



FACULTAD DE CIENCIAS

GRADO EN MATEMÁTICA COMPUTACIONAL Y ANALÍTICA DE DATOS

**ANÁLISIS TOPOLOGICO APLICADO A LA  
COMPRENSIÓN DE LAS REDES NEURONALES  
CONVOLUCIONALES**

*Trabajo de Fin de Grado*

Autor:

Alberto Real Quereda

---

Supervisado por:

Albert Ruiz

Febrero 2024

## Dedicatoria

Quiero agradecer de corazón a todas las personas que han sido parte de este viaje. En especial, agradezco a Albert, mi supervisor, que ha sido fundamental en cada fase de este proyecto. Un especial reconocimiento a Martín, por estar siempre disponible, no solo con el código, sino para cualquier duda que surgiera.

A mis amigos, que han hecho tanto este Trabajo de Final de Grado como el propio grado mucho más llevadero: Cris, Jowy (Joel), Oliva (Marc), Subi (David) y muchos más, gracias por todo.

No puedo olvidarme de mi familia, que ha estado siempre a mi lado. A mis padres, gracias por su apoyo constante en todo momento. Un agradecimiento muy especial para mi hermana y mi cuñado, que han sido pilares increíbles en mi vida, no solo en el ámbito académico, sino en todos los aspectos.

A mi pareja, que ha compartido conmigo las alegrías y los desafíos.

Y, por supuesto, a Roser, mi profesora de matemáticas, que me inspiró a siempre esforzarme más y nunca conformarme con poco.

A todos, gracias de verdad. Sin vosotros, nada de esto habría sido posible.

## Abstract

Este trabajo investiga la evolución y estructura de los filtros en redes neuronales convolucionales (CNN) a través de la aplicación de diversas técnicas de visualización y análisis de datos, incluyendo PCA, UMAP y Vietoris-Rips. Los resultados revelan que los pesos de la CNN tienen una estructura no trivial, lo cual se vuelve más evidente con el aumento de *epochs*. Además, se observó que los filtros incrementan su definición y especificidad con el aumento de *epochs*, mejorando así la precisión del modelo. Aunque PCA, UMAP y Vietoris-Rips proporcionaron perspectivas valiosas sobre la organización de los filtros, Mapper enfrentó limitaciones significativas al visualizar estructuras complejas claramente. Este estudio destaca la importancia de combinar múltiples técnicas para entender completamente las dinámicas internas de las CNN y sugiere direcciones futuras para optimizar el rendimiento y eficacia de estas redes en aplicaciones prácticas.

This work investigates the evolution and structure of filters in convolutional neural networks (CNN) through the application of various visualization and data analysis techniques, including PCA, UMAP, and Vietoris-Rips. The results reveal that the weights of the CNN have a non-trivial structure, which becomes more evident with an increase in *epochs*. Additionally, it was observed that the filters increase in definition and specificity with more *epochs*, thus improving the model's accuracy. Although PCA, UMAP, and Vietoris-Rips provided valuable insights into the organization of the filters, Mapper faced significant limitations in clearly visualizing complex structures. This study highlights the importance of combining multiple techniques to fully understand the internal dynamics of CNNs and suggests future directions for optimizing the performance and efficiency of these networks in practical applications.

# Índice

<b>1. Introducción</b>	<b>4</b>
1.1. Objetivos . . . . .	5
<b>2. Marco teórico</b>	<b>6</b>
2.1. Revisión de la Literatura . . . . .	6
2.2. Conceptos fundamentales . . . . .	7
2.2.1. Topología . . . . .	7
2.2.2. Redes neuronales convolucionales . . . . .	11
2.2.3. Dataset MNIST . . . . .	14
2.2.4. Técnicas de reducción de dimensionalidad . . . . .	15
<b>3. Metodología</b>	<b>21</b>
<b>4. Desarrollo</b>	<b>22</b>
4.1. Preparación de los datos . . . . .	22
4.2. Entrenamiento de las CNN . . . . .	23
4.3. Visualización de los filtros aprendidos . . . . .	23
4.4. Aplicación de técnicas de análisis topológico . . . . .	26
4.4.1. Aplicando PCA . . . . .	26
4.4.2. Aplicando UMAP . . . . .	28
4.4.3. Homología persistente de Vietoris-Rips . . . . .	29
4.4.4. Aplicación del Mapper . . . . .	30
<b>5. Resultados</b>	<b>32</b>
5.1. Tiempos de ejecución y precisión de los modelos . . . . .	32
5.2. Comparación de filtros por etapa de entrenamiento . . . . .	33
5.3. Resultados de las técnicas de análisis topológico aplicadas . . . . .	34
5.3.1. Resultados del PCA . . . . .	34
5.3.2. Resultados del UMAP . . . . .	37
5.3.3. Resultados del Vietoris-Rips . . . . .	41
5.3.4. Resultados Mapper . . . . .	43
<b>6. Conclusiones</b>	<b>46</b>
<b>7. Recomendaciones para futuras investigaciones</b>	<b>49</b>

## 1. Introducción

En las últimas décadas, las redes neuronales convolucionales (CNN) han revolucionado el campo de la visión por computador y el aprendizaje automático, proporcionando avances importantes en tareas como el reconocimiento de imágenes, la detección de objetos y la segmentación semántica. Estos modelos han demostrado ser muy eficaces en el manejo de datos visuales, gracias a su capacidad para capturar información de alto nivel en imágenes, lo que les permite identificar patrones, texturas, formas y objetos dentro de grandes conjuntos de datos visuales.

Sin embargo, a pesar de su éxito, una de las críticas recurrentes a las CNN es la falta de comprensión sobre cómo procesan y toman las decisiones basadas en los datos de entrada, actuando como una caja negra.

Una solución que se ha visto para este problema ha sido el análisis topológico de datos (TDA). El TDA emerge como una herramienta poderosa para explorar la estructura de datos complejos y de alta dimensión. Se centra en estudiar la forma de los datos, permitiendo identificar patrones, características y conexiones que no son evidentes mediante técnicas de análisis tradicionales.

Aplicar el análisis topológico a las redes neuronales convolucionales nos va a permitir observar la estructura interna y el funcionamiento de estos modelos, proporcionando diferentes perspectivas sobre las características que las CNN capturan de los datos y cómo estas influyen en las decisiones finales del modelo.

Este TFG se inspira en las investigaciones recientes sobre la aplicación del TDA en las CNN, particularmente en el estudio de Gabrielsson y Carlsson (2019) [4]. Este estudio demuestra cómo el TDA puede ser utilizado para obtener nuevas perspectivas sobre los espacios de características que las CNN aprenden durante su entrenamiento. La metodología y los hallazgos presentados por Gabrielsson y Carlsson resaltan el potencial del TDA para revelar estructuras topológicas complejas y significativas en los datos, lo cual podría mejorar nuestra comprensión y explicación de los modelos de aprendizaje profundo.

Inspirado por este enfoque, este TFG busca explorar si estos métodos topológicos pueden ser efectivamente aplicados a un conjunto de datos más accesible y con un esquema de entrenamiento considerablemente reducido, utilizando el dataset MNIST.

### **1.1. Objetivos**

El objetivo principal de este Trabajo de Fin de Grado es profundizar en el análisis topológico de datos aplicado a las CNN, específicamente utilizando el dataset MNIST. Se investigará la influencia de la topología de los pesos de las capas de las CNN en sus procesos de aprendizaje y decisiones, explorando si, bajo condiciones de entrenamiento reducidas, las técnicas topológicas conservan su capacidad para ofrecer perspectivas valiosas sobre las características fundamentales de las CNN.

Como objetivos secundarios, este estudio se propone desarrollar e implementar herramientas de visualización topológica adaptadas a las necesidades de análisis de las CNN. Entre estas herramientas se incluyen Mappers 2.2 y diversas técnicas de reducción de dimensionalidad, que permiten una interpretación visual más clara y detallada de la estructura de los pesos dentro de los modelos. Además, se llevará a cabo un análisis exhaustivo de la información topológica contenida en los pesos para evaluar su impacto en la eficacia del modelo y en la distribución espacial de los filtros aprendidos.

Este enfoque espera aportar significativamente al conocimiento existente sobre el análisis topológico y las redes neuronales convolucionales. Al proporcionar una mejor comprensión de cómo las CNN procesan y aprenden de los datos, el proyecto busca facilitar su aplicación en contextos donde la transparencia y la capacidad de explicar el funcionamiento de los modelos son esenciales.

## 2. Marco teórico

### 2.1. Revisión de la Literatura

**Análisis Topológico de Datos (TDA):** La topología es una rama de las matemáticas que estudia las propiedades espaciales de los objetos, como su forma y conectividad. El análisis topológico de datos utiliza conceptos topológicos para analizar un conjunto de datos y descubrir su estructura subyacente [8]. El TDA resulta eficaz para examinar conjuntos de datos complejos y de alta dimensión, permitiendo descubrir patrones y características que no son visibles mediante técnicas de análisis convencionales. Una de las herramientas más destacadas dentro del TDA es la homología persistente, que facilita la comprensión de las características topológicas de los datos a diferentes escalas.

**Redes Neuronales Convolucionales (CNN):** Las CNN son una clase de redes neuronales profundas, estas son particularmente útiles para identificar patrones en imágenes, siendo extremadamente efectivas en tareas de procesamiento de imágenes y visión por computador. Se caracterizan por su capacidad para aprender patrones jerárquicos de características visuales mediante el uso de filtros convolucionales.

