Clasificación de Obras de Arte por Autor con Diversas Tipologías de Redes Neuronales

UNIVERSIDAD
COMPLUTENSE
MADRID

Gradiente Explosivo
Carlos José Benítez Navas
Elliot Díaz Fernández
Alvaro González González
Patricia Maroto Martín

Contents

1 Introducción

Un gran uso de las redes neuronales es la clasificación de objetos. En este trabajo se van a implementar distintas redes para evaluar la eficiencia de cada una para la clasificación de obras de arte por su autor.

La IA es una tecnología que evoluciona a gran velocidad; tanto es así, que un artículo publicado puede quedar obsoleto en cuestión de meses. Incluso es posible que aún no existan documentos que recojan ciertos desarrollos recientes. La eficiencia de los métodos ya existentes (e incluso de enfoques emergentes como las redes KAN) mejora día a día. Tecnologías que se consideraban lentas o ineficientes hace apenas unos años, hoy ofrecen resultados sorprendentes, alcanzando niveles de rendimiento comparables a las soluciones utilizadas comercialmente.

Hay gran interés en estudiar este fenómeno, en particular con la clasificación de obras de arte por autor. La base de datos utilizada en este trabajo proviene de una famosa página web llamada WikiArt, la cual tiene una gran colección de obras de arte de cientos de autores de todas las épocas de la historia. Se va a realizar este estudio con obras de la corriente artística del impresionismo de la cual se toman 4 artistas muy famosos: Pierre Auguste Renoir, Claude Monet, Camille Pissarro y Edgar Degas.

Es de suma importancia recalcar que las obras de esta corriente artística comparten muchas similitudes, incluso siendo de artistas distintos. Es por esto que, si los resultados de estos métodos de clasificación son positivos, refleja la gran calidad y potencial que tienen estas tecnologías.

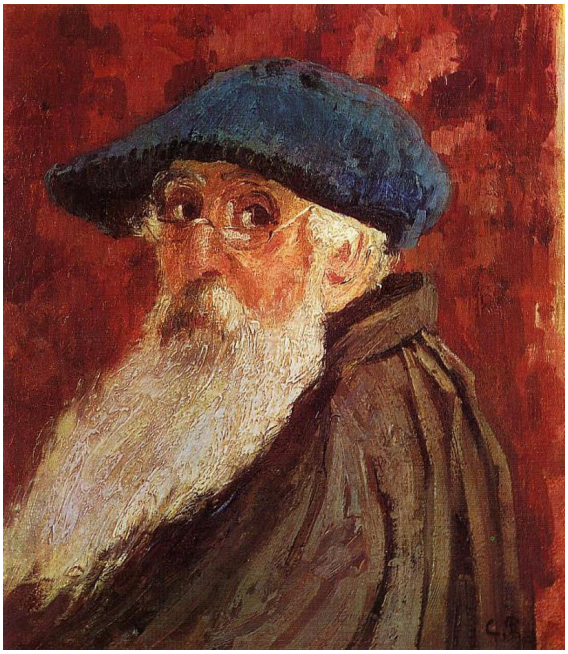
Para este estudio se van a realizar distintas redes neuronales convolucionales (CNN) con tecnologías diversas. La primera es una muy famosa, creada por Google, y recibe el nombre de EfficientNet. De esta se han obtenido resultados muy prometedores. Luego esta se compara con una CNN creada por los integrantes de este grupo que, aunque sus resultados son peores, también logra una clasificación eficiente. Para estas dos redes se utiliza una arquitectura MLP.

A modo de comparación, también se va a desarrollar otra CNN con una red KAN implementada. Hace tan solo unos años se tachaban de ineficientes; no obstante, gracias al estudio de estas y su gran potencial para ciertos problemas, cada vez adquieran más popularidad. Los resultados también son prometedores. Por último, por curiosidad, se desarrolló un Autoencoder con una MLP para ver si este tipo de tecnologías pudieran funcionar para la clasificación de artistas, pero los resultados no fueron buenos.

Para poder visualizar el funcionamiento de las CNNs programadas, se realizó una aplicación móvil con las tecnologías de EfficientNet que fue la que dió mejores resultados. Esta permite tomar una foto a un cuadro de alguno de los cuatro autores y predecir de muy buena manera a quién le pertenece.

Pierre Auguste Renoir

Claude Monet



Camille Pissarro

Edgar Degas

2 Preprocesado de Datos

Antes de pasar la base de datos por las redes de clasificación, es muy importante preprocesar las imágenes de manera correcta. Como se ha explicado en la introducción, se toman imágenes de obras de arte de 4 autores del impresionismo de las bases de datos de WikiArt, obteniendose alrededor de 4000 imágenes.

El primer problema a resolver es que estas imágenes son de dimensiones distintas; es decir, se tienen obras que son cuadradas y otras más rectangulares. Para el correcto estudio, todas deben de tener las mismas dimensiones. No obstante, no bastaría con estirarlas o encogerlas, ya que esto deformaría las obras y podría causar resultados incorrectos. En su lugar, se utiliza el llamado "Padding". Esta es una técnica usada en redes neuronales para añadir valores (normalmente ceros) alrededor de los datos de entrada, especialmente en imágenes o secuencias. Su objetivo principal es controlar cómo cambian las dimensiones de los datos al pasar por las capas. En el caso de las imáganes de la base de datos lo que se hace es añadirle bloques negros a los lados o por encima y abajo (depende si la imagen es vertical u horizontal) para convertirla en una imágen cuadrada.

