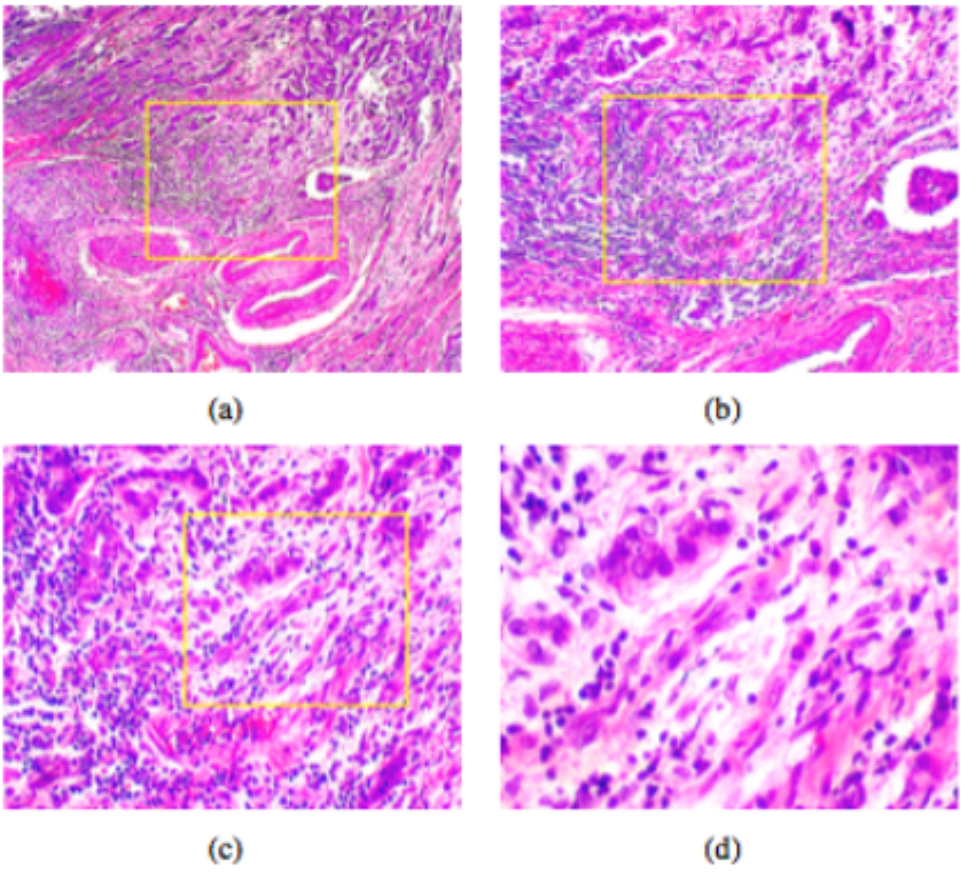
Detección de cáncer de mama a través de redes convolucionales



Realizado por:

Marina Calero López
Lucas Manuel Herencia Solís
Juan Antonio Moreno Moguel

May 10, 2025

Resumen

Este proyecto se enmarca en la asignatura de Procesamiento de Imágenes Digitales (PID) y tiene como objetivo desarrollar un sistema que detecte el cáncer de mama mediante la identificación y comparación de células cancerígenas benignas y malignas. La metodología se basará en el uso de redes neuronales convolucionales (CNN) para analizar imágenes digitales de tejido mamario y extraer patrones característicos. Se realizan procesos de preprocesamiento y segmentación para aislar las áreas de interés, seguidos de la extracción de características específicas que permitan distinguir entre células malignas y benignas. Con este enfoque, se busca crear una herramienta de apoyo al diagnóstico clínico, que contribuya a una detección temprana y más precisa de la enfermedad. Además se realiza un estudio comparativo entre este enfoque y un enfoque más tradicional el cual aplica el clasificador K-Nearest Neighbours(KNN).

Palabras clave: cáncer de mama, redes neuronales convolucionales (CNN), células, benigno, maligno.

Índice

1. Introducción

El cáncer de mama es una de las principales causas de mortalidad en el mundo, habiendo sido la responsable de 670.000 muertes en 2022 siendo a su vez el tipo de cáncer más común en las mujeres según [1]. Esto provoca que durante los últimos años se haya estado realizando una labor social enorme referente a la concienciación sobre el cáncer de mama dando gran importancia a su pronta detección.

Gracias a los avances en Deep Learning [2], se han desarrollado métodos innovadores para analizar imágenes histológicas y detectar el cáncer de mama. En este contexto, existen dos aproximaciones que se han estudiado en la literatura, como en el artículo ***"Classification of breast cancer based on histology images using convolutional neural networks"*** [3], soportada por los 445 artículos en los que se realiza un estudio entre dos enfoques.