**Estudios Previos:** Investigaciones recientes como la de Gabrielsson y Carlsson (2019) [4] han comenzado a explorar la relación entre el Análisis de Datos Topológico y las redes neuronales convolucionales, aplicando técnicas topológicas para interpretar y mejorar nuestra comprensión de estas redes. Este trabajo ha demostrado cómo el análisis topológico puede ofrecer perspectivas valiosas sobre la topología de los espacios de características aprendidos por las CNN.

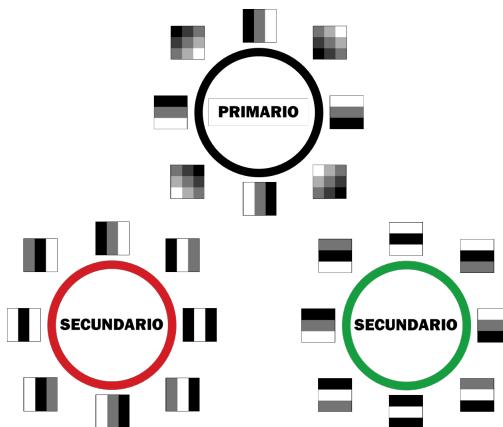


Figura 1: Círculos primario y secundario

En esta investigación, Gabrielsson y Carlsson analizan CNNs entrenadas con varios conjuntos de datos ampliamente utilizados, incluyendo MNIST, CIFAR-10, SVHN e ImageNet. Los autores descubren que, en algunos casos, los modelos logran reproducir estructuras topológicas específicas que

se habían observado en trabajos anteriores, identificadas como los círculos primarios y secundarios 1. Sin embargo, también señalan que en otras situaciones emergen fenómenos distintos, lo que sugiere una variabilidad en la forma en que las CNNs capturan y modelan la topología de los datos.

## 2.2. Conceptos fundamentales

### 2.2.1. Topología

La topología es un área de las matemáticas enfocada en el estudio de las propiedades de los espacios que son invariantes bajo transformaciones continuas, como estiramiento o encogimiento, sin cortar ni pegar dicho espacio. Se centra en comprender la ‘forma’ de los objetos matemáticos, mirando más allá de las características geométricas exactas.

#### Vietoris-Rips

La filtración de Vietoris-Rips es una herramienta muy útil en topología y homología persistente. Se utiliza para analizar la estructura y características topológicas de conjuntos de datos dispersos.

Funciona mediante la construcción gradual de complejos simpliciales, estructuras compuestas por vértices, aristas, triángulos, y otras formas geométricas, a partir de un conjunto de puntos en un espacio métrico, como por ejemplo, el espacio euclíadiano.

La filtración de Vietoris-Rips comienza con un conjunto de puntos en un espacio  $\mathbb{R}^n$

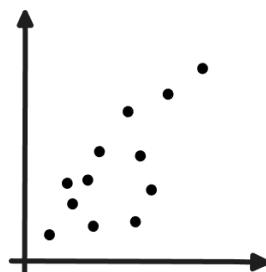


Figura 2: Nube de puntos para la filtración de Vietoris-Rips

Una vez se tienen los datos se establece un parámetro de distancia  $r$ , no negativo, que define el radio dentro del cual se buscan conexiones entre puntos, el cual se irá modificando a medida que se ajusta la filtración.

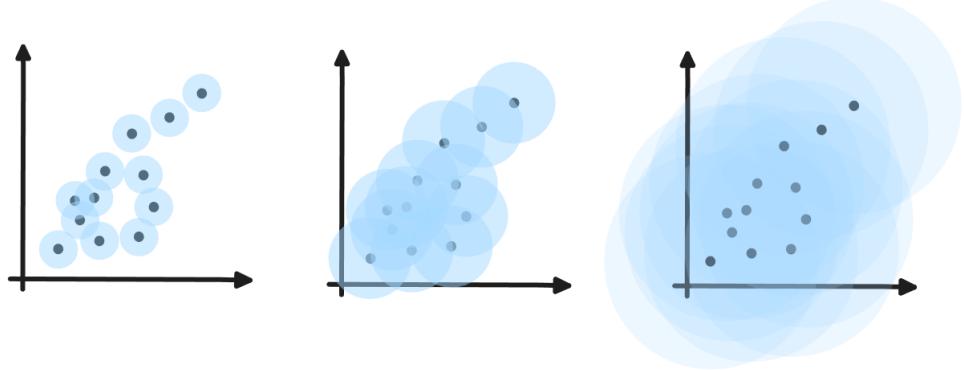


Figura 3: Diferentes  $r$  para la filtración de Vietoris-Rips

Para cada  $r$  se construye un complejo simplicial [11] [9] siguiendo las siguientes reglas:

- **Vértices:** Cada punto es un vértice.
- **Aristas:** Se conectan dos vértices si la distancia entre ellos es menor o igual a  $r$ .
- **Triángulos y tetraedros:** Si un grupo de tres o más puntos están mutuamente conectados por aristas, forman un simplex de mayor dimensión.

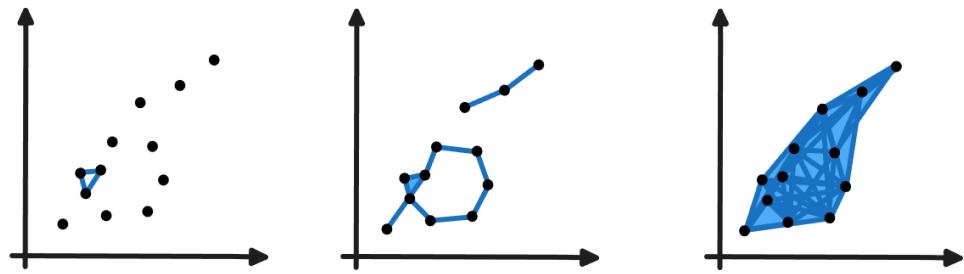


Figura 4: Complejos simpliciales de cada  $r$

Por lo general se observa que al incrementar  $r$  se añaden más conexiones entre los puntos, generando complejos simpliciales más grandes. Hasta que quedan todos los vértices conectados entre ellos. Esto nos permite observar cómo la estructura global de los datos va evolucionando.

La filtración de Vietoris-Rips es especialmente útil en análisis de datos donde las relaciones entre puntos son más significativas que las métricas individuales, ofreciendo información sobre la robustez de las características topológicas frente a cambios en la densidad de conexión.

Este método es fundamental para estudiar la forma y las características de un conjunto de datos y entender la estructura global y las propiedades de forma de los datos en diferentes niveles de conexión.

## Homología persistente

La homología persistente analiza el complejo simplicial formado por cada  $r$  para identificar y rastrear características topológicas, como componentes conexas y agujeros, a través de los diferentes valores de  $r$ . Este análisis proporciona una visión profunda sobre cómo las características topológicas “viven” y cambian conforme el nivel de conexión entre los puntos se altera.

Esto mismo se puede representar en un diagrama de persistencia como el siguiente:

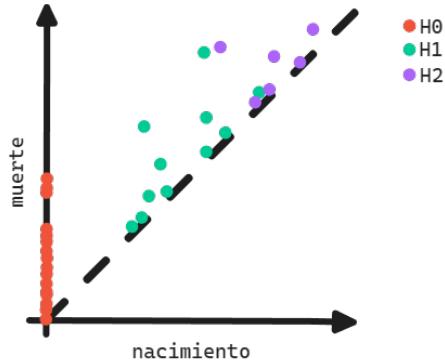


Figura 5: Ejemplo de diagrama de persistencia

Donde en el eje horizontal (nacimiento), se indica el valor del parámetro  $r$  en el que aparece una característica topológica. El eje vertical (muerte), muestra el valor de  $r$  en el que la característica desaparece o se transforma debido a nuevas conexiones.

Los marcadores H0, H1 y H2 del gráfico representan diferentes tipos de características topológicas: H0 para componentes conexas, H1 para agujeros o ciclos de una dimensión, y H2 cavidades.

Un punto en la línea diagonal, donde el nacimiento es igual a la muerte, señala una característica que aparece y desaparece inmediatamente, que se puede clasificar como ruido. Por otro lado, puntos significativamente por encima de la diagonal indican características que persisten a lo largo de un amplio rango de  $r$ , es decir, elementos estructurales significativos del conjunto de datos. [11]

## Mapper

Otra herramienta de topología que se usa en este trabajo es el algoritmo Mapper, una técnica de TDA que crea una representación simplificada de los datos. Se basa en la filtración de los datos a través de una función y luego en la agrupación y visualización de estos datos. Es útil para obtener una visión de alto nivel de la estructura de datos complejos. El método Mapper fue introducido por Singh, Memoli y Carlsson en 2007. [12]

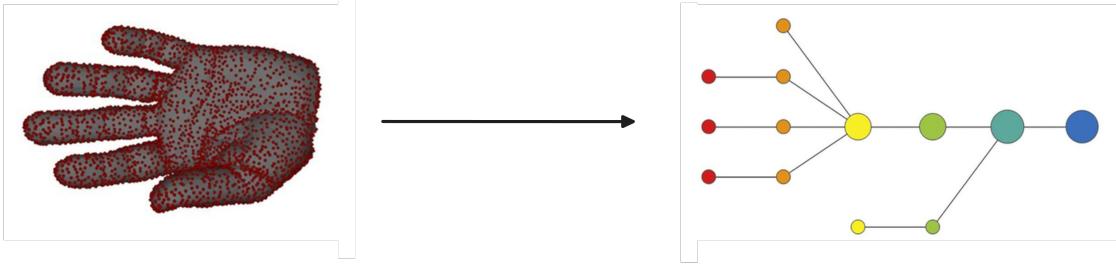


Figura 6: Ejemplo del funcionamiento del Mapper

El proceso comienza con un conjunto de datos en el que se consideran nociones de distancia entre los puntos. Para simplificar la explicación, se hará uso de la distancia euclidiana.

