columna «Imagen original» de la figura 5.6).
- Plasmar las etiquetas reales: Cada parche individual venía etiquetado en función de si era tejido benigno (clase 0) o tejido maligno (clase 1). Con esta información, hemos reconstruido la imagen original, pero, sustituyendo las muestras de tejido maligno por parches completamente verdes, para así tener una referencia verdadera de cuál debería ser la predicción perfecta del modelo sobre la imagen original. Las imágenes de tejido benigno se han dejado como estaban (estas imágenes están en la columna «Verdadero» de la figura 5.6).
- Predecir sobre la imagen: En este caso, también hemos reconstruido la imagen original, pero esta vez, cada parche se ha introducido al modelo (en

nuestro caso, el mejor, la EfficientNetB6, véase 5.1) y se ha obtenido una predicción, si es tejido benigno, se deja como está, en caso de ser tejido maligno, se transforma en un parche completamente verde (estas imágenes están en la columna «Predicción» de la figura 5.6).

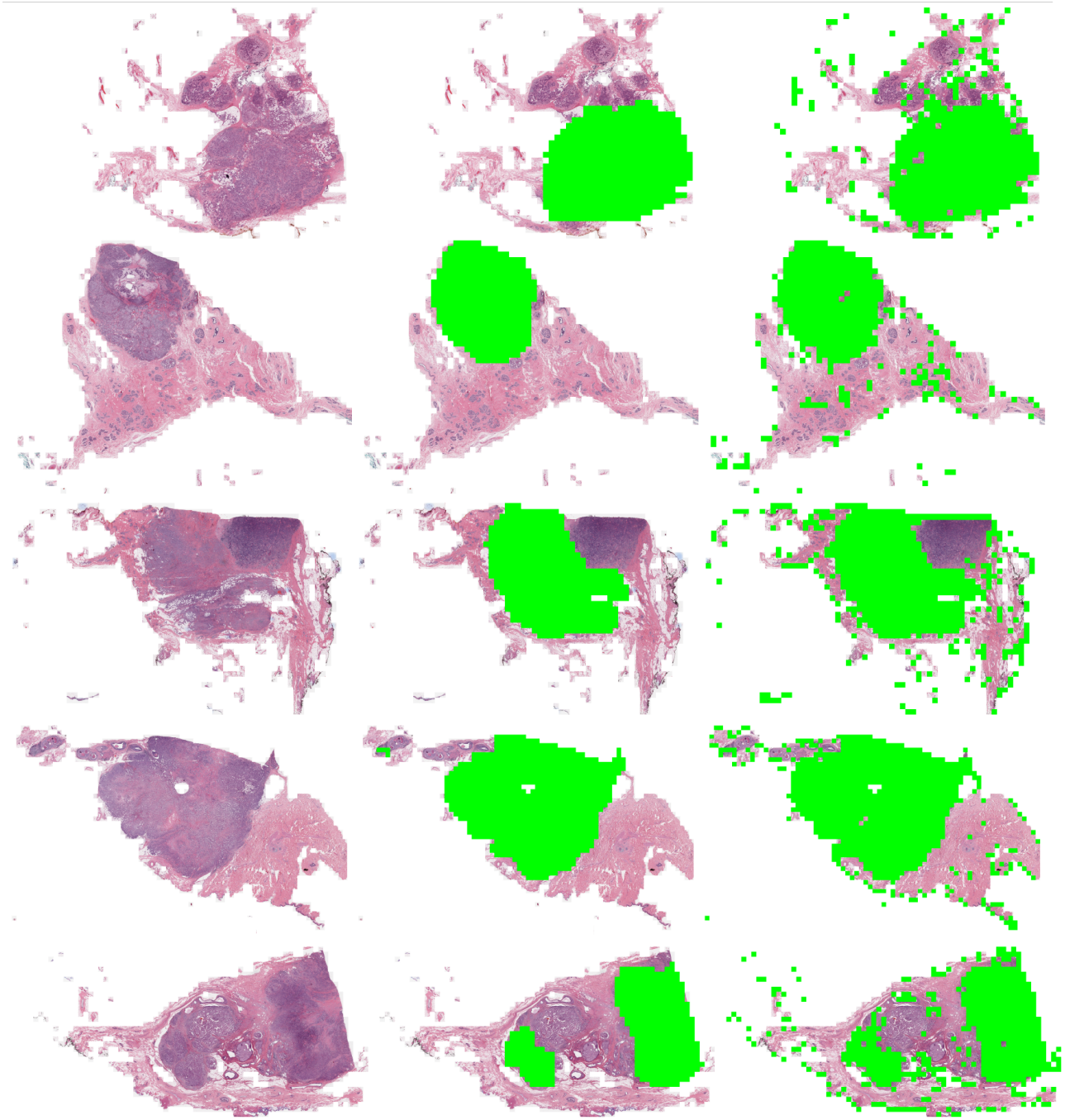


Figura 5.6: Comparación de imágenes histopatológicas, de izquierda a derecha: imagen original, imagen con el tejido cancerígeno marcado en verde, predicción del modelo .