Con el objetivo de desarrollar una aplicación capaz de detectar cáncer de mama a partir de imágenes histológicas, utilizaremos dos enfoques distintos. El *primer enfoque* se basa en una red neuronal convolucional (**CNN**), construida desde cero siguiendo la arquitectura de [3], que permite realizar una clasificación directa de las imágenes entre tejido benigno y maligno. El *segundo enfoque* emplea un modelo autoencoder convolucional preentrenado, obtenido desde un repositorio en línea, del cual se extrae la capa de menor dimensionalidad conocida como cuello de botella. Esta representación comprimida de las imágenes, siendo un espacio latente de 48 dimensiones, se utiliza como entrada para el algoritmo K-Nearest Neighbors (**KNN**), que permite clasificar nuevas muestras en función de su similitud con otras ya conocidas.

Ambos métodos son implementados y evaluados con el objetivo de comparar su rendimiento, eficiencia y aplicabilidad dentro de un sistema de diagnóstico asistido por computadora. Esta comparación busca no solo medir la precisión, sino también explorar qué tipo de arquitectura resulta más adecuada para un entorno clínico real, donde la interpretabilidad, la escalabilidad y el tiempo de respuesta son factores clave.

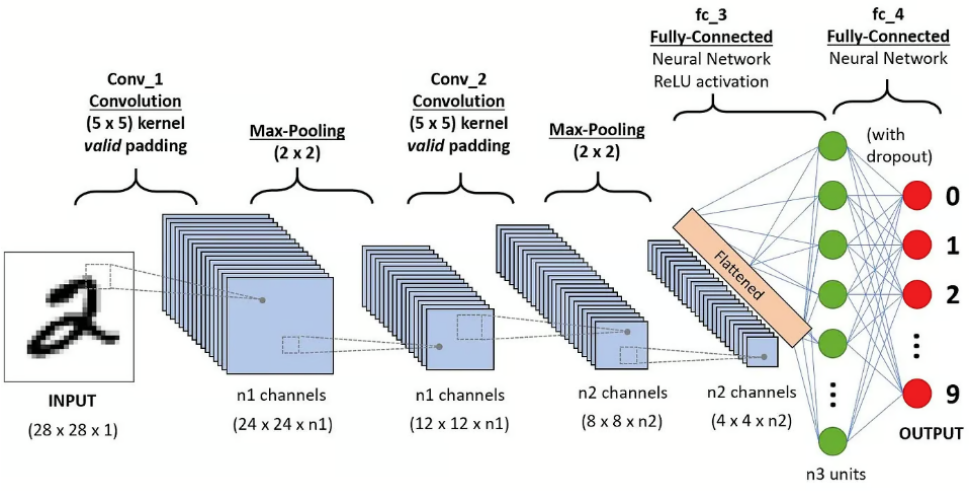


Figura 1: Capas convolucionales [4]

2. Planteamiento del Teórico

2.1. Objetivos del Proyecto

Este trabajo tiene como objetivo principal analizar y comparar diferentes metodologías de clasificación de imágenes médicas, con especial atención en la detección temprana del cáncer de mama. Para ello, se estudian tanto enfoques tradicionales de clasificación en torno a la extracción manual de características, como enfoques modernos basados en aprendizaje profundo.

Los objetivos específicos del proyecto son los siguientes:

1. **Crear** una aplicación que permita a los médicos detectar el cáncer de mama solo dando la imagen.
2. **Desarrollar** distintos modelos que permitan la clasificación binaria de las imágenes.
3. **Evaluar** el rendimiento de distintos modelos, tanto CNN en tareas de clasificación binaria como KNN.
4. **Realizar experimentación para análisis** del impacto de las técnicas de aumento de datos sobre el desempeño de los modelos.

2.2. Tecnologías Utilizadas

Para el desarrollo del proyecto se ha utilizado el lenguaje de programación **Python**[5], ampliamente adoptado en la comunidad científica por su simplicidad, legibilidad y amplio ecosistema de bibliotecas para el análisis de datos, visión por computador y aprendizaje automático.

El desarrollo de la interfaz de nuestra aplicación ha sido desarrollado utilizando **Tkinter**[6], permitiendo de esta forma una selección y visualización de resultados cómoda de las imágenes junto a su clasificación en benigna o maligna.

Entre estas bibliotecas, se ha seleccionado **TensorFlow**[7] como herramienta principal para la implementación y comparación de modelos. TensorFlow es una librería de código abierto desarrollada por Google que permite construir, entrenar y desplegar modelos de *machine learning*, especialmente redes neuronales profundas. Su flexibilidad y eficiencia lo convierten en una plataforma ideal tanto para métodos tradicionales como para arquitecturas más complejas basadas en aprendizaje profundo.

2.3. Redes Neuronales Convolucionales (CNN)

Una **Red Neuronal Convolucional** (CNN, por sus siglas en inglés) son una clase especializada de redes neuronales artificiales diseñadas para procesar datos con una estructura de tipo cuadrícula, siendo las imágenes un ejemplo típico. A diferencia de las redes neuronales tradicionales, las CNN son capaces de aprovechar la correlación espacial entre píxeles para identificar patrones jerárquicos, lo que las convierte en una herramienta

sumamente eficaz en tareas de visión por computadora, análisis biomédico, reconocimiento de voz, entre otras aplicaciones.