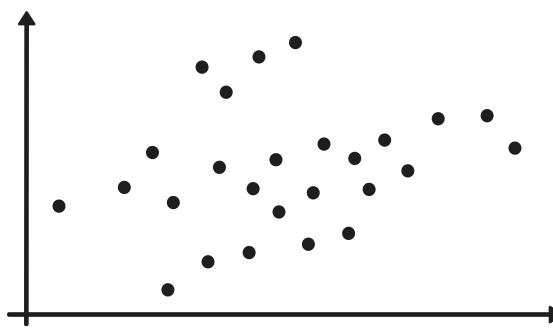


Figura 7: Nube de puntos para aplicar un Mapper

Inicialmente, es necesario aplicar una técnica de clusterización al conjunto de datos. Esta clusterización puede realizarse mediante cualquier método existente.

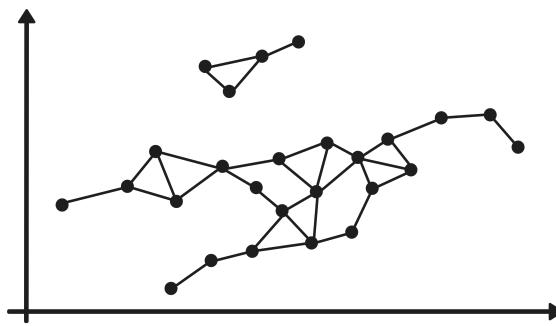


Figura 8: Grafo para aplicar un Mapper

Posteriormente, se aplica una función de filtro para reducir la dimensionalidad de los datos. Ejemplos comunes de esto incluyen técnicas como el Análisis de Componentes Principales (PCA), Uniform Manifold Approximation and Projection (UMAP), o simplemente una proyección sobre uno de los ejes. A continuación, se define una cubierta (*cover*). Si el resultado del filtro es unidimensional, la cubierta consistirá en intervalos unidimensionales que se solapan. Si el resultado es bidimensional, la cubierta será formada por rectángulos que también se solapan en el plano, y así sucesivamente para dimensiones mayores.

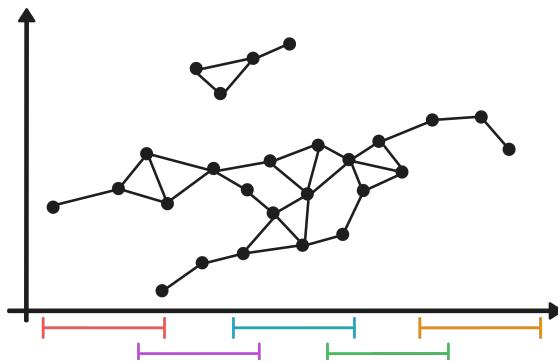


Figura 9: Cubierta definida

Para cada elemento de la cubierta, se examinan las componentes conexas de su preimagen.

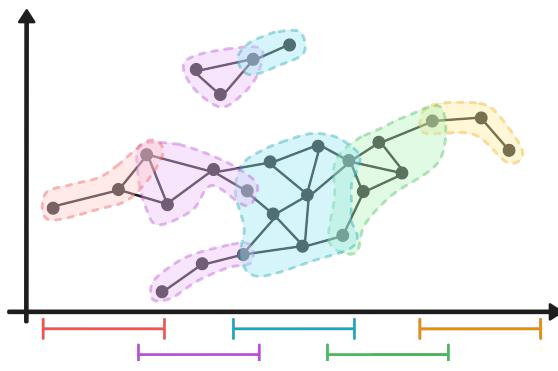


Figura 10: Cubierta aplicada al grafo

El Mapper resultante es el “nervio” de esta cubierta, es decir, una estructura en la cual los nodos conectados en esta sección se unen. En consecuencia, este tipo específico de agrupación, función de filtro y cubierta produce un Mapper como resultado. Este proceso permite visualizar y entender mejor la estructura subyacente y las relaciones en conjuntos de datos complejos.

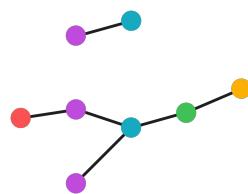


Figura 11: Resultado final del Mapper

### 2.2.2. Redes neuronales convolucionales

Las redes neuronales convolucionales están diseñadas específicamente para procesar imágenes. Su arquitectura está optimizada para reconocer patrones espaciales en los datos, lo que las hace muy eficaces en tareas de visión por computador, como la clasificación de imágenes y la detección de objetos.

## Epoch de una CNN

En el contexto del entrenamiento de una CNN, el término “*epoch*” se refiere a una fase durante la cual la red neuronal procesa todo el conjunto de datos una vez. Durante una *epoch*, la red ajusta sus pesos y sesgos con el objetivo de minimizar la función de pérdida, un proceso que mejora la precisión del modelo al predecir los datos no vistos. Cada *epoch* se compone de varios pasos, en los cuales los lotes (*batches*) de datos son procesados secuencialmente. Este ciclo es fundamental para el entrenamiento eficaz de cualquier modelo de aprendizaje profundo.

## Convolución en CNN

Una de las principales partes de una CNN es la operación de convolución, un proceso matemático que transforma la imagen de entrada en mapas de características, capturando así los patrones espaciales cruciales.

Durante este proceso, un *kernel* o filtro (una pequeña matriz de pesos aprendibles) se desliza sobre la imagen de entrada. En cada posición, se realiza el producto y la suma los valores de cada píxel vecino para obtener así el nuevo valor del píxel a generar.

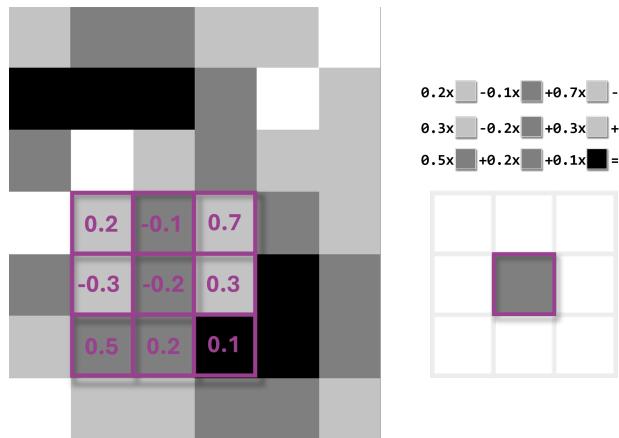


Figura 12: Ejemplo de kernel

El resultado es un mapa de activaciones que muestra una representación abstracta de la entrada original. Cada uno de estos mapas nos indica en qué parte de la imagen se ha detectado la característica buscada por el filtro, cada píxel blanco es una activación que nos indica que el elemento buscado estaba ahí. Diferentes filtros pueden detectar bordes, esquinas, o incluso patrones más complejos a medida que se avanza en las capas de la red. [1]

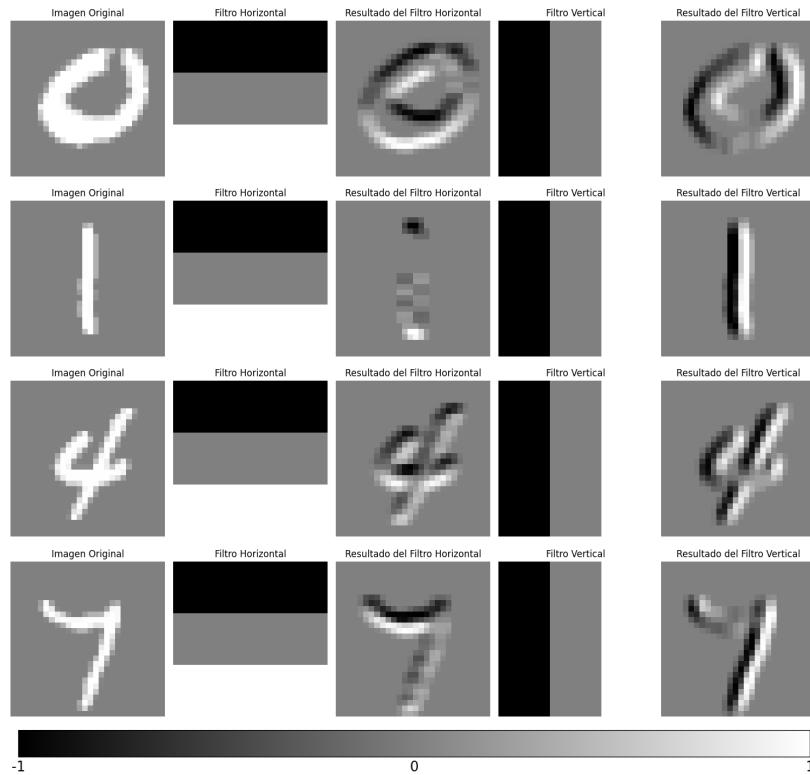


Figura 13: Ejemplos de filtros de bordes horizontal y vertical

En la Figura 13 tenemos el ejemplo de dos filtros, uno que detecta bordes horizontales y otro que detecta bordes verticales en las imágenes. Estos dos filtros son matrices  $3 \times 3$  donde el color negro representa un  $-1$  y el color blanco un  $1$ .