Una vez que todas las imágenes son cuadradas, estas deben tener la misma resolución. Es por esto que se comprimen a una dimensión de 224×224. Aunque las redes permiten la entrada de imágenes con dimensiones mayores, los resultados obtenidos con esta resolución son bastante satisfactorios y aumentar la resolución causaría tiempos de entrenamiento muy altos.

3 EfficientNet

3.1 Conceptos Básicos

En 2019, Google saca EfficientNet. Este es un modelo desarrollado para la clasificación de objetos, en especial la clasificación de imágenes. El enfoque de estas redes es diferente a lo que se estaba estudiando en el momento. En vez de aumentar las capas para poder obtener una mejor precisión (lo cual hacía las redes muy lentas para tampoco obtener mucha mejoría), usaron un método llamado escalado compuesto donde se aumenta la profundidad, la anchura y la resolución de las imágenes de forma equilibrada; es decir, en vez de aumentar solamente una de ellas, estas se van ampliando de manera proporcional. El resultado final es un modelo más pequeño, rápido y efeciente.

La arquitectura de la CNN de EfficientNet funciona con bloques. Esta procesa las imágenes a través de ellos, los cuales son compuestos a su vez por módulos más pequeños. La cantidad de bloques y modulos varía según el modelo y complejidad de la red de EfficientNet. En este trabajo se usa el modelo de EfficientNet-B0, que es de los más sencillos.

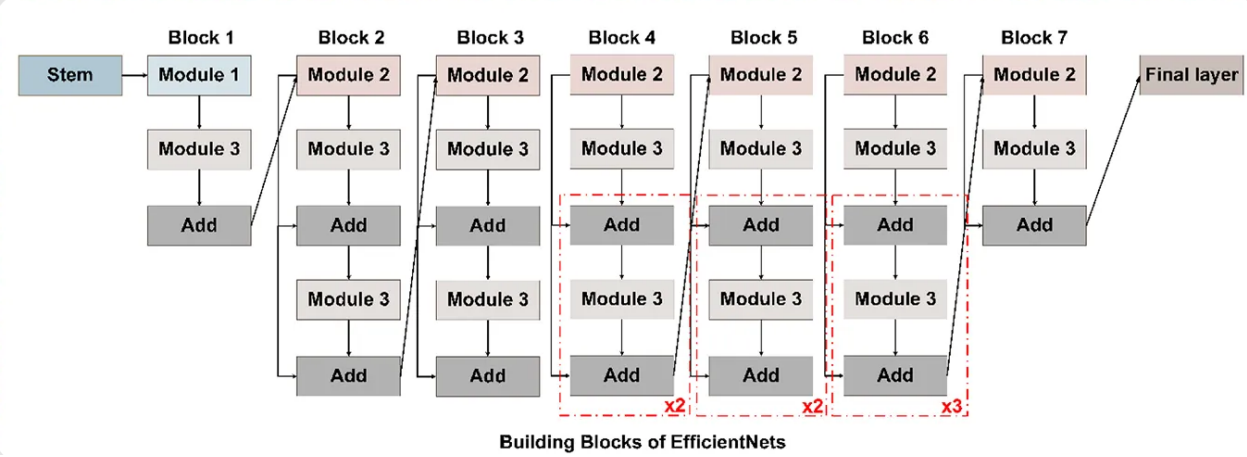


Figure 2: Arquitectura de EfficientNet

Escalado Compuesto: El método de escalado compuesto amplía la profundidad, el ancho y la resolución de la imagen de un modelo, pero los mantiene en equilibrio. Google empezó con el modelo más sencillo (el implementado en este trabajo, como ya hemos comentado antes) llamado EfficientNet-B0, para luego ir aumentando la cantidad de canales, capas y la resolución de las imágenes (EfficientNet-B0 funciona mejor con 224×224). De ahí salieron modelos como el EfficientNet-B1 hasta EfficientNet-B7.

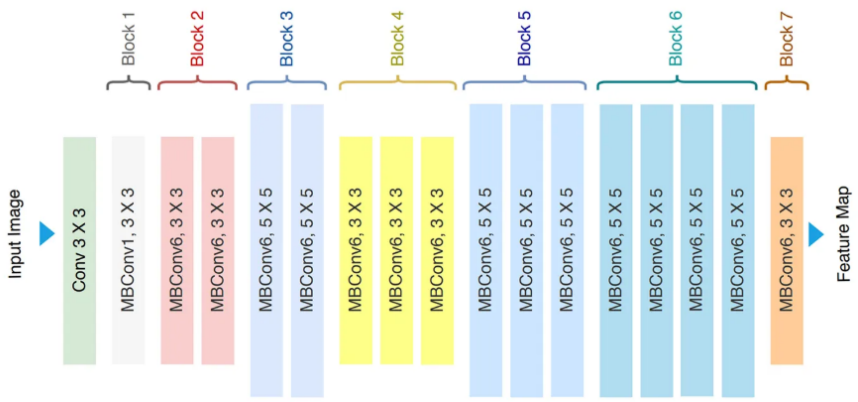


Figure 3: Arquitectura de EfficientNet-B0

En cuanto a la arquitectura de la red, esta se basa en MobileNetV2, que es una red neuronal optimizada para que los dispositivos pequeños puedan reconocer imágenes de forma rápida y eficiente, sin perder demasiada precisión. EfficientNet utiliza una función de activación llamada Swish, la cual es más suave que ReLU. Puede ayudar a que la red aprenda mejor y optimiza el rendimiento en algunos modelos.