La arquitectura típica de una CNN está compuesta por una secuencia de capas que extraen representaciones abstractas de los datos de entrada. La primera etapa esencial en una CNN es la capa convolucional. Esta capa aplica filtros o *"kernels"*, que recorren la imagen de entrada realizando una operación matemática denominada convolución. Cada filtro aprende a detectar características locales específicas, como formas, texturas o regiones de contraste. El resultado de esta operación es un conjunto de mapas de activación que reflejan la presencia de determinadas características en distintas regiones de la imagen [8].

Tras la convolución, se aplica comúnmente una función de activación no lineal. La más utilizada en redes modernas es la función **ReLU** (Rectified Linear Unit), que introduce no linealidades en el modelo al anular los valores negativos. Esta operación permite a la red aprender representaciones más complejas que no podrían modelarse únicamente con funciones lineales [9]. Sin embargo, una limitación conocida de ReLU es que, si las entradas son negativas, la función devuelve cero, lo que puede causar que algunas neuronas dejen de actualizarse durante el entrenamiento, un fenómeno conocido como el "*problema de las neuronas muertas*". Para mitigar esta situación, se utiliza con frecuencia una variante llamada **Leaky ReLU**, la cual modifica la función para que permita un pequeño gradiente también en la región negativa. Matemáticamente, se define como:

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{para } x > 0 \\ \alpha x & \text{para } x \leq 0 \end{cases}, \quad \alpha = 0{,}01 \text{ (valor típico)}$$

. Esta modificación mejora la capacidad de aprendizaje de la red al evitar que las neuronas queden inactivas de manera permanente [10].

Posteriormente, se emplea una capa de agrupamiento, o pooling, que tiene como objetivo reducir la dimensionalidad espacial de los mapas de activación. Esta reducción permite disminuir el número de parámetros y operaciones computacionales, al tiempo que se mantiene la información más relevante. El agrupamiento también aporta robustez frente a pequeñas variaciones o distorsiones en la entrada [11].

Las características extraídas a lo largo de las capas anteriores son finalmente procesadas por una o más capas completamente conectadas, también conocidas como *fully connected layers*. En esta etapa, cada neurona está conectada a todas las salidas de la capa anterior, y la red combina las representaciones obtenidas para realizar una inferencia basada en el conocimiento aprendido. Esta parte del modelo funciona como un clasificador que transforma las representaciones espaciales en una predicción final.

La red concluye con una capa de salida que depende del tipo de problema que se desea resolver. En tareas de clasificación multiclase, se utiliza generalmente una función *softmax* que devuelve una distribución de probabilidad sobre las clases. Para problemas de clasificación binaria, la función sigmoide es comúnmente empleada, ya que proporciona una probabilidad entre 0 y 1 que indica la pertenencia de la entrada a una de las dos clases posibles [12].

El proceso de aprendizaje de una CNN se realiza mediante retropropagación, un algoritmo que ajusta los parámetros de la red a partir del error entre las predicciones y las etiquetas verdaderas. Este error se cuantifica mediante una función de pérdida, siendo la entropía cruzada una de las más utilizadas en problemas de clasificación. A su vez, algoritmos de optimización como Adam o Stochastic Gradient Descent (**SGD**) se encargan de actualizar los pesos del modelo en cada iteración, guiando el proceso de entrenamiento hacia una solución que minimice la pérdida [13], [14].

En el apartado de Implementación se describe detalladamente como se ha construido la CNN usada.

2.4. Autoencoder

La estructura básica de un autoencoder consta de dos componentes principales: un codificador (**encoder**) y un decodificador (**decoder**). El codificador actúa de manera análoga a las primeras capas de una *CNN*, tomando la entrada y transformándose progresivamente en una representación de menor dimensionalidad. Esta representación comprimida se denomina **espacio latente**, y constituye el núcleo informativo del autoencoder. Es en este espacio donde se concentran las características más relevantes de la entrada, de manera que, incluso tras eliminar redundancias o ruido, se retiene la información esencial [15].

En aplicaciones que integran CNN con autoencoders frecuentemente llamadas **convolutional autoencoders**, el codificador suele incluir capas convolucionales que permiten aprovechar las estructuras espaciales locales, de forma similar a una CNN estándar. Este diseño no solo reduce la dimensionalidad, sino que también conserva información espacial crítica para la reconstrucción posterior [16].

El **espacio latente** cumple una función crucial: actúa como un cuello de botella que obliga al modelo a aprender una codificación eficiente. A diferencia de una simple reducción de tamaño, esta compresión es aprendida automáticamente por la red en función de su capacidad para reconstruir los datos de entrada. Por esta razón, la calidad de la representación latente determina directamente el rendimiento del autoencoder. Si la codificación es adecuada, el decodificador será capaz de reconstruir la entrada con alta fidelidad, incluso cuando se utilice una representación mucho más compacta [17], [18].

El **decodificador**, por su parte, invierte el proceso realizado por el codificador. A partir del espacio latente, genera una reconstrucción de la entrada original utilizando capas que gradualmente expanden la dimensionalidad. En autoencoders convolucionales, esta expansión se realiza mediante **capas convolucionales transpuestas** o **upsampling**, que permiten reconstruir las estructuras espaciales originales. A diferencia del codificador, que se enfoca en extraer características, el decodificador se centra en aprender cómo esas características pueden combinarse para generar una salida coherente [16].