Es fundamental tener en cuenta que mientras más profundas sean las capas, más abstractas y representativas son las características identificadas.

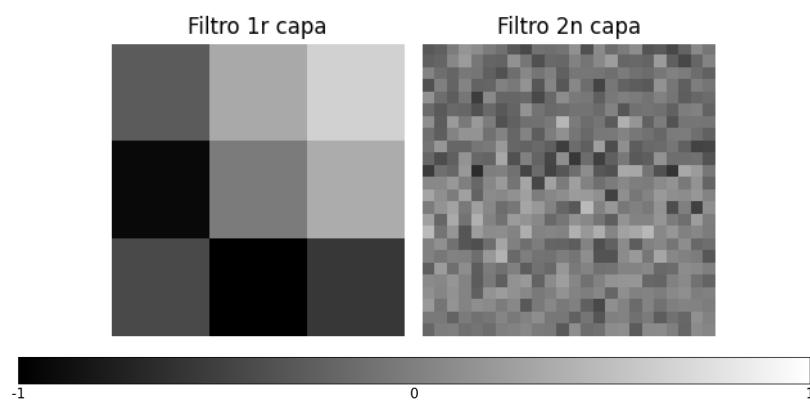


Figura 14: Comparación entre un filtro de la capa 1 y de la capa 2

## Función de Activación

Tras cada operación de convolución, las CNN típicamente aplican una función de activación no lineal. Estas se utilizan para introducir no linealidad en el modelo, permitiendo a la red aprender y realizar tareas más complejas. Sin estas funciones, la red no podría capturar adecuadamente las

interacciones entre los píxeles, lo que llevaría a no poder reconocer correctamente las imágenes. Las funciones de activación ayudan a controlar el problema del desvanecimiento del gradiente durante el entrenamiento de la red. Algunos ejemplos comunes de funciones de activación son: la función ReLU (Rectified Linear Unit), la función sigmoide o la función tangente hiperbólica entre otras. [1]

### 2.2.3. Dataset MNIST

El dataset MNIST es un conjunto de datos clásico ampliamente extendido y utilizado en el campo del aprendizaje automático y visión por computador. Consta de 70000 imágenes en escala de grises de dígitos manuscritos que van del 0 al 9. Se utiliza frecuentemente para entrenar y probar sistemas de reconocimiento de imágenes debido a su tamaño manejable y porque sus patrones son lo suficientemente complejos para probar algoritmos de aprendizaje automático.

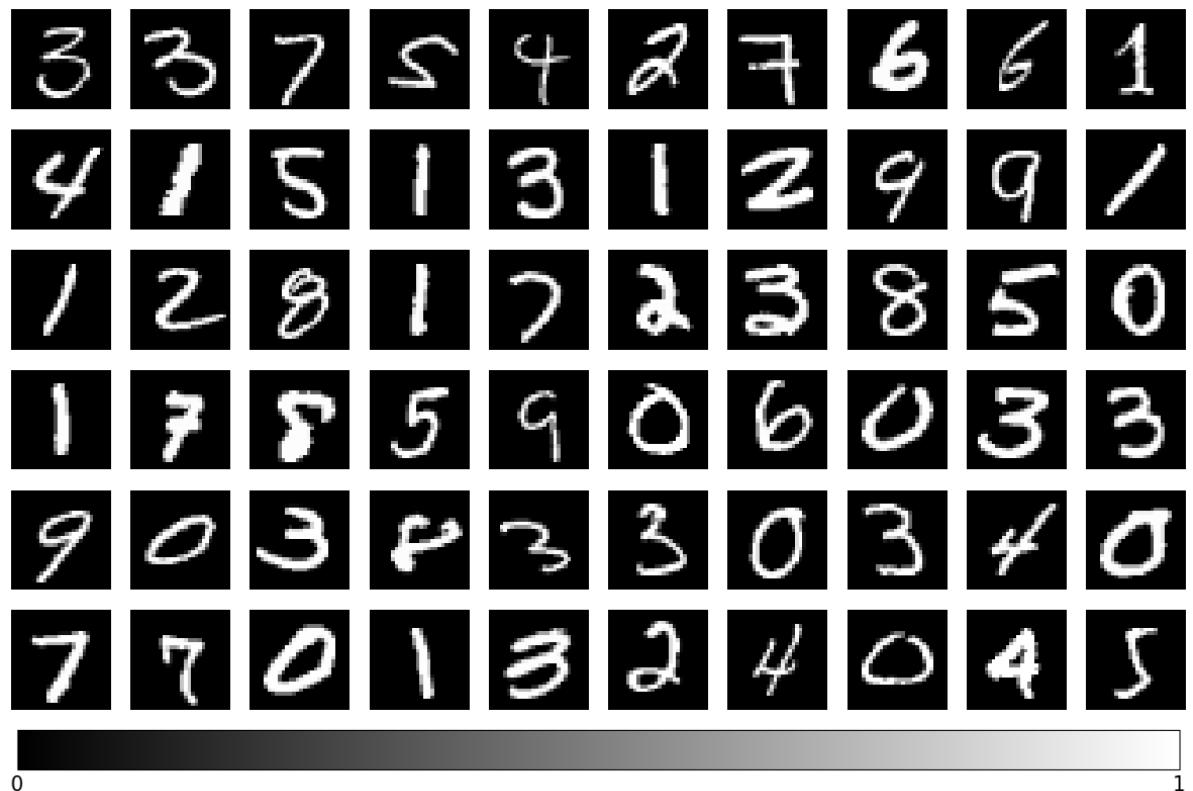


Figura 15: Algunas muestras del dataset MNIST

#### 2.2.4. Técnicas de reducción de dimensionalidad

Las técnicas de reducción de dimensionalidad son métodos utilizados en el análisis de datos y aprendizaje automático para reducir la cantidad de variables o características en un conjunto de datos. Esta reducción se realiza conservando la mayor cantidad posible de información relevante.

La necesidad de reducir la dimensionalidad surge en situaciones donde los conjuntos de datos contienen una gran cantidad de características, lo que puede llevar a problemas como dificultades de visualización o un aumento en la complejidad computacional de los modelos.

Estas técnicas permiten simplificar la representación de los datos al eliminar características redundantes, irrelevantes o correlacionadas, lo que puede llevar a una mejor comprensión de los datos, modelos más eficientes y una mejora en el rendimiento predictivo.

#### PCA

El Análisis de Componentes Principales (PCA, por sus siglas en inglés) es un algoritmo de aprendizaje automático no supervisado. A diferencia de los métodos supervisados, el PCA no se enfoca en predecir una variable de respuesta, más bien, su objetivo es explorar las características que conforman nuestro conjunto de datos  $X$ .

La finalidad de este algoritmo es reducir la cantidad de variables, pasando de  $N$  a  $K$ , donde  $K < N$ . Esto permite comprimir los datos originales en un conjunto de menor dimensión, facilitando así su representación gráfica o la reducción de su volumen.

$$X \in \mathbb{R}^N \xrightarrow{\text{PCA(n_components=K)}} X' \in \mathbb{R}^K \quad (1)$$

Para calcular las componentes principales, el PCA de manera iterativa descompone linealmente las variables originales en estas nuevas componentes.

Las variables originales están multiplicadas por factores llamados *loading* que han de cumplir:

$$\begin{aligned} Q_1 &= f_{11}X_1 + f_{21}X_2 + \cdots + f_{N1}X_N \\ Q_2 &= f_{12}X_1 + f_{22}X_2 + \cdots + f_{N2}X_N \end{aligned} \quad (2)$$

⋮

$$Q_K = f_{1K}X_1 + f_{2K}X_2 + \cdots + f_{NK}X_N$$

$$\sum_{n=1}^N f_{nk}^2 = 1 \quad \forall k \in [1, \dots, K] \quad (3)$$

Para calcular las componentes principales se siguen los siguientes pasos:

1. Se estandarizan los datos originales, es decir, se transforman los datos para que tengan media 0 y desviación estándar 1.
2. Mediante un proceso de optimización se busca el valor de los *loadings* que maximice la varianza. A continuación, se calculan los valores y vectores propios de la matriz de covarianzas de los datos.

Los valores propios indican la cantidad de varianza retenida por cada componente principal, mientras que los vectores propios determinan las direcciones de máxima varianza en el espacio de los datos.

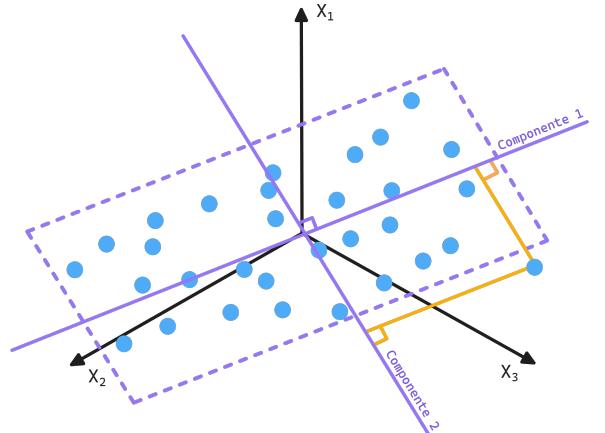


Figura 16: PCA de 2 componentes en datos de dimensión 3

El cálculo de las demás componentes sigue el mismo procedimiento, pero teniendo en cuenta que cada nueva componente debe ser ortogonal a las anteriores.