De manera simplificada, el funcionamiento de la red utilizada en este trabajo sigue la siguiente estructura:

- Imagen entra
- Convoluciones eficientes extraen características
- Bloques repetidos
- Capa final (MLP)
- Clasificación

3.2 Implementación

Los resultados de EfficientNet-B0 fueron muy fructíferos. Para programar esta red se hizo uso de Python, ya que tiene implementada la librería timm donde se puede hacer uso de las distintas versiones de EfficientNet. Este programa empieza leyendo la base de datos y se asegura de que las imágenes cumplan las características pedidas (tamaño 224×224, lo convierte a números y lo normaliza para mejor entrenamiento). Luego, como la base de datos no tiene balance entre la cantidad de pinturas por artísta, se toma de manera aleatoria 250 pinturas por cada autor. Se hace esto también para disminuir un poco la cantidad de datos a disposición (4000 imágenes).

De las 1000 pinturas a estudiar se dividen en dos grupos de "train" y "test" con la primera teniendo el 80% de los datos para luego poder sacar resultados estadísticos. Luego, se carga el modelo

EfficientNet-B0, se configuran la cantidad de clasificadores, la función de pérdida y se realiza el entrenamiendo con 10 etapas y un learning rate de 0.0001.

Una vez terminado el entrenamiento, se calculan métricas de gran interés con los datos test como el accuracy, precisión y la matriz de confusión. Este modelo se guarda y se hace otro documento donde este se lee y al añadir una imágen te da las probabilidades de que sea de cada autor.

3.3 Resultados

El programa se ejecutó al principio con 250 imágenes por artista, de los cuales obtenemos un accuracy de casi el 90%.

Accuracy: 0.895

Clase	Precision	Recall	F1-score	Support
camille-pissarro	0.87	0.91	0.89	57
claude-monet	0.90	0.93	0.91	46
edgar-degas	0.97	0.85	0.91	46
pierre-auguste-renoir	0.87	0.88	0.87	51

Matriz de Confusión

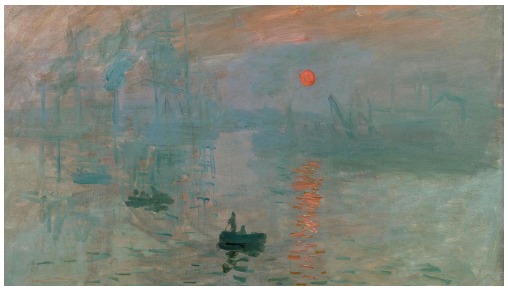
$$\begin{bmatrix} 52 & 3 & 1 & 1 \\ 2 & 43 & 0 & 1 \\ 2 & 0 & 39 & 5 \\ 4 & 2 & 0 & 45 \end{bmatrix}$$

(Las filas son los artistas Reales, las columnas las Predicciones)

Además, al ir pasandole al programa imágenes nuevas de estos mismos artistas, es capaz de predecir con bastante precisión de quién es la pintura.

Camille Pissarro
camille-pissarro: 80.87%
claude-monet: 13.64%
edgar-degas: 0.79%
pierre-auguste-renoir: 4.70%

Edgar Degas
camille-pissarro: 2.31%
claude-monet: 11.07%
edgar-degas: 79.89%
pierre-auguste-renoir: 6.72%



Claude Monet
camille-pissarro: 27.06%
claude-monet: 59.33%
edgar-degas: 12.71%
pierre-auguste-renoir: 0.90%

Pierre Auguste Renoir
camille-pissarro: 3.69%
claude-monet: 2.14%
edgar-degas: 9.59%
pierre-auguste-renoir: 84.57%

Luego, se hizo otro modelo con 400 imágenes por artista para ver si los resultados cambiaban. Mejoró un poco el accuracy, pero tampoco hay una diferencia significatica.

Accuracy: 0.903

Clase	Precision	Recall	F1-score	Support
camille-pissarro	0.91	0.87	0.89	82
claude-monet	0.85	0.93	0.89	67
edgar-degas	0.92	0.92	0.92	85
pierre-auguste-renoir	0.93	0.91	0.92	86

Matriz de Confusión

$$\begin{bmatrix} 71 & 5 & 3 & 3 \\ 3 & 62 & 1 & 1 \\ 2 & 3 & 78 & 2 \\ 2 & 3 & 3 & 78 \end{bmatrix}$$

(Las filas son los artistas Reales, las columnas las Predicciones)

4 RED CNN-MLP

Tras ver los resultados que podemos obtener con la EfficientNet de Google, ahora vamos a iniciar un proceso para ver qué resultados podemos obtener con redes implementadas por nostros mismos. Lo más básico que podemos pensar hacer es una red convolucional elemental unida a una red de clasificación multicapa.

4.1 Implementación

Lo primero que realizamos es la estandarización previamente citada, es decir, normalización de los datos, padding y reescalado a dimensiones 224*224*3, ya que tenemos una imagen con tres canales de color correspondientes al estandar RGB. La estructura de la red es la siguiente: 5 capas de convolución en las que se dividen las dimensiones a la mitad en cada capa y se aplica una activación ReLu para añadir no linealidades a la red, además, se aplia un drop-out de infromación para intentar evitar sobre ajustes y mal aprendizaje. Tras ello, se ejecuta una red MLP para la clasificación de los artistas.

Con el objetivo de evitar sesgos durante el entrenamiento, se realizó un balanceo del conjunto de datos, seleccionando el mismo número de imágenes para cada artista. Posteriormente, el conjunto total se dividió en dos subconjuntos de imagenes, el conjunto train y test, que albergan el 80% y 20% de los cuadros, manteniendo la proporción de clases.