2.5. K-Nearest Neighbours (KNN)

El algoritmo **K-Nearest Neighbors** (k-NN) es uno de los métodos más simples y efectivos dentro del aprendizaje automático supervisado. A diferencia de modelos como las redes neuronales, que requieren un proceso de entrenamiento explícito, el k-NN es un método perezoso (*lazy learner*), es decir, no construye un modelo a priori, sino que realiza la clasificación directamente a partir de los datos de entrenamiento disponibles [19].

La idea fundamental detrás del k-NN es que una instancia se clasifica en función de las clases mayoritarias de sus **k vecinos más cercanos** en el espacio de características. Para determinar estos vecinos, se calcula la distancia entre la instancia a clasificar y cada punto del conjunto de entrenamiento, siendo la distancia euclidiana la medida más común, aunque otras como la distancia de Manhattan o la distancia de Mahalanobis también son utilizadas dependiendo del contexto del problema [20].

En el marco de arquitecturas más complejas como las **Redes Convolucionales** (CNN) o los autoencoders, el k-NN se utiliza frecuentemente como clasificador final sobre las representaciones extraídas por estas redes. Por ejemplo, en lugar de utilizar capas completamente conectadas o *softmax* para la **clasificación final**, es posible aplicar k-NN sobre el **espacio latente** de un autoencoder o sobre los mapas de activación extraídos por las capas convolucionales. Esta estrategia es particularmente útil cuando se desea evaluar la **capacidad discriminativa de la representación aprendida**, independiente de la arquitectura del clasificador [21].

Una de las principales ventajas del k-NN en estos contextos es su **robustez** y **simplicidad**. No requiere entrenamiento adicional una vez se ha obtenido la representación latente, y su rendimiento depende únicamente de la calidad del espacio de características. Esto lo convierte en una herramienta ideal para comparar distintas representaciones, evaluar embeddings o realizar pruebas rápidas sin necesidad de reentrenar redes profunda [22].

Sin embargo, KNN también presenta limitaciones. Su eficiencia computacional disminuye cuando el número de muestras o la dimensionalidad del espacio de características es muy alta un problema conocido como la "*maldición de la dimensionalidad*". Por ello, su uso suele estar precedido por una etapa de reducción de dimensionalidad o extracción de características, como las proporcionadas por autoencoders o CNN, que ayudan a preservar solo la información más relevante [23].

3. Implementación

El desarrollo de este proyecto se fundamenta en el uso de técnicas de aprendizaje automático y profundo para la clasificación de imágenes histológicas de cáncer de mama. Con el objetivo de comparar el rendimiento entre distintos enfoques, se han implementado y entrenado dos tipos de modelos: el primero, una CNN, y el segundo, un autoencoder del cual se obtiene la capa de menor dimensionalidad o *bottleneck* y usando después un clasificador K-Nearest neighbours para obtener el resultado. Ambos modelos han sido entrenados utilizando un conjunto de imágenes histológicas públicamente disponibles.

3.1. Descripción del dataset

Para el desarrollo de este proyecto se utilizó el **dataset de imágenes histológicas de cáncer de mama BreaKHis** [24], asociado al trabajo de [3], ampliamente reconocido en la literatura científica por su utilidad en investigaciones de diagnóstico asistido por computadora en cáncer de mama. Este dataset contiene un total de 9,109 imágenes histopatológicas de tejido tumoral mamario, recolectadas de 82 pacientes en [25].

Las muestras se obtuvieron mediante biopsia excisional (**SOB**), también conocida como mastectomía parcial, un procedimiento quirúrgico realizado bajo anestesia general que permite extraer una porción significativa de tejido para su análisis. Las imágenes fueron teñidas con hematoxilina y eosina (H&E), y capturadas mediante un microscopio óptico en cuatro niveles de aumento: 40X, 100X, 200X y 400X. Todas las imágenes presentan una resolución de 700×460 píxeles, están codificadas en formato PNG, y utilizan un modelo de color RGB de 3 canales a 8 bits por canal.

BreaKHis está dividido en dos clases principales: tumores **benignos** (2,480 imágenes) y tumores **malignos** (5,429 imágenes). Cada una de estas clases se subdivide en cuatro tipos histológicos distintos, basados en la morfología observada al microscopio:

- **Benignos:** Adenosis (A), Fibroadenoma (F), Tumor filoides (PT) y Adenoma tubular (TA).
- **Malignos:** Carcinoma ductal (DC), Carcinoma lobulillar (LC), Carcinoma mucinoso (MC) y Carcinoma papilar (PC).

En esta investigación se utilizó una versión del dataset disponible públicamente en [24], la cual se encuentra organizada de forma estructurada para entrenamiento, validación y evaluacion, dividiendose a su vez en magnificacion y tipo, lo que facilitó la gestión y preparación de los datos para el entrenamiento de los modelos.