Sin embargo, también existen casos en los que la predicción no concuerda del todo con las etiquetas reales, y por tanto, hay regiones que constituyen mayoritariamente falsos positivos.

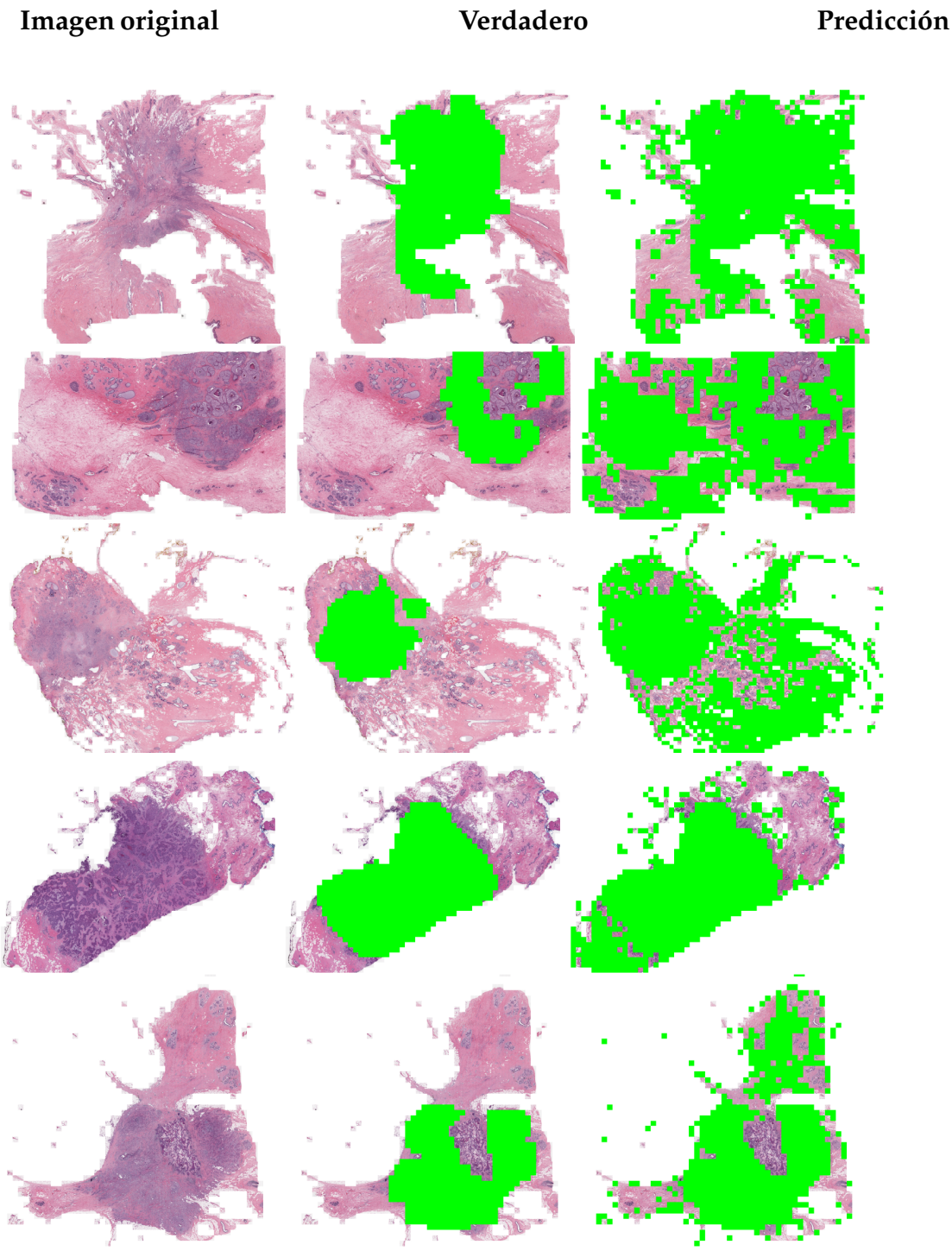


Figura 5.7: Comparación de imágenes histopatológicas cuya predicción no concuerda con las etiquetas reales, de izquierda a derecha: imagen original, imagen con el tejido cancerígeno marcado en verde, predicción del modelo .