La red fue entrenada durante 20 épocas utilizando 400 imagenes por artista y utilizando como paso de optimización un paso Adam con tasa de aprendizaje de 0.001

4.2 Resultados

Matriz de Confusión

$$\begin{bmatrix} 54 & 10 & 10 & 6 \\ 7 & 53 & 11 & 9 \\ 6 & 4 & 47 & 23 \\ 4 & 4 & 11 & 61 \end{bmatrix}$$

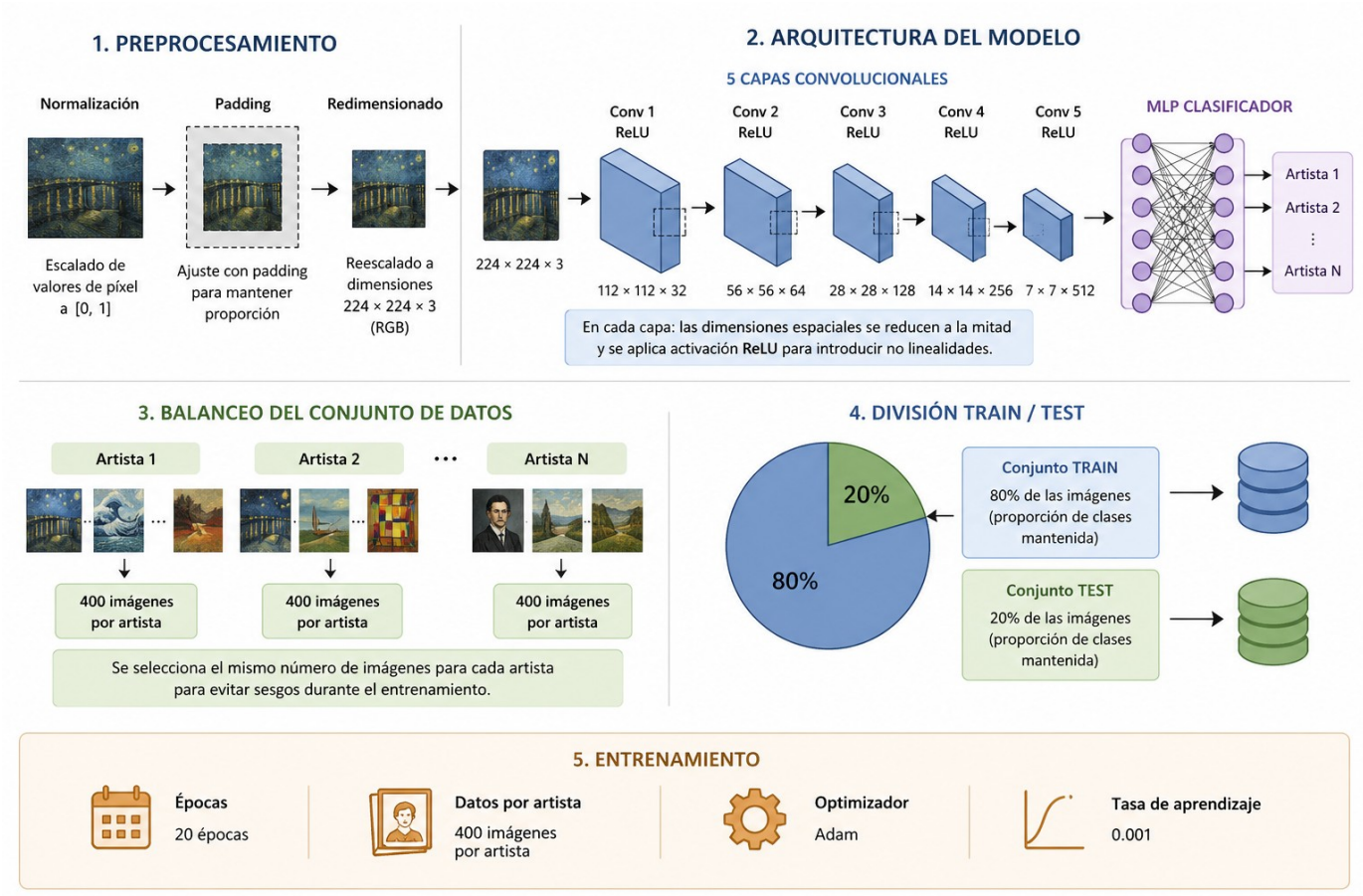
(Las filas son los artistas Reales, las columnas las Predicciones)

Accuracy: 67.19%

Clase	Precision	Recall	F1-score	Support
camille-pissarro	0.76	0.68	0.72	80
claude-monet	0.75	0.66	0.70	80
edgar-degas	0.59	0.59	0.59	80
pierre-auguste-renoir	0.62	0.76	0.68	80

Para ser transparentes debemos decir que estas métricas pertenencen a una buena ejecución, pues debido a la aleatoriedad del proceso ha habido entrenamientos en los que se ha obtenido malas métricas en el train pero buenas métricas en el test y viceversa, así como ejecuciones que ha funcionado todo mal y veces, como la mostrada, en la que las métricas son relativamente buenas. Recordar que para modelos de clasificación multiclase, en nuestro caso con 4 posibles respuestas, un accuracy mayor al 25% es mejor que la predicción aleatoria. Para entender lo que quiere destacar este párrafo, se ejecuto una simulación Montecarlo del entrenamiento de la red CNN con 10 iteraciones la cual tomo cerca de 8 horas de procesado y el accuracy medio fue de 48.67% debido a esa inestabilidad comentada.

En la sigueinte imagen se presenta un esquema resumen del proceso llevado a cabo en esta red.