3.2. Arquitectura de la aplicación

Esta aplicación de escritorio ha sido desarrollada con Python utilizando la biblioteca **Tkinter** y la extensión **tkinterdnd2** para soporte de arrastrar y soltar. Está diseñada

para facilitar el análisis automatizado de imágenes microscópicas de tejidos mamarios, permitiendo la clasificación entre lesiones benignas y malignas mediante modelos de machine learning. La interfaz ofrece una experiencia visual amigable y funcional para profesionales o investigadores del área médica.

Principales funcionalidades:

1. **Carga de imágenes eficiente:** Permite seleccionar múltiples imágenes desde el sistema de archivos o mediante arrastrar y soltar directamente sobre el área de previsualización. Soporta formatos como PNG, JPG, JPEG, BMP y GIF.

2. **Previsualización y navegación:** Muestra una imagen ampliada de la muestra seleccionada y un carrete con miniaturas generadas automáticamente, facilitando la revisión manual. Se utiliza PIL.Image para procesar imágenes y ImageTk.PhotoImage para su integración en la GUI.

3. **Selector de modelo de análisis:** Incluye un desplegable con modelos disponibles: KNN y CNN, con variantes según niveles de aumento (40x, 100x, 200x, 400x), permitiendo seleccionar el tipo de entrenamiento más adecuado según la resolución de las imágenes.

4. **Ejecución y control del análisis:** Los análisis se realizan de forma secuencial mediante hilos (threading.Thread) para mantener la interfaz fluida. Los modelos son invocados a través de funciones importadas desde los módulos knn.py y cnn.py, que devuelven una predicción y un color representativo.

5. **Visualización de resultados y estadísticas:** Para cada imagen se muestra el resultado (Benigno/Maligno) con retroalimentación visual (círculo coloreado) y se actualizan las estadísticas globales: cantidad total de imágenes, benignas y malignas. Se incluye también una barra de progreso (ttk.Progressbar).

6. **Gestión del análisis:** Se ofrecen botones para iniciar, pausar/reanudar y eliminar las imágenes seleccionadas. Los resultados se limpian automáticamente al cargar nuevas imágenes o al reiniciar el análisis.

3.3. Arquitectura CNN

Para la implementación del modelo CNN se ha seguido la arquitectura descrita en el artículo "***Classification of Breast Cancer Based on Histology Images Using Convolutional Neural Networks***" [3], en el cual se desarrolla una red neuronal convolucional (CNN) especialmente diseñada para abordar el desafío de clasificar imágenes histológicas de tejido mamario teñidas con hematoxilina y eosina (H&E).

El modelo desarrollado presenta una estructura jerárquica compuesta por *5 capas convolucionales*, seguidas de dos capas completamente conectadas y una capa de salida con activación *softmax*. Cada capa convolucional aplica filtros (también denominados *kernels*) de tamaño 3×3, que se deslizan sobre la imagen para detectar patrones locales como bordes, texturas o regiones celulares características. En detalle, la *primera* capa genera 64 mapas de características; la *segunda*, 96; la *tercera*, 128; y tanto la *cuarta* como la *quinta*, 256 mapas. Estas capas permiten una extracción progresiva de características, pasando

de representaciones simples a otras más abstractas y complejas.

Tras las capas convolucionales, se emplean operaciones de **max pooling** en la primera, segunda y quinta capa, utilizando ventanas de 3×3 con un stride de 2. El *max pooling* selecciona el valor máximo dentro de cada región, reduciendo así la resolución espacial de las representaciones internas y proporcionando invariancia ante pequeñas traslaciones. El *stride*, o paso de desplazamiento, controla cuántos píxeles se avanza al aplicar esta operación; un valor de 2 reduce la dimensionalidad de forma más agresiva. Las capas tercera y cuarta no aplican *pooling*, lo cual permite conservar la resolución espacial de las características extraídas, favoreciendo una mayor sensibilidad a detalles finos del tejido.

Todas las capas convolucionales están seguidas por una función de activación ReLU (`Rectified Linear Unit`), definida como:

$$f(x) = \max(0, x) \tag{1}$$

, que introduce no linealidad en el modelo y mejora la eficiencia del entrenamiento al evitar el problema del desvanecimiento del gradiente. Los pesos de estas capas se inicializan a partir de una distribución normal con desviación estándar pequeña (0,01), y se aplica regularización L2 (*weight decay*) con un coeficiente de penalización de $\lambda = 10^{-3}$, lo cual ayuda a evitar el sobreajuste penalizando los pesos excesivamente grandes.

A la salida de las capas convolucionales, los mapas de activación se aplanan mediante una capa *Flatten*, que convierte las matrices tridimensionales en un vector unidimensional para conectarlas con las capas densas. La primera capa densa contiene 2000 neuronas con activación ReLU y se acompaña de una capa *Dropout* con una probabilidad de desactivación del 50 %. Esta técnica consiste en *"apagar"* aleatoriamente la mitad de las neuronas durante el entrenamiento, reduciendo así la dependencia entre unidades y mejorando la capacidad de generalización del modelo. Finalmente, la segunda capa densa contiene tantas neuronas como clases en el problema (en este caso, dos) y emplea una activación *softmax*, que convierte las salidas en una distribución de probabilidad sobre las clases, facilitando la clasificación final.