El número máximo de componentes principales que se pueden calcular es igual al número menor entre la cantidad de observaciones y la cantidad de variables en el conjunto de datos.

$$K_{\max} = \min(n_{\text{muestras}}, n_{\text{variables}}) - 1 \quad (4)$$

Si se calcula el máximo de componentes principales, la varianza total de estas ha de ser igual a la varianza total de los datos originales.

$$\sum_{n=1}^N \text{Var}(X_n) = \sum_{k=1}^{K_{\max}} \text{Var}(Q_k) \quad (5)$$

Si se tienen  $K$  componentes, la varianza total explicada será la varianza explicada de estas  $K$  componentes entre la varianza total inicial.

$$\text{Var}(Q_1, \dots, Q_K) = \frac{\sum_{k=1}^K \text{Var}(Q_k)}{\sum_{n=1}^N \text{Var}(X_n)} \quad (6)$$

Se pueden graficar estas variables, con esto se puede ver la varianza de cada componente, donde cada componente se muestra en orden descendente de varianza explicada, siendo la primera la que más tiene, seguida de la segunda con más varianza, y así sucesivamente hasta  $K$ .

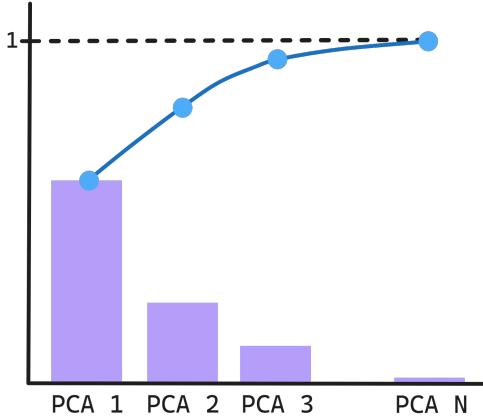


Figura 17: Varianzas del PCA

La suma total de varianzas de todas las componentes principales ha de ser igual a 1. [3]

## UMAP

El Uniform Manifold Approximation and Projection (UMAP), es una técnica de reducción de dimensionalidad desarrollada por Leland McInnes, John Healy y James Melville. [10] Se utiliza para visualizar grandes conjuntos de datos, especialmente cuando se busca preservar tanto las relaciones locales como globales dentro de los datos, facilitando así la interpretación visual de estructuras complejas de manera más sencilla.

El algoritmo de UMAP funciona creando un grafo de la siguiente manera:

Siendo  $k$  el número de vecinos de un *cluster*,  $X$  una nube de puntos y  $P_n$ ,  $n$  puntos que cumplen  $P_n \in X$ , se define el peso entre dos puntos cómo:

$$W(P_i, P_j) = e^{-(d_E(P_i, P_j) - \min(d_E(P_i, P_*))) / \sigma} \quad \text{donde: } \sum_{j=0}^n W(P_i, P_j) = \log_2(k) \quad (7)$$

En el ejemplo de la Figura 18 se ve una nube de tres puntos con sus respectivas distancias euclidianas y como quedarían los pesos del grafo calculados solo para un punto:

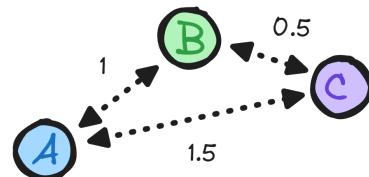


Figura 18: Distancias de una nube de puntos

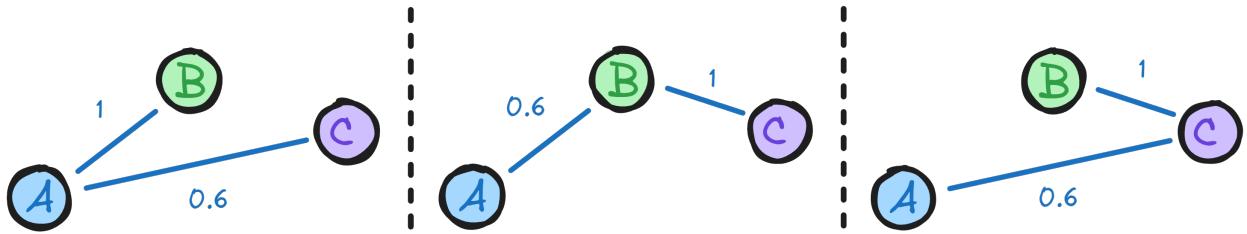


Figura 19: Pesos asimétricos de la nube de puntos

Se modifica  $\sigma$  para que se cumpla en todo momento  $\sum_{j=0}^n W(P_i, P_j) = \log_2(k)$ . Para los nodos que estén más alejados (en otros *clusters*) estos pesos serán tan pequeños que se consideran iguales a cero.

Después se unifican los pesos de cada par de nodos usando la siguiente operación, donde  $S_i$  son los pesos asimétricos entre dos puntos iguales del grafo:

$$W_{\text{simétrico}} = (S_1 + S_2) - S_1 S_2 \quad (8)$$

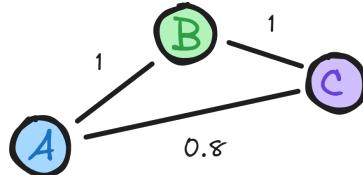


Figura 20: Pesos unificados de la nube de puntos

Se sigue el mismo procedimiento para el resto de los puntos de la nube.



Figura 21: Nube de puntos aumentada

Una vez calculados todos los pesos, el UMAP inicializa un gráfico de menor dimensión utilizando el método de *spectral embedding*. [2]



Figura 22: Gráfico de menor dimensión inicializado por el UMAP

Seguidamente, escoge dos pares aleatorios de un mismo *cluster* y otro de un *cluster* diferente al

ya seleccionado, por ejemplo en nuestro caso  $B$ ,  $C$  y  $D$ . UMAP querrá mover el punto  $B$  cerca de  $C$  y lejos de  $D$ . Para esto utiliza la siguiente función de coste:

$$\text{Coste}(x_{\text{vecino}}, x_{\text{no-vecino}}) = \log\left(\frac{1}{x_{\text{vecino}}}\right) + \log\left(\frac{1}{1 - x_{\text{no-vecino}}}\right) \quad (9)$$

siendo  $x_i$  el valor del punto  $P_i$  en la dimensión reducida que se calcula de la siguiente manera:

$$x_i = \frac{1}{1 + \alpha d^{2\beta}} \quad (10)$$

donde:

$$\alpha = 1,577$$

$$\beta = 0,8951$$

$d$  = distancia entre los puntos que queremos acercar o alejar en esta nueva dimensión

UMAP minimizará esta función de coste para colocar el punto (en nuestro caso  $B$ ) en el lugar correcto. Para encontrar el gráfico óptimo de menor dimensión, UMAP utiliza el Descenso de Gradiente Estocástico (SGD, por sus siglas en inglés).



Figura 23: Resultado final UMAP

El uso de SGD implica que, aunque se introduzca el mismo input en UMAP, los resultados serán similares pero con algunas diferencias menores. [13] [14]

Para cuantificar como de bien funciona este algoritmo se utilizan las métricas de *trustworthiness* y *continuity*.

La *Trustworthiness*, o fiabilidad, es una métrica que evalúa qué tan bien se conserva la estructura de los datos cuando estos son reducidos de una alta a una baja dimensión. Específicamente, esta medida verifica si los vecinos más cercanos de cada punto en el espacio de alta dimensión siguen siendo cercanos en el espacio reducido. Se expresa como una puntuación, donde valores más altos indican una mejor preservación de la estructura original de los datos.

Por otro lado, la *Continuity*, o continuidad, complementa a la *Trustworthiness* al enfocarse en la preservación de las relaciones globales entre los datos durante el proceso de reducción dimensional. La *Continuity* asegura que los grupos de puntos o *clusters* que estaban juntos en el espacio original sigan agrupados en el espacio de baja dimensión. [15]

Sus fórmulas son respectivamente:

$$T(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^n \sum_{j \in U_i^{(k)}} (r(i, j) - k) \quad (11)$$

$$C(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^n \sum_{j \in U_i^{(k)}} (\hat{r}(i, j) - k) \quad (12)$$

donde:

$n$  = número de muestras

$i, j$  = punto en el espacio de alta dimensión

$k$  = número de vecinos más cercanos

$r$  = distancia entre los puntos  $i$  y  $j$  en el espacio de alta dimensión

$\hat{r}$  = distancia entre los puntos  $i$  y  $j$  en el espacio de baja dimensión

### 3. Metodología

Para analizar el impacto del análisis topológico en la comprensión de las redes neuronales convolucionales y cumplir con los objetivos previamente establecidos, se utilizará el dataset MNIST. La metodología consta de seis fases principales, detalladas a continuación:

#### 1. Preparación de los datos

Se seleccionarán y se preprocesarán las imágenes del dataset MNIST para adaptarlas al uso en redes neuronales convolucionales. Este proceso incluye la normalización de las imágenes, la conversión de formatos según sea necesario, y la división del dataset en conjuntos de entrenamiento, validación y prueba.

#### 2. Entrenamiento de las CNN

Se entrenarán las CNN utilizando los datos preparados. Tras entrenarlas, se guardarán los modelos para evitar la necesidad de reentrenarlos en cada sesión de análisis, facilitando así la reproducción y la comparación entre resultados.

#### 3. Visualización de los filtros aprendidos

Se visualizarán los filtros aprendidos para proporcionar una mejor idea de que han aprendido las CNN en cada *epoch*, y como estos filtros están distribuidos.

#### 4. Aplicación de técnicas de análisis topológico

Utilizando la librería de *giotto-tda* [5] [6] [7], se aplicarán técnicas de análisis topológico, para examinar en profundidad los pesos de las capas de la CNN. Las técnicas aplicadas serán: PCA, UMAP, *Vietoris-Rips* y el algoritmo *Mapper*.