Estructura de la red CNN + MLP

5 Red CNN-KAN

El objetivo de implementar esta red KAN es poder comparar con la red MLP descrita en el anterior apartado. En una estructura MLP tradicional, las activaciones se aplican en las neuronas, mientras que las aristas contienen pesos escalares lineales. Por otro lado, una red KAN propone que las funciones de activación sean aprendibles y se sitúen en las aristas. En lugar de optimizar una matriz de pesos W, la red optimiza una serie de funciones univariadas $\phi_{i,j}$. Estas funciones, conocidas como funciones de activación pueden ser tangentes hiperbólicas, sigmoides o funciones SiLu. A través de estas funciones se permite que la red capture relaciones no lineales con un número relativamente menor de parámetros que las redes MLP tradicionales.

5.1 Implementación

Se han llevado a cabo varias redes KAN obtienendo diversos resultados en cada una de ellas. Lo primero que hicimos en todas ellas fue reducir la dimensionalidad de las imágenes para que la memoria RAM del ordenador pudiese trabajar con todas las imágenes.

La primera de las redes KAN fue lo más sencilla posible. La función de activación que utilizamos fue una sigmoide y la dimensionalidad de entrada a la red KAN es de d = 8192. Dado que cada conexión en una red KAN implica la optimización de un polinomio de grado n, esta configuración introduce una carga computacional elevada y riesgos de estabilidad numérica. De hecho, cuando se ejecutó esta red se obtuvieron resultados inferiores al 50 % de accuracy. La primera mejora que se le aplicó a la red fue cambiar la función de activación por la tangente hiperbólica. Aunque los resultados no mejoraron drásticamente, se consiguió un accuracy del 60 %, que aunque sigue siendo bajo, mejoró bastante lo anterior.

Posteriormente, se pensó que el problema era que llegaban vectores de dimensión demasiado grande a la red KAN por lo que optamos por reducir la dimensionalidad de estas imágenes. Esta reducción de la dimensionalidad se hizo mediente una transformación lineal seguida de una ReLU, lo que transformaba el vector de características de dimensión 8192 a un espacio latente de 256 antes de ingresar en la red KAN.

La última mejora que se introdujo fue en la función de activación. En la última red KAN, en cada una de las capas la salida se calcula como la suma de una base lineal y una base no lineal:

SiLU (Sigmoid Linear Unit): Se aplica sobre la componente de la base lineal de la capa.

$$f(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}}$$

Tanh (Tangente Hiperbólica): Se aplica a la entrada x como paso previo a la realización de la expansión polinómica.

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

5.2 Resultados

Una vez entendida la implementación de esta red KAN, procedemos a evaluar los resultados obtenidos a través de ella. Para ello se utilizo un conjunto de test que consta del 20 % de la imágenes que se tienen. Cada artista cuenta con el siguiente número de imágenes.

- Camile Pisarro: 170 imágenes
- Claude Monet: 262 imágenes
- Edgar Degas: 128 imágenes
- Pierre Auguste Renoir: 258 imágenes

Tras la ejecución obtenemos la siguiente matriz de confusión:

$$\begin{bmatrix} 122 & 22 & 8 & 17 \\ 20 & 228 & 2 & 6 \\ 5 & 12 & 80 & 20 \\ 21 & 27 & 12 & 216 \end{bmatrix}$$

(Las filas son los artistas Reales, las columnas las Predicciones)

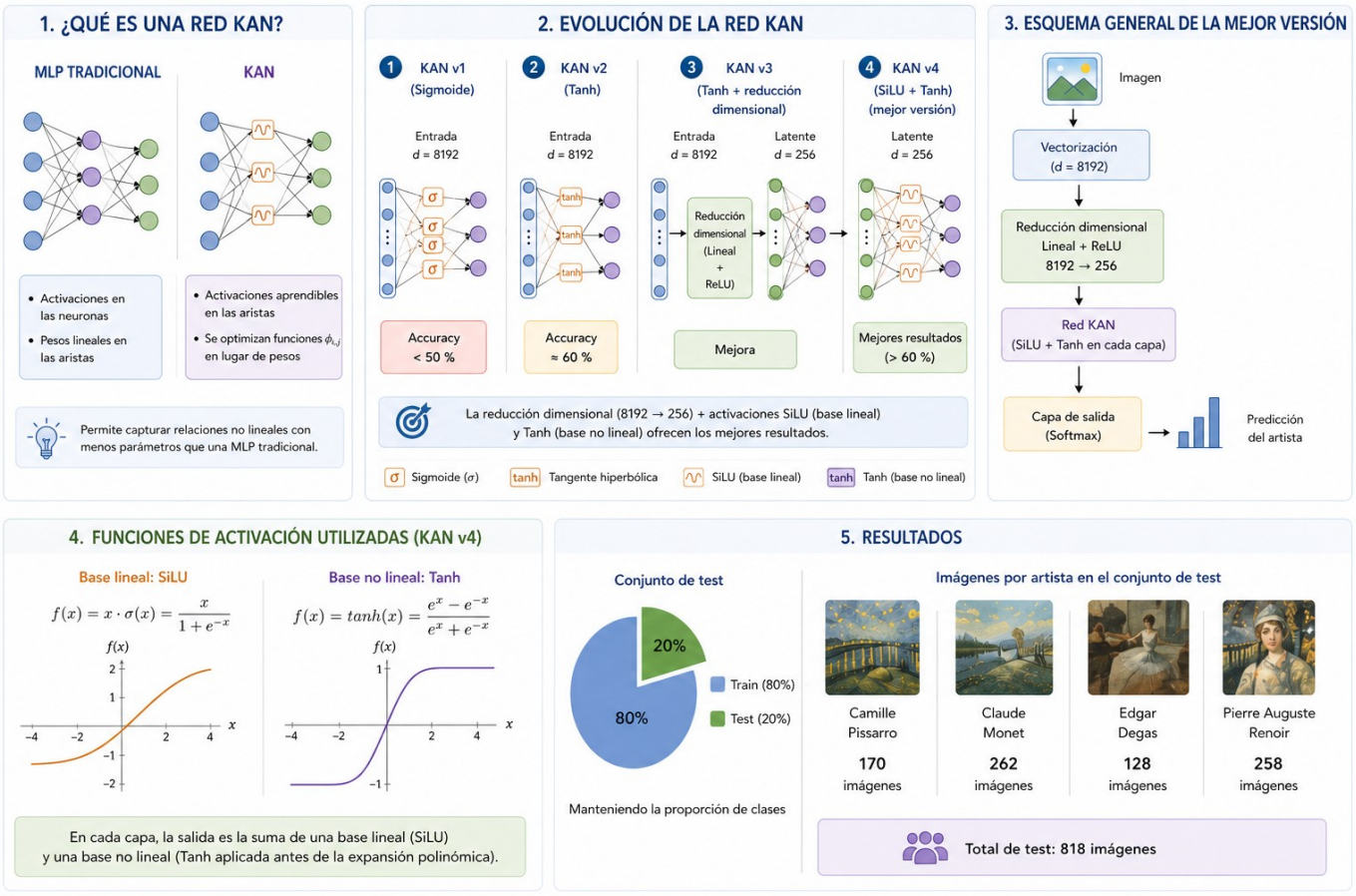
Para el entrenamiento del modelo se utilizó una tasa de aprendizaje 0.01 y un entrenamiento de 20 épocas.