El modelo se entrena utilizando el algoritmo Stochastic Gradient Descent (SGD), que actualiza los pesos en función de pequeños subconjuntos aleatorios de datos (*minibatches*), en lugar de utilizar todo el conjunto de entrenamiento, lo que acelera el proceso y mejora la generalización. La tasa de aprendizaje se fija inicialmente en 0,001, controlando la magnitud de los ajustes en los pesos, y se aplica un *momentum* de 0,9, que introduce una memoria del gradiente pasado para suavizar las oscilaciones y acelerar la convergencia. La función de pérdida utilizada es categorical *crossentropy*, adecuada para problemas de clasificación multiclase con etiquetas codificadas en formato *one-hot*.

Finalmente, el modelo puede entrenarse sobre imágenes con distintas magnificaciones o bien utilizando un conjunto combinado, y se proporciona la funcionalidad para guardar tanto el modelo completo como únicamente los pesos aprendidos. Esta flexibilidad resulta útil tanto para entrenamientos posteriores como para su integración en entornos clínicos.

Al finalizar el proceso de entrenamiento, se generan 5 archivos con el mismo modelos y con los pesos obtenidos tras el entrenamiento, cada uno de estos se ha entrenado con

un conjunto de datos distintos, ya sea con todos los datos, con los de magnificación 40X, 100X, 200X o 400X.

3.4. Arquitectura KNN

Se emplea un autoencoder convolucional basado en el artículo [26] para la recuperación de imágenes histológicas de cáncer de mama. Este modelo, compuesto por un codificador y un decodificador, extrae representaciones latentes útiles para clasificación o búsqueda por similitud. Se entrena minimizando el error de reconstrucción mediante optimizadores como SGD o Adam, utilizando conjuntos de entrenamiento y validación para evitar el sobreajuste.Este tipo de red se caracteriza por una estructura simétrica compuesta por dos partes principales: el codificador y el decodificador..

1. **Codificación (Encoder):** cse encarga de transformar la imagen de entrada en una representación latente de baja dimensionalidad mediante una serie de capas convolucionales con filtros de tamaño 3×3 y funciones de activación ReLU. Estas capas están acompañadas por operaciones de max pooling que reducen progresivamente la resolución espacial, extrayendo características jerárquicas de alto nivel. El resultado es un vector latente de **48 dimensiones**, correspondiente al cuello de botella del modelo, que encapsula la información visual más relevante de la imagen.

2. **Decodificación (Decoder):** intenta reconstruir la imagen original a partir de esta representación comprimida, utilizando capas de upsampling y convolucionales dispuestas simétricamente respecto al codificador. El entrenamiento del autoencoder se realiza minimizando el error cuadrático medio (MSE) entre la imagen de entrada y su reconstrucción, obligando así al modelo a aprender una codificación eficiente y significativa.

Tras el proceso de extracción de características mediante el autoencoder convolucional, cada imagen del conjunto de datos es transformada en un vector latente de 48 dimensiones, que representa las características más relevantes de la imagen histológica en un espacio comprimido. Estos vectores latentes, junto con su etiqueta de clase correspondiente (por ejemplo, benigno o maligno), se almacenan en un archivo **JSON** que actúa como base de datos para el sistema de clasificación.

Para realizar la clasificación de una nueva imagen, se sigue el siguiente procedimiento:

1. **Extracción del vector latente:** La imagen de entrada se procesa a través del codificador previamente entrenado, extrayendo su correspondiente vector de características.

2. **Cálculo de distancias:** Se calcula la **distancia euclidiana** entre este vector y todos los vectores almacenados en el conjunto indexado. La distancia euclídea entre dos vectores x e y se define como:

$$d(x, y) = \sqrt{\sum_{i=1}(x_i - y_i)^2} \tag{2}$$

3. **Selección de los vecinos más cercanos:** Una vez calculadas todas las distancias, se seleccionan los **k vectores más cercanos**, siendo k= 5 en esta implementación. Estos representan las imágenes más similares en el espacio latente.

4. **Clasificación por mayoría:** Se toma la clase más común entre los cinco vectores más próximos, y esta se asigna como la **predicción final** para la imagen de entrada.

Este enfoque no requiere un entrenamiento explícito del clasificador, ya que el método k-NN funciona de forma **perezosa** (lazy learning) y opera directamente sobre las instancias almacenadas. Gracias a la alta calidad de las características extraídas por el autoencoder, el sistema logra clasificaciones precisas sin necesidad de arquitecturas más complejas.