- **Análisis de la primera capa convolucional:** Se estudiarán los pesos de la primera capa convolucional para entender su funcionamiento individual dentro de la red. Tratando de replicar el trabajo previamente explicado [4] a menor escala.

Se analizará cómo están distribuidos los filtros aprendidos por la CNN, cómo van evolucionando a medida que aumentan las *epochs* y cómo esto afecta al modelo.

Se compararán los filtros aprendidos de diferentes modelos, el modelo será siempre el mismo pero se cambiará el número de *epochs* para poder comparar los valores dados a los pesos en cada modelo. Se utilizarán las siguientes *epochs*: 1, 10, 100, 500, 1000 y 10000.

## 4. Desarrollo

Para cumplir con la metodología previamente mencionada, se han creado diversos archivos ‘.ipynb’ disponibles en el siguiente repositorio de GitHub.

- **CNN\_(64,32,64).ipynb:** Este archivo está enfocado en la construcción y entrenamiento de modelos CNN para el dataset MNIST utilizando Keras y TensorFlow.
- **carga\_modelo\_<n>\_epoch.ipynb:** Este cuaderno se encarga de cargar 10 modelos previamente entrenados y almacenar sus pesos en una hoja de cálculo.
- **filtros\_1r\_capa\_<n>\_epoch.ipynb:** Este archivo se centra en el análisis y visualización de los filtros de la primera capa de los 10 modelos CNN después de  $n$  epochs.
- **analisis\_filtros\_<n>\_epoch.ipynb:** Este archivo incluye una colección de pruebas y visualizaciones avanzadas, que abarcan el uso de algoritmos de aprendizaje automático y técnicas de reducción de dimensiones, además de la aplicación de análisis topológico.

### 4.1. Preparación de los datos

Se carga los datos del dataset MNIST utilizando el método `mnist.load_data()`, esta función viene dada por la biblioteca de aprendizaje profundo TensorFlow. Al llamar a `load_data()`, este método nos devuelve cuatro arrays de NumPy divididos en dos tuplas, para la primera tupla nos devuelve:

- **‘train\_images’:** Este array contiene las imágenes que se usarán para entrenar los modelos. En nuestro caso `train_images` tiene una forma de  $(60000, 28, 28)$ , es decir, tiene 60,000 imágenes de  $28 \times 28$  píxeles cada una.
- **‘train\_labels’:** Este contiene las etiquetas de entrenamiento, es decir, las etiquetas correspondientes a cada imagen de `train_images`. Cada etiqueta es un número entero entre 0 y 9 que representa el número escrito a mano de la imagen que corresponda.

La segunda tupla nos devuelve:

- **‘test\_images’:** Similar a `train_images`, este array contiene las imágenes que usaremos para evaluar qué tan bien funciona nuestro modelo de clasificación. Este array tiene una forma de  $(10000, 28, 28)$ , lo que significa que tiene 10,000 imágenes de  $28 \times 28$  píxeles cada una.
- **‘test\_labels’:** Este contiene las etiquetas que corresponden con las imágenes de `test_images`.

Una vez que se tienen los datos de entrenamiento y de testeo separados se normalizan las imágenes y se cambia el formato de algunas variables para el futuro uso de estas en la CNN.

## 4.2. Entrenamiento de las CNN

La arquitectura que se ha utilizado para las CNN es muy común para trabajar con imágenes, inspirada en la usada en el trabajo de Rickard Brüel Gabrielsson y Gunnar Carlsson [4], en este caso el tamaño del input es  $(28 \times 28 \times 1)$  la estructura de la CNN es la siguiente:

Tabla 1: Arquitectura de la CNN

Capa Conv 1	Capa Conv 2	Capa FC	Readout
$3 \times 3 \times 64$ filtros	$3 \times 3 \times 32$ filtros	64 nodos	10 nodos
ReLU	ReLU	ReLU	Softmax
$2 \times 2$ max-pooling	$2 \times 2$ max-pooling		

Las capas convolucionales aprenden filtros que reconocen patrones específicos en los datos, las capas de *pooling* disminuyen la cantidad de parámetros para prevenir el sobreajuste y simplificar la red. La capa densamente conectada FC (*fully connected*) aprende patrones globales. Finalmente, la capa de salida calcula la probabilidad de que la imagen de entrada pertenezca a cada una de las 10 clases.

El algoritmo de optimización para actualizar los pesos de la red es el ‘*adam*’, para la ‘*loss*’ se utiliza ‘*categorical\_crossentropy*’, la cual es apropiada para problemas de clasificación donde las etiquetas están ‘*one-hot encoded*’, es decir, cada etiqueta es un vector con un solo valor verdadero y el resto falso. La métrica de evaluación que usaremos para ver el rendimiento del modelo es el ‘*accuracy*’, que mide la fracción de imágenes correctamente clasificadas entre el total de imágenes.

El conjunto de imágenes de entrenamiento se pasará  $n$  veces por cada red ( $epochs = n$ ), y la actualización de los pesos de la red se hará cada 128 imágenes ( $batch\_size = 128$ ).

Además, se guardan los tiempos de ejecución y la precisión de los modelos aparte para después analizarlos.

Una vez los 10 modelos estén entrenados se extraen sus pesos para analizarlos posteriormente.

## 4.3. Visualización de los filtros aprendidos

Se extraen los filtros de la primera capa convolucional de la CNN entrenada con 1 *epoch* y se muestran en matrices de  $3 \times 3$  para poder visualizar cómo son. Lo primero que se puede observar son las siguientes métricas sobre los pesos de la primera capa convolucional de los 10 modelos entrenados con 1 *epoch*:

	max	min	mean	std
1 epoch	0.228	-0.306	0.025	0.095

Tabla 2: Métricas de los 10 modelos con 1 *epoch* de la primera capa

Se puede ver cómo son los filtros visualmente en la siguiente imagen:

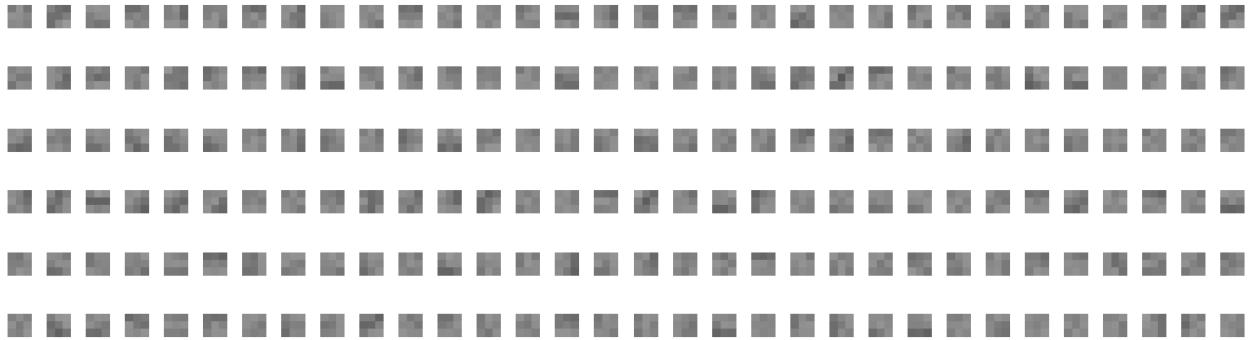


Figura 24: Filtros de la primera capa de los 10 modelos de 1 *epoch*

Se observa que, en general, todos los filtros presentan tonos predominantemente grises, lo cual es coherente con los valores del máximo y del mínimo, que son cercanos a cero, y con la desviación estándar, que es muy pequeña.

Para ver si estos filtros se parecen a los filtros teóricos se comparan píxel a píxel para poder clasificar a cuál de los filtros teóricos se parecen más.

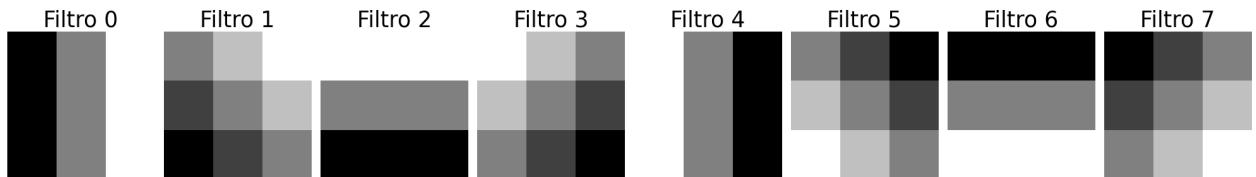


Figura 25: Filtros teóricos de la primera capa

Para ver si se parecen bien a estos filtros se han creado otros 4 filtros de manera aleatoria.

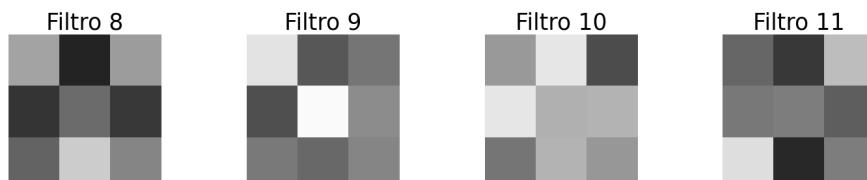


Figura 26: Filtros aleatorios para la primera capa

Una vez hecha la clasificación mirando la distancia euclídea entre las dos matrices se ven los siguientes resultados:

tipo <i>epoch</i>	0	1	2	3	4	5	6	7	8	9	10	11	tasa de éxito
1	0	3	0	0	0	0	0	0	147	131	163	196	0.0046

Tabla 3: Clasificación de los filtros de la primera capa de los 10 modelos de 1 *epoch*

La tasa de éxito está definida como la suma de las clasificaciones de los tipos 0-7 dividido entre

el total de clasificaciones, con esto se puede evaluar la proporción de imágenes que coinciden con los filtros teóricos de la primera capa.

Se muestran las medias de las clasificaciones para compararlas con los filtros teóricos.

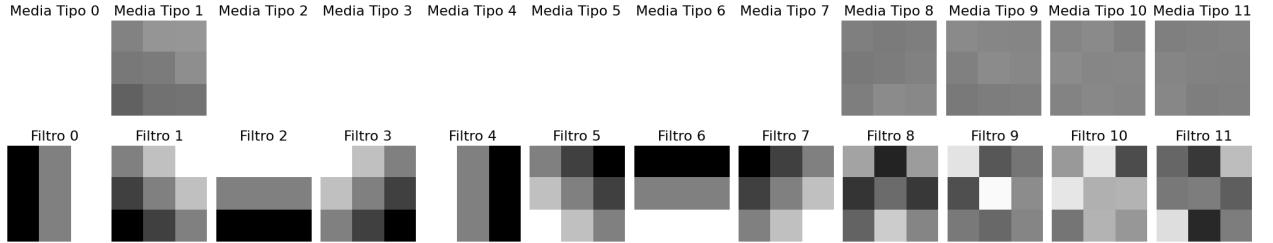


Figura 27: Comparación entre la media de las clasificaciones y los filtros teóricos y aleatorios

Se realiza el mismo procedimiento para el resto de modelos entrenados con diferentes *epochs*. Los resultados son los siguientes:

	max	min	mean	std
1 epoch	0.228	-0.306	0.025	0.095
10 epochs	0.430	-0.717	0.009	0.158
100 epochs	0.639	-1.151	-0.014	0.232
500 epochs	0.818	-1.462	-0.025	0.302
1000 epochs	0.948	-1.935	-0.041	0.366
10000 epochs	1.729	-2.630	-0.074	0.513

Tabla 4: Métricas de todos los modelos de la primera capa

epoch \ tipo	0	1	2	3	4	5	6	7	8	9	10	11	tasa de éxito
1	0	3	0	0	0	0	0	0	147	131	163	196	0.0046
10	0	47	0	41	0	28	0	20	119	117	95	173	0.2125
100	0	63	0	41	0	40	0	38	148	82	69	159	0.2844
500	3	76	2	64	2	56	0	43	149	60	56	129	0.3844
1000	8	59	13	56	7	64	0	41	135	70	42	145	0.3875
10000	18	53	30	38	15	49	18	30	156	74	30	129	0.3922

Tabla 5: Clasificación de los filtros de la primera capa de todos los modelos

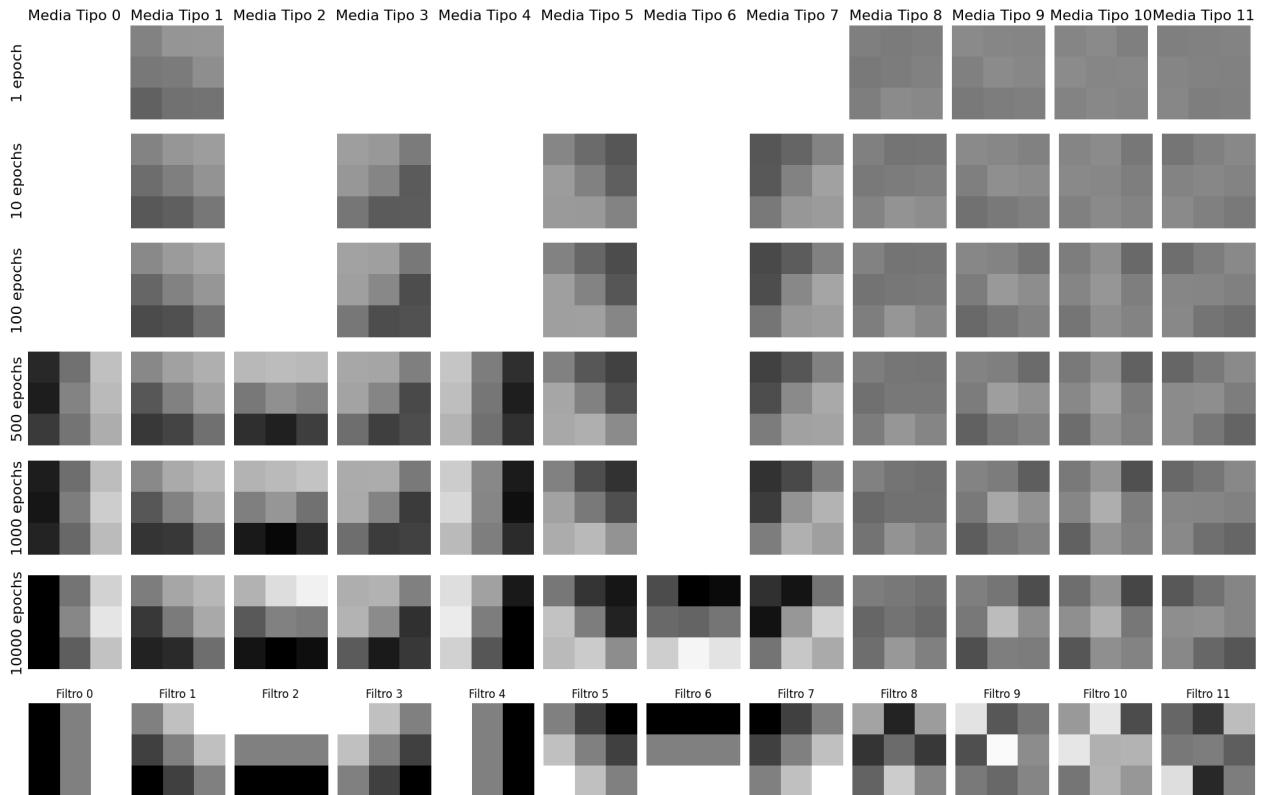


Figura 28: Comparación entre la media de las clasificaciones y los filtros teóricos y aleatorios

#### 4.4. Aplicación de técnicas de análisis topológico

Igual que en el apartado anterior se empieza con los modelos de 1 *epoch* para después hacer las comparaciones con los demás.

Primero se cargan los filtros aprendidos del modelo, para ver como estos están distribuidos. Como estos datos están en dimensión 9 no se pueden mostrar directamente, por lo tanto, primero se tiene que hacer una reducción de la dimensionalidad, esta se hará usando las 2 técnicas mencionadas anteriormente, PCA y UMAP.

##### 4.4.1. Aplicando PCA

Para poder ver bien la distribución en el espacio de estos datos se aplica un PCA con 2 y 3 componentes:

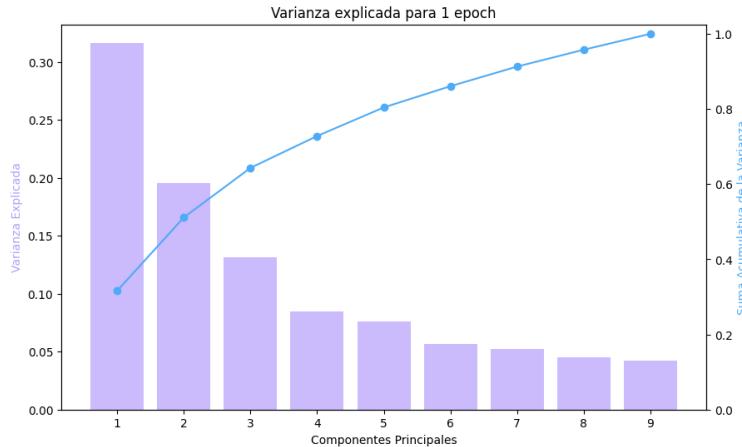


Figura 29: Varianza explicada para los modelos entrenados con 1 *epoch*

Tal y como se puede ver en la Figura 31, las varianzas explicadas para 2 y 3 son 0.511 y 0.643 respectivamente.

Al mostrar el resultado de los dos PCA se ve el siguiente resultado:

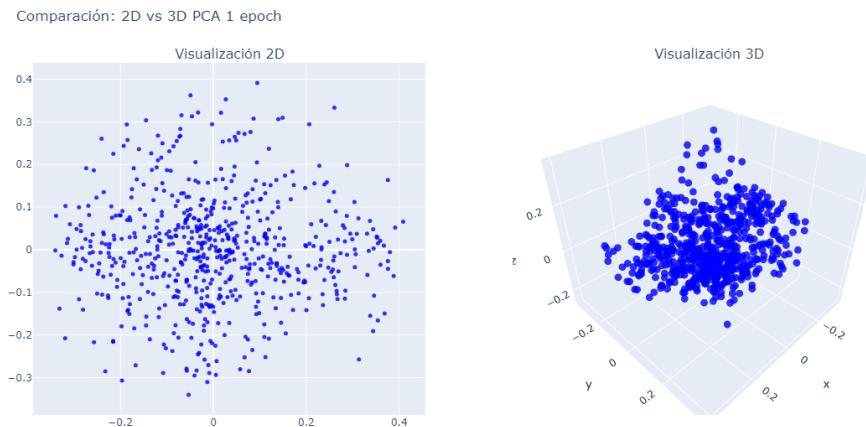


Figura 30: Comparativa PCA: 2D y 3D para los modelos de 1 *epoch*

Se hace el mismo análisis para el resto de modelos entrenados con más *epochs* y se obtienen los siguientes valores:

	Varianza explicada PCA 2D	Varianza explicada PCA 3D
1 <i>epoch</i>	0.512	0.643
10 <i>epochs</i>	0.667	0.787
100 <i>epochs</i>	0.570	0.712
500 <i>epochs</i>	0.595	0.750
1000 <i>epochs</i>	0.573	0.736
10000 <i>epochs</i>	0.590	0.741

Tabla 6: Varianza explicada PCA 2D vs 3D de todos los modelos

#### 4.4.2. Aplicando UMAP

Se aplica el UMAP para ver cómo este distribuyen los datos en el espacio, se hace con 2 y 3 componentes también.

Se utilizan las métricas *trustworthiness* y *continuity* para evaluar la eficacia del UMAP, dando los siguientes resultados:

<i>1 epoch</i>	<i>trustworthiness</i>	<i>continuity</i>
UMAP 2D	0.954	0.413
UMAP 3D	0.978	0.490

Tabla 7: Métricas de validación UMAP para 1 *epoch*

Al mostrar el resultado de los dos UMAP se observa el siguiente resultado:

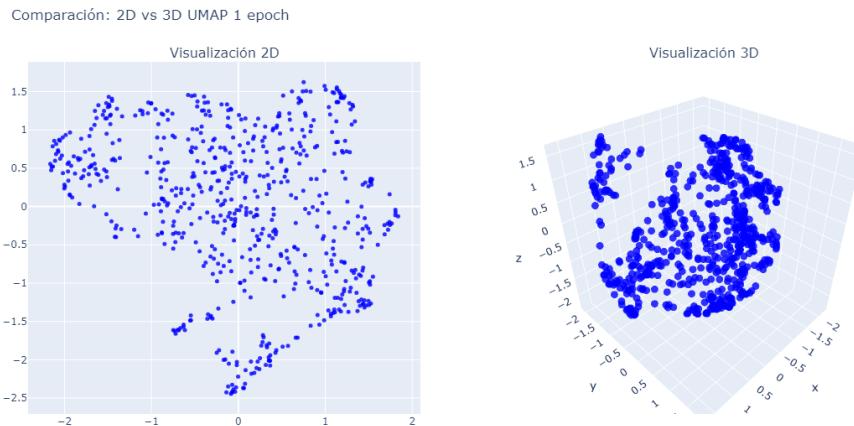


Figura 31: Comparativa UMAP: 2D y 3D para los modelos de 1 *epoch*

Se realiza el mismo análisis para el resto de modelos entrenados con más *epochs* y se obtienen los siguientes valores:

	<i>trustworthiness</i>		<i>continuity</i>	
	UMAP 2D	UMAP 3D	UMAP 2D	UMAP 3D
1 <i>epoch</i>	0.958	0.978	0.428	0.480
10 <i>epochs</i>	0.973	0.978	0.442	0.484
100 <i>epochs</i>	0.972	0.975	0.458	0.486
500 <i>epochs</i>	0.966	0.976	0.444	0.491
1000 <i>epochs</i>	0.967	0.978	0.470	0.503
10000 <i>epochs</i>	0.962	0.969	0.459	0.497

Tabla 8: Métricas de validación UMAP 2D vs 3D de todos los modelos

#### 4.4.3. Homología persistente de Vietoris-Rips

Para entender la estructura topológica de los datos sin simplificarlos, se usan los diagramas de persistencia de Vietoris-Rips. Estos diagramas ayudan a identificar las características relevantes de los datos, mostrando cuándo aparecen y cuánto duran, para distinguir aquello significativo del simple ruido.

A continuación se muestran cómo son estos diagramas de persistencia para cada entrenamiento con distintas *epochs*.

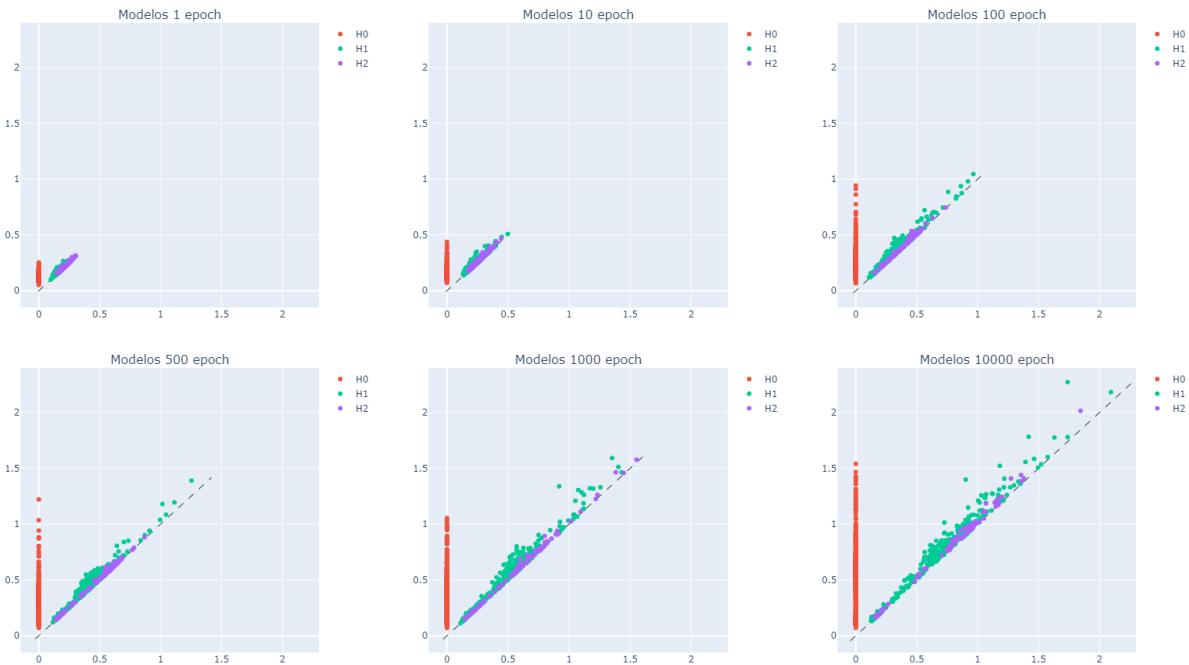


Figura 32: Comparación de la homología persistente para todos los datos sin tratar

También se pueden ver estos diagramas de persistencia con los datos después de haberles aplicado la reducción de dimensionalidad.

Se muestran las diferencias entre el PCA y el UMAP con 2 y 3 componentes para 1 *epoch* en las siguientes gráficas:

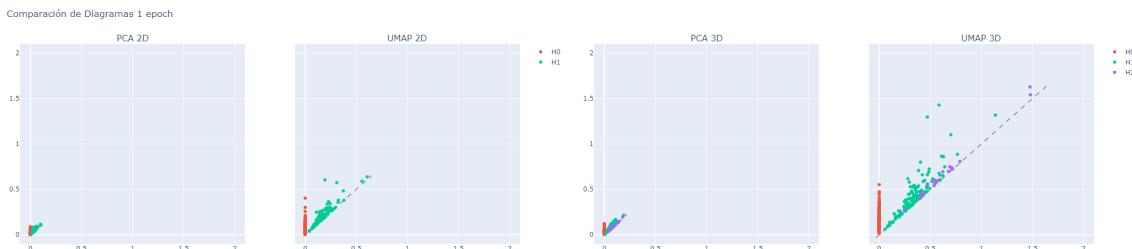


Figura 33: Vietoris-Rips para 1 *epoch* de PCA y UMAP 2D

Se pueden observar los mismos gráficos para los modelos entrenados con 10000 *epochs*:

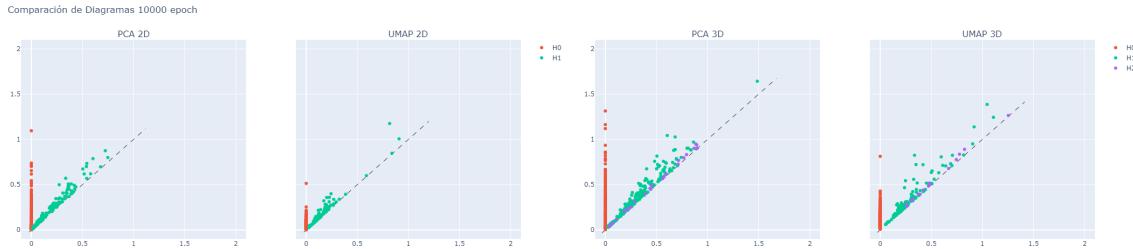


Figura 34: Vietoris-Rips para 10000 *epochs* de PCA y UMAP 2D

#### 4.4.4. Aplicación del Mapper

Finalmente se aplica un *Mapper* para ver si este puede dar más información de las estructuras que forman los filtros en el espacio. El *Mapper* se hará utilizando la librería ‘*kmapper*’. Para el ‘*cover*’ se utiliza ( $n\_cubes = 2$ ) y ( $prec\_overlap = 0,05$ ), es decir, se separa el rango de datos en 2 cubos y se utiliza un 5% de sobreposición entre los cubos. El método de *clustering* es el ‘*AffinityPropagation*’ de la librería ‘*sklearn.cluster*’.

Para las primeras figuras no se usa ninguna ‘*lens*’ es decir, no se reduce la dimensionalidad de los datos.