Además de la matriz de confusión podemos observar una tabla que resume los resultados obtenidos con los distintos artistas:

Clase	**Precision**	**Recall**	**F1-score**	**Support**
Camille Pissarro	0.73	0.72	0.72	169
Claude Monet	0.79	0.89	0.84	256
Edgar Degas	0.78	0.68	0.73	117
Pierre-Auguste Renoir	0.83	0.78	0.81	276

Teniendo en cuenta esta tabla, el accuracy final de la CNN con una red KAN es de 78'97% lo que mejora las redes MLP. Sin embargo, no alacanza la precisión de la red de Google.

Para finalizar la sección se añade una imagen resumen para resumir la información de esta red:



Estructura de la red CNN + KAN

6 Red Autoencoder-MLP

6.1 Implementación

El último modelo incorporado a nuestro trabajo es otro enfoque distinto, entrenar un autoencoder para posteriormente proyectar las imágenes sobre el espacio latente y generar una clasificación a través del entrenamiento de un MLP.

Primero se implementa el entrenamiento del perceptrón, el cual posee 4 capas, y se entrena para reducir a un vector de 16 dimensiones la infomación de los cuadros. Esto se realiza como forma de sustituir a la red CNN para intentar valorar un enfoque diferente en la compresión de información.

Una vez entrenado el autoencoder se lleva a cabo el entrenamiento de la red MLP, la idea de esta parte es la siguiente: para una imagen dada de un artista, se proyecta la información de esta sobre el espacio latente generado en el entrenamiento del autoencoder, lo cual nos da un vector de 16 dimensiones. Este vector es el que se utiliza para entrenar la red MLP, por tanto la red MLP lo que buscará será regularidades en diferentes dimensiones del vector para asignar una clasificación de los artistas.

6.2 Resultados

Matriz de Confusión

$$\begin{bmatrix} 58 & 19 & 23 & 20 \\ 23 & 61 & 20 & 16 \\ 9 & 7 & 62 & 42 \\ 19 & 2 & 31 & 68 \end{bmatrix}$$

(Las filas son los artistas Reales, las columnas las Predicciones)

Accuracy: 51,44%

Clase	Precision	Recall	F1-score	Support
camille-pissarro	0.53	0.48	0.51	120
claude-monet	0.69	0.51	0.58	120
edgar-degas	0.46	0.52	0.48	120
pierre-auguste-renoir	0.47	0.57	0.51	120

Se dejó entrenar la MLP durante 100 épocas y los datos no mejoraron, dando un accuracy en el test de 48.56%

Se puede suponer que el mal funcionamiento de la red MLP viene derivada de que espacialmente los puntos proyectados en el espacio latente del autoencoder están severamente juntos como se puede ver en las siguientes imágenes:

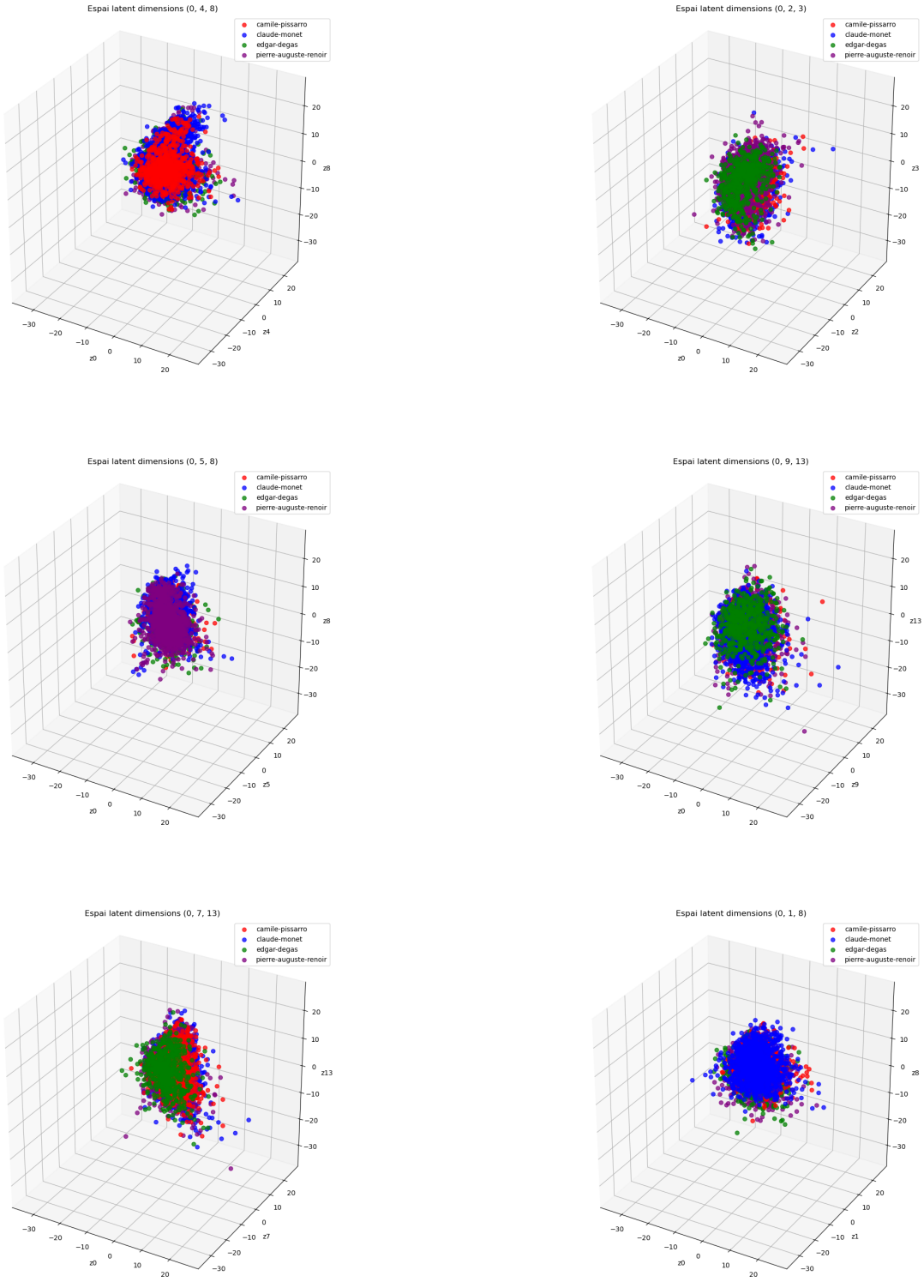
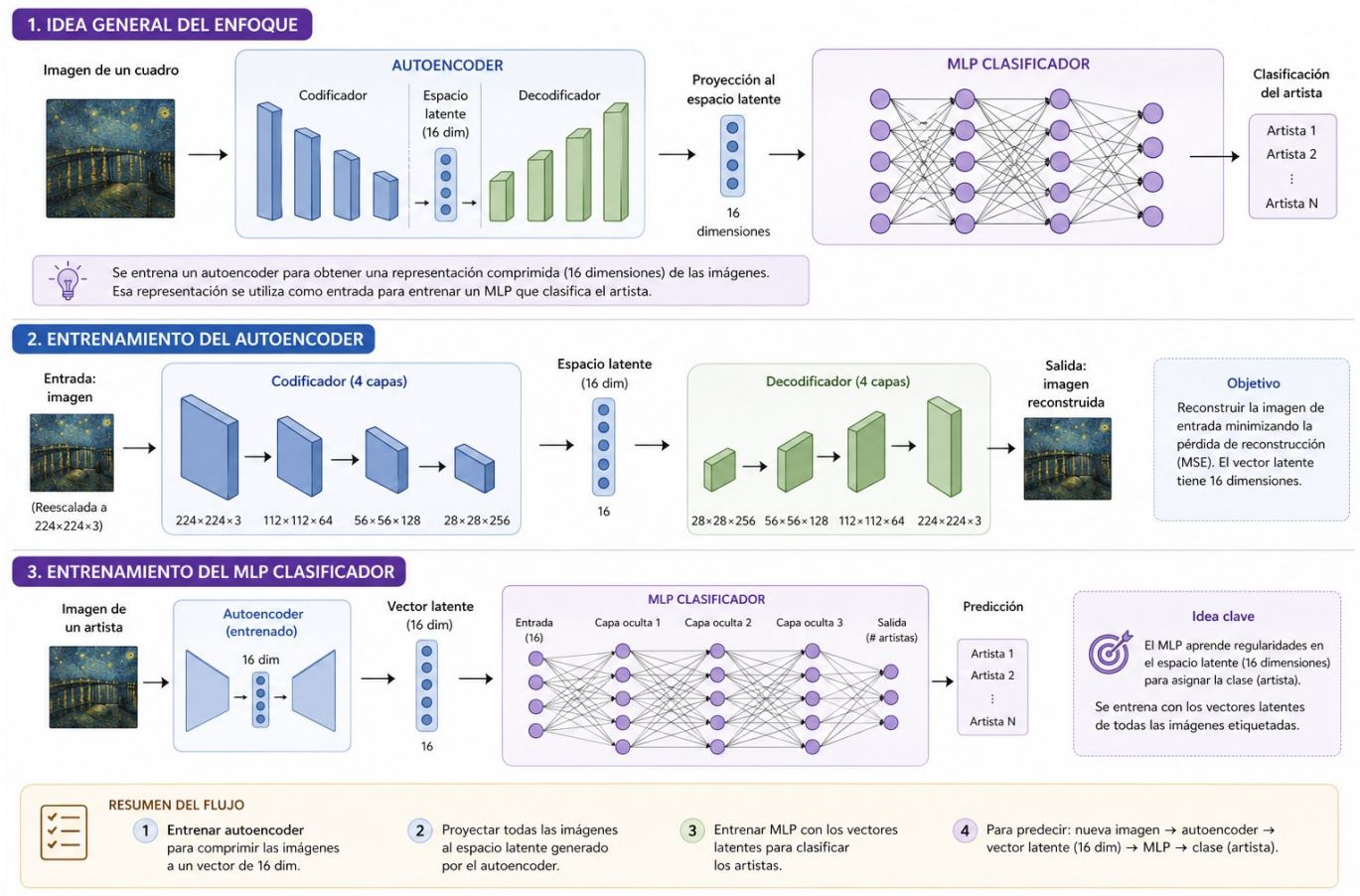


Figure 4: Imagenes 3Dimensionales del Autoencoder

Como se mencionaba, visualmente se ve que observando diferentes vistas de la proyección de los cuadros en el espacio latente del autoencoder todos los puntos de los artistas están muy juntos, lo cual es coherente con que la MLP no sea capaz de encontrar regularidades en las coordenadas para poder clasificar distintos artistas.

Para finalizar la sección se añade una imagen resumen para resumir la información de esta red:



Estructura de la red Autoencoder + MLP

7 Aplicación web y para teléfono

Aunque la utilidad de las IAs de clasificación de obras de arte que hemos desarrollado es clara y notoria, parte de ella se perdería si no fuera accesible y fácil de usar. Es por ello que, tras el desarrollo y evaluación de los distintos modelos, surge la necesidad de trasladar estos resultados a un entorno de uso práctico: un aplicación web y descargable en el teléfono. El enlace de la aplicación es: https://artistas-app-5jrapopvapeziwgw299cg7.streamlit.app/

7.1 Implementación

Para el desarrollo de dicha aplicación, hemos contado con ayuda de otro grupo de compañeros que desarrollaron una aplicación para la detección por foto de hojas de parra enfermas. Para desarrollarla se subieron varios archivos de python con cada parte de la app en Github. Se hizo uso de la platforma Streamlit la cual se conecta directamente a Github para darle una interfaz a este programa. Con eso se obtuvo un enlace el cual puede ser abierto tanto en ordenador como en dispositivos móviles, en estos últimos se puede descargar para tener una aplicación.

Los archivos usados en este proyecto son:

- **modelo_artistas.pth**: Para el desarrollo de nuestra aplicación se usó la CNN de Google EfficientNet B0, ya que fue con la que obtuvimos los mejores resultados con una precisión del 90%. Este modelo, ya entrenado, fue guardado en el archivo modelo_artistas.pth. Este es el que guarda los datos en binario y representa las conexiones de la red neuronal.
- **arquitectura.py**: Luego se definió la arquitectura de la aplicación en el archivo arquitectura.py. Este archivo lo que hace es descargar el cerebro de EfficientNet y cambia la última capa para que se esté ordenando solamente en 4 clases: los 4 artistas (Camille Pissaro, Claude Monet, Edgar Degas y Pierre Auguste Renoir)
- **app.py:** Es la cara del proyecto y el programa principal de la aplicación web. Su trabajo se resume en llamar a arquitectura.py para construir el cerebro de la CNN pero vacío. Luego, carga modelo_artistas.pth y se lo pone a ese cerebro vacío. Posteriormente dibuja la página web (los títulos, botones, y la cámara) usando la librería Streamlit. Cuando tomas una foto, la recorta a 224×224 (igual que en el entrenamiento), se la pasa al cerebro, recibe las probabilidades y pinta el resultado en pantalla (artista predicho y probabilidades de los cuatro).
- **utils.py:** Es como una caja de herramientas donde se guardaron funciones secundarias para no hacer que app.py se viera muy desordenado.
- **requirements.txt:** Los servidores de la nube son como ordenadores recién configurados; tienen Python, pero no incluyen las librerías específicas para IA. Este archivo es una lista de texto que le indica al servidor de Streamlit que, antes de arrancar app.py, debe descargar e instalar PyTorch, OpenCV, Pillow, Numpy y Streamlit. Sin este archivo, la app fallaría porque no encontraría sus herramientas de trabajo.

En resumen el flujo es así: Se utilizó arquitectura.py y CNN.py para generar el conocimiento sobre las obras de arte de nuestros 4 artistas (modelo_artistas.pth). Luego se construyó app.py para conectar ese conocimiento con una cámara y una interfaz web, indicando qué herramientas descargar previamente para tener todas las necesarias para que todo ejecute sin problemas (requirements.txt).

7.2 Resultados

Se han realizado sucesivas prubas con obras de nuestros cuatro artistas (se han tomado fotos a imagenes de sus cuadros obtenidas de Google), y hemos observado que predice el autor correcto la gran mayoría de las veces. Además suele hacerlo con una precisión vertiginosa, esto es, la probabilidad del autor predicho es muy alta (es decir, predice correctamente y no hay duda con los otros 3).

Incluso, a modo de diversión, se puede probar a tomar fotos a otros objetos. Por ejemplo, nuestro grupo probó a fotografiar personas y es curioso que la red suele predecir como "autor del cuadro" a Renoir. De hecho, tiene mucho sentido, ya que de los cuatro artistas es el que más retratos hacía (los utilizaba como principal fuente de ingresos). La red comprende tan bien el estilo de cada uno que, a pesar de que Edgar Degas también pintaba a menudo personas, EfficientNet no confunde su estudio de la figura humana natural (centrado en el cuerpo y el movimiento sin posados) con el estilo de Renoir (que buscaba la armonía y el encanto colocando a sus modelos en posiciones concretas).