3.5. Evaluación del Sistema

Los modelos fueron evaluados mediante un conjunto de métricas de clasificación binaria, entendiendo que los benignos serán los casos negativos y los malignos los casos positivos:

1. **Precisión (accuracy):** mide la proporción de predicciones correctas respecto al total de muestras evaluadas. Se calcula como:

$$\text{accuracy} = \frac{VP + VN}{VP + VN + FP + FN} \tag{3}$$

2. **Sensibilidad (recall):** evalúa la capacidad del modelo para identificar correctamente las muestras positivas. Se calcula como:

$$\text{recall} = \frac{VP}{VP + FN} \tag{4}$$

3. **Precisión (precision):** cuantifica la proporción de muestras clasificadas como positivas que realmente lo son. Se calcula como:

$$\text{precision} = \frac{VP}{VP + FP} \tag{5}$$

4. **F1 Score:** proporciona una medida balanceada entre la precisión y la sensibilidad, siendo especialmente útil en contextos donde existe un desequilibrio entre clases. Se calcula como:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

Estas métricas fueron calculadas de forma individual para cada uno de los modelos entrenados, lo que permitió realizar un análisis cuantitativo del impacto que tiene el tipo de arquitectura como *K-Nearest Neighbors* (KNN) y *Convolutional Neural Networks* (CNN) sobre el rendimiento en la tarea de clasificación.

Adicionalmente, se evaluó la influencia de distintos niveles de *data augmentation* aplicados a las imágenes de entrada, tanto para KNN como con CNN con el fin de observar cómo afectan la capacidad de generalización de los modelos.

Como parte del trabajo, se llevó a cabo una experimentación adicional orientada a evaluar el impacto de una integración más profunda entre los enfoques basados en KNN y CNN. En esta etapa, se creó un sistema de almacenamiento que registró todas las representaciones extraídas de las imágenes mediante el modelo KNN, sin restricciones en cuanto a la dimensión o amplitud de dichas características. Paralelamente, se entrenó un modelo CNN utilizando la totalidad del conjunto de imágenes disponibles, y se extrajeron los pesos resultantes tras el entrenamiento completo del modelo. Esta experimentación se realizó con el fin de observar qué resultados podían obtenerse al disponer de ambas fuentes de representación, y explorar su posible utilidad en tareas de clasificación médica.

4. Experimentación

Para llevar a cabo la experimentación, se realizaron pruebas sobre un conjunto de datos al que se le aplicará *data augmentation* con el objetivo de evaluar la robustez y capacidad de generalización de los modelos. Las técnicas de aumento de datos incluirán transformaciones como rotaciones de un rango entre $-30°$ a $30°$ y modificaciones en la intensidad de las imágenes, con un factor de entre 0,7 (oscurece) a 1,3 (ilumina), simulando variaciones realistas que podrían encontrarse en escenarios del mundo real. Esto permitirá analizar cómo afecta la variabilidad de entrada al rendimiento de los modelos evaluados.

En esta etapa, se llevará a cabo una comparación de rendimiento entre dos enfoques de clasificación: una red neuronal convolucional (*CNN*) entrenada sobre cada variedad de las imágenes aumentadas y un entrenamiento con todas las imágenes en global, y un modelo *K-Nearest Neighbors* (*KNN*) diseñado de forma personalizada para cada uno de las posibles variaciones de imágenes respecto a su aumento. Esta comparación permitirá observar las diferencias entre un enfoque basado en aprendizaje profundo y otro basado en métodos clásicos de aprendizaje supervisado, en cuanto a métricas de *precisión*, *recall* y *F1-score*.

A continuación se muestran los resultados de las pruebas realizadas para entender de mejor manera la robustez y adaptabilidad de los distintos modelos

5. Manual de usuario

Esta herramienta permite cargar una imagen desde el dispositivo y realizar un análisis para determinar si corresponde a un caso **benigno** o **maligno**. Los pasos a seguir son los siguientes:

Paso 1: Seleccionar una Imagen

Opción 1: Cargar por Ruta

- Haz clic en el botón **Seleccionar Imágenes**.
- Navega por tus carpetas y selecciona las imágenes que deseas analizar.
- Las imágenes seleccionadas aparecerán en el panel de previsualización.

Opción 2: Arrastrar y Soltar

- Puedes arrastrar la imagen directamente desde tu explorador de archivos y soltarla en el área indicada de la aplicación.
- Las imágenes se cargarán automáticamente en la aplicación.

Paso 2: Iniciar el Análisis

- Una vez que hayas cargado la imagen correctamente, haz clic en el botón **Analizar**.
- La aplicación procesará la imagen y mostrará si corresponde a un caso maligno o benigno.

Paso 3: Ver el Resultado

- Después de un breve momento, se mostrará el resultado en pantalla.
- Si la imagen es **benigna**, se mostrará un mensaje y un círculo indicador en color verde.
- Si la imagen es **maligna**, se mostrará un mensaje y un círculo indicador en color rojo.

Paso 4: Realizar Nuevos Análisis

- Puedes cargar otras imágenes haciendo clic nuevamente en el botón **Seleccionar** o arrastrándolas directamente.
- El proceso de análisis es el mismo para cada imagen que cargues.

6. Conclusiones

Aquí van las conclusiones.