CAPÍTULO 6
Conclusión

El carcinoma ductal invasivo es el tipo más común de cáncer de mama, detectar esta enfermedad en sus fases iniciales es crucial para poder realizar un tratamiento exitoso, sin embargo, este proceso suele ser lento y complicado. Los avances en el campo del aprendizaje automático han demostrado que son capaces de resolver una amplia variedad de tareas complicadas eficientemente y que por tanto, pueden contribuir en muchos sectores como la medicina.

Con esto, surge el principal objetivo de este estudio: crear un modelo que sea capaz de clasificar correctamente un imagen de tejido cualquier en tejido benigno (clase 0) o tejido maligno (clase 1).

Los datos usados en este estudio, en primer lugar, han sido anonimizados, y se ha borrado cualquier referencia a las pacientes de las que se han obtenido las imágenes originales. Posteriormente, para resolver unos de los problemas que más temprano han surgido, como son: el desequilibrado de clases y la escasez de muestras, se ha realizado un preprocesado que consiste limpiar manualmente los datos y usar diversas técnicas de aumento de datos para compensar, la posible escasez de estos. Por tanto, se ha logrado adecuar los datos a su uso y se ha cumplido el subobjetivo dos.

Para construir nuestra propuesta de solución, se han explorado distintos modelos de aprendizaje automático para la clasificación de imágenes: redes neuronales convolucionales y *transformers* para reconocimiento de imágenes. Con esto, podemos confirmar que el primer subobjetivo ha sido alcanzado.

Después, los modelos han sido entrenados sobre el conjunto de datos y se ha medido su rendimiento y efectividad de acuerdo con un conjunto de métricas y se han comparado los resultados con el trabajo de autores que han tratado de resolver este mismo problema.

En último lugar, se han reconstruido las imágenes originales a partir de los parches del conjunto de datos según las etiquetas reales asociadas y las predicciones del mejor modelo. Esta tarea ha sido particularmente interesante ya que ha permitido comprender mejor el resultado del trabajo de una manera muy visual en lo que respecta a la clasificación de una imagen de tejido cualquiera, y en lo que podría ser una aplicación real de este estudio. Así pues, el último subobjetivo y el objetivo principal de este trabajo, también han sido alcanzados.

Este trabajo ha demostrado ser un reto en lo profesional, debido a la necesidad de estudiar y comprender en detalle las arquitecturas y decisiones de implementación de cada uno de los modelos expuestos, los cuales, no han sido explicados

durante el grado. Además de ser una gran oportunidad de profundizar en el lenguaje de programación Python y en el uso de la librería PyTorch, ampliamente usada en el aprendizaje automático por empresas como Google, Facebook y Microsoft.

6.1 Relación con los estudios cursados

En la rama de computación, hay asignaturas que han sido de gran ayuda a la hora de realizar este trabajo.

En primer lugar, conceptos básicos sobre probabilidad, estudiados en la asignatura de estadística (EST), han sido fundamentales para entender los conceptos básicos sobre el aprendizaje automático.

Por otro lado, los conocimientos obtenidos sobre programación y las estructuras de datos existentes, explicados en la asignatura estructuras de datos y algoritmos (EDA), han sido necesarios para desarrollar un código sólido y eficiente que implemente todo lo necesario para realizar este trabajo.

Por último, las asignaturas de percepción (PER) y aprendizaje automático (APR), han introducido los conceptos teóricos básicos sobre el aprendizaje automático, el procesado de imagen, texto y audio, y el funcionamiento de las redes neuronales.

6.2 Objetivos de desarrollo sostenible

Los objetivos de desarrollo sostenible son un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad como parte de una nueva agenda de desarrollo sostenible.

En concreto, este estudio está relacionado con los siguiente objetivos:

- 3.- Salud y bienestar: El resultado de este trabajo puede ayudar a realizar detecciones preliminares de carcinoma ductal invasivo y a agilizar el proceso de detección, y por tanto, puede contribuir a salvar vidas.
- 9.- Industria, innovación e infraestructura: El campo del aprendizaje automático está en continuo crecimiento, desarrollando nuevos modelos y aplicaciones. En este trabajo se ha expuesto una aproximación con modelos del estado del arte a la clasificación de imágenes.

Bibliografía

[1] Cruz-Roa A, Basavanhally A, González F, Gilmore H, Feldman M, Ganesan S, et al. Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks. *SPIE Medical Imaging*. Vol. 9041.; 2014. p. 904103-904103-15

[2] Janowczyk A, Madabhushi A. Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases. J Pathol Inform. 2016;7:29. Published 2016 Jul 26. doi:10.4103/2153-3539.186902

[3] Saad Awadh Alanazi, M. M. Kamruzzaman, Md Nazirul Islam Sarker, Madallah Alruwaili, Yousef Alhwaiti, Nasser Alshammari, Muhammad Hameed Siddiqi, "Boosting Breast Cancer Detection Using Convolutional Neural Network", Journal of Healthcare Engineering, vol. 2021, Article ID 5528622, 11 pages, 2021. https://doi.org/10.1155/2021/5528622

[4] M. Saini and S. Susan. VGGIN-Net: Deep Transfer Network for Imbalanced Breast Cancer Dataset. IEEE/ACM Transactions on Computational Biology and Bioinformatics, doi: 10.1109/TCBB.2022.3163277.

[5] F. A. Spanhol, L. S. Oliveira, C. Petitjean and L. Heutte; A Dataset for Breast Cancer Histopathological Image Classification. IEEE Transactions on Biomedical Engineering, vol. 63, no. 7, pp. 1455-1462, July 2016, doi: 10.1109/TBME.2015.2496264

[6] Hubel, D. H.; Wiesel, T. N.. (1959) Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology* 148 (3): 574–591.

[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel; Backpropagation Applied to Handwritten Zip Code Recognition. Neural Comput 1989; 1 (4): 541–551. doi: https://doi.org/10.1162/neco.1989.1.4.541

[8] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. (2017-05-24). Ïmage-Net classification with deep convolutional neural networks". Communications of the ACM. 60 (6): 84–90. doi:10.1145/3065386

[9] Simonyan, Karen. Zisserman, Andrew. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2014. https://doi.org/10.48550/arxiv.1409.1556

[10] Lee, H., Whang, M. (2018). Heart Rate Estimated from Body Movements at Six Degrees of Freedom by Convolutional Neural Networks. Sensors (Basel, Switzerland), 18(5), 1392. https://doi.org/10.3390/s18051392

[11] Szegedy, Christian and Liu, Wei and Jia, Yangqing and Sermanet, Pierre and Reed, Scott and Anguelov, Dragomir and Erhan, Dumitru and Vanhoucke, Vincent and Rabinovich, Andrew. Going Deeper with Convolutions. 2014. https://doi.org/10.48550/arxiv.1409.484

[12] Lin, Min and Chen, Qiang and Yan, Shuicheng. Network In Network. 2013. https://doi.org/10.48550/arxiv.1312.4400

[13] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian. Deep Residual Learning for Image Recognition. 2015. https://doi.org/10.48550/arxiv.1512.03385

[14] Howard, Andrew G. and Zhu, Menglong and Chen, Bo and Kalenichenko, Dmitry and Wang, Weijun and Weyand, Tobias and Andreetto, Marco and Adam, Hartwig. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. https://doi.org/10.48550/arxiv.1704.04861

[15] Hu, Jie and Shen, Li and Albanie, Samuel and Sun, Gang and Wu, Enhua. Squeeze-and-Excitation Networks. 2017. https://doi.org/10.48550/arxiv.1709.01507

[16] Tan, Mingxing and Le, Quoc V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. 2019. https://doi.org/10.48550/arxiv.1905.11946

[17] Liu, Zhuang and Mao, Hanzi and Wu, Chao-Yuan and Feichtenhofer, Christoph and Darrell, Trevor and Xie, Saining. A ConvNet for the 2020s. 2022. https://doi.org/10.48550/arxiv.2201.03545

[18] Agarap, Abien Fred. Deep Learning using Rectified Linear Units (ReLU). 2018. https://doi.org/10.48550/arxiv.1803.08375

[19] Hendrycks, Dan and Gimpel, Kevin. Gaussian Error Linear Units (GELUs). 2016. https://doi.org/10.48550/arxiv.1606.08415

[20] Mooney, Paul. "Breast Histopathology Images."Kaggle. December 19, 2017. https://www.kaggle.com/paultimothymooney/breast-histopathology-images.

[21] Balestriero, Randall and Bottou, Leon and LeCun, Yann. The Effects of Regularization and Data Augmentation are Class Dependent. 2022. https://doi.org/10.48550/arxiv.2204.03632

[22] Herbert E. Robbins. A Stochastic Approximation Method. 2007. Annals of Mathematical Statistics. 22, 400-407

[23] Kingma, Diederik P. Ba, Jimmy. Adam: A Method for Stochastic Optimization. 2014. https://doi.org/10.48550/arxiv.1412.6980

[24] Loshchilov, Ilya. Hutter, Frank. Decoupled Weight Decay Regularization. 2017. https://doi.org/10.48550/arxiv.1711.05101

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[26] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, https://doi.org/10.48550/arxiv.1412.3555, 2014.

[27] Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N. and Kaiser, Lukasz and Polosukhin, Illia. Attention Is All You Need. 2017. https://doi.org/10.48550/arxiv.1706.03762

[28] Dosovitskiy, Alexey and Beyer, Lucas and Kolesnikov, Alexander and Weissenborn, Dirk and Zhai, Xiaohua and Unterthiner, Thomas and Dehghani, Mostafa and Minderer, Matthias and Heigold, Georg and Gelly, Sylvain and Uszkoreit, Jakob and Houlsby, Neil. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020. https://doi.org/10.48550/arxiv.2010.11929

APÉNDICE A
Reproducibilidad de los resultados

En este apéndice se explica cómo se puede acceder al código fuente, a los datos y cómo reproducir los resultados expuestos.

Este proyecto ha sido desarrollado con la intención de ser de código abierto, y por tanto, el código fuente está disponible en:

https://github.com/OverKoder/IDC-Detection

En la descripción del repositorio se encuentran instrucciones para cómo ejecutar el código disponible para reproducir los resultados.

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**			X	
ODS 9. **Industria, innovación e infraestructuras.**		X		
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Los Objetivos de Desarrollo Sostenible son un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos y todas como parte de una nueva agenda de desarrollo sostenible. Cada objetivo tiene metas específicas como son:

- I. Fin de la pobreza
- II. Hambre cero
- III. Bienestar y salud
- IV. Educación de calidad
- V. Igualdad de g´enero
- VI. Saneamiento y agua limpia
- VII. Energía asequible y no contaminante
- VIII. Crecimiento económico y trabajo decente
- IX. Industria, innovación e infraestructura
- X. Reducción de las desigualdades
- XI. Ciudades y comunidades sostenibles
- XII. Consumo y producción responsables
- XIII. Acción por el clima
- XIV. Vida submarina
- XV. Vida de ecosistemas terrestres
- XVI. Paz, justicia e instituciones
- XVII. Alianzas para lograr objetivos.

En cuanto a lo que respecta al contenido de este documento, de todos los objetivos, podemos relacionar directamente este estudio con los siguientes objetivos:

- Objetivo 3 - Bienestar y salud: El resultado final de este estudio puede ser aplicado directamente al campo de la medicina, y usado, para realizar de forma automática detecciones preliminares de carcinoma ductal invasivo, contribuyendo as´ı a aliviar carga de trabajo y posiblemente, salvando vidas.

- Objetivo 8 - Crecimiento económico y trabajo decente: Los modelos y arquitecturas diseñadas en el campo del aprendizaje automático son cada vez m´as y m´as utilizados por empresas para: aliviar carga de trabajo costosa o tediosa, obtener predicciones sobre el modelo de negocio y desarrollar estrategias de mercado eficaces y rentables, y también, para apoyar a los empleados en el desempeño de tareas delicadas. Por tanto, crear un modelo que pueda servir de guía para la detección del carcinoma ductal invasivo, es sin duda, un aporte sustancial a este objetivo.

• Objetivo 9 - Industria, innovación e infraestructura: El campo del aprendizaje automático es una disciplina que se encuentra en constante crecimiento y expansión. Donde cada vez surgen modelos y arquitecturas más sofisticados y refinados para resolver problemas de todo tipo y clase en todas las ramas del campo de la inteligencia artificial. Los modelos explorados en este documento proporcionan una aproximación desde el estado del arte en la clasificación de imágenes al objetivo de este trabajo, que además, innovan en todos los estudios previos de este mismo problema.

En cuanto al resto de Objetivos de Desarrollo Sostenible, no se puede establecer ninguna relación directa con ninguno de los objetivos sobrantes. Debido a que algunos de ellos, durante el planteamiento, desarrollo y mantenimiento de este estudio, no se han propuesto como objetivos alcanzables y por tanto, se ha decidido emplear tiempo y recursos en otras áreas del proyecto cuyas metas sí han sido definidas y aceptadas.

Por otro lado, hay Objetivos de Desarrollo Sostenible, cuyas metas no son posibles de alcanzar o de realizar un aporte sustancial a ellas, debido a que no hay ningún aspecto, ámbito o habilidad que se haya adquirido o estudiado durante el cursado del grado que permita desarrollar un trabajo o estudio útil para alcanzar más objetivos o metas propuestas a parte de los ya explicados anteriormente.