7. Autoevaluación de cada miembro del equipo

Criterio	**Lucas Manuel Herencia Solís**	**Marina Calero López**	**Juan Antonio Moreno Moguel**
Comprensión y dominio	Excelente		
Exposición didáctica	Excelente		
Integración del equipo	Excelente		
Objetivos	Excelente		
Aspectos didácticos	Excelente		
Experimentación y conclusiones	Excelente		
Contenidos	Excelente		
Divulgación de contenidos	Excelente		
Bibliografía / Recursos científicos	Excelente		

8. Tabla de tiempos

Aquí van las conclusiones.

Referencias

[1] World Health Organization, «Breast Cancer,» 2025, Accessed: 2025-04-30.

[2] P. P. Shinde y S. Shah, «A review of machine learning and deep learning applications,» págs. 1-6, 2018.

[3] D. Bardou, K. Zhang y S. M. Ahmad, «Classification of breast cancer based on histology images using convolutional neural networks,» *Ieee Access*, vol. 6, págs. 24 680-24 693, 2018.

[4] DataCamp, «Capas convoluciones: CNN con TensorFlow en Python,» 2025, Accessed: 2025-04-30.

[5] Python Software Foundation, «Welcome to Python.org,» 2023, Accessed: 2023-11-15. dirección: `https://www.python.org/`.

[6] Python Software Foundation, «Tkinter — Python interface to Tcl/Tk,» es, nov. de 2023, Versión 3.x, consultado el 2023-11-15. dirección: `https://docs.python.org/es/3/library/tkinter.html`.

[7] Google Brain Team and TensorFlow contributors, «TensorFlow: An end-to-end open source platform for machine learning,» 2023, Accessed: 2023-11-15. dirección: `https://www.tensorflow.org/`.

[8] Y. LeCun, L. Bottou, Y. Bengio y P. Haffner, «Gradient-based learning applied to document recognition,» *Proceedings of the IEEE*, vol. 86, n.º 11, págs. 2278-2324, 1998.

[9] V. Nair y G. E. Hinton, «Rectified linear units improve restricted boltzmann machines,» págs. 807-814, 2010.

[10] A. L. Maas, A. Y. Hannun, A. Y. Ng et al., «Rectifier nonlinearities improve neural network acoustic models,» vol. 30, n.º 1, pág. 3, 2013.

[11] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus e Y. LeCun, «Overfeat: Integrated recognition, localization and detection using convolutional networks,» *arXiv preprint arXiv:1312.6229*, 2013.

[12] C. M. Bishop y N. M. Nasrabadi, «Pattern recognition and machine learning,» vol. 4, n.º 4, 2006.

[13] D. P. Kingma y J. Ba, «Adam: A method for stochastic optimization,» *arXiv preprint arXiv:1412.6980*, 2014.

[14] D. E. Rumelhart, G. E. Hinton y R. J. Williams, «Learning representations by back-propagating errors,» *nature*, vol. 323, n.º 6088, págs. 533-536, 1986.

[15] D. Bank, N. Koenigstein y R. Giryes, «Autoencoders,» *Machine learning for data science handbook: data mining and knowledge discovery handbook*, págs. 353-374, 2023.

[16] J. Masci, U. Meier, D. Cireşan y J. Schmidhuber, «Stacked convolutional auto-encoders for hierarchical feature extraction,» págs. 52-59, 2011.

[17] G. E. Hinton y R. R. Salakhutdinov, «Reducing the dimensionality of data with neural networks,» *science*, vol. 313, n.º 5786, págs. 504-507, 2006.

[18] I. Goodfellow, Y. Bengio, A. Courville e Y. Bengio, «Deep learning,» vol. 1, n.º 2, 2016.

[19] T. Cover y P. Hart, «Nearest neighbor pattern classification,» *IEEE transactions on information theory*, vol. 13, n.º 1, págs. 21-27, 1967.

[20] K. Chomboon, P. Chujai, P. Teerarassamee, K. Kerdprasop y N. Kerdprasop, «An empirical study of distance metrics for k-nearest neighbor algorithm,» vol. 2, pág. 4, 2015.

[21] L. Van Der Maaten, E. O. Postma, H. J. Van Den Herik et al., «Dimensionality reduction: A comparative review,» *Journal of machine learning research*, vol. 10, n.º 66-71, pág. 13, 2009.

[22] Y. Bengio, A. Courville y P. Vincent, «Representation learning: A review and new perspectives,» *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, n.º 8, págs. 1798-1828, 2013.

[23] C. C. Aggarwal y C. K. Reddy, «Data clustering,» *Algorithms and applications. Chapman&Hall/CRC Data mining and Knowledge Discovery series, Londra*, 2014.

[24] Google Drive, «Carpeta compartida de recursos,» 2025, Accessed: 2025-04-30.

[25] F. A. Spanhol, L. S. Oliveira, C. Petitjean y L. Heutte, «A dataset for breast cancer histopathological image classification,» *Ieee transactions on biomedical engineering*, vol. 63, n.º 7, págs. 1455-1462, 2015.

[26] A. E. Minarno, K. M. Ghufron, T. S. Sabrila, L. Husniah y F. D. S. Sumadi, «Cnn based autoencoder application in breast cancer image retrieval,» págs. 29-34, 2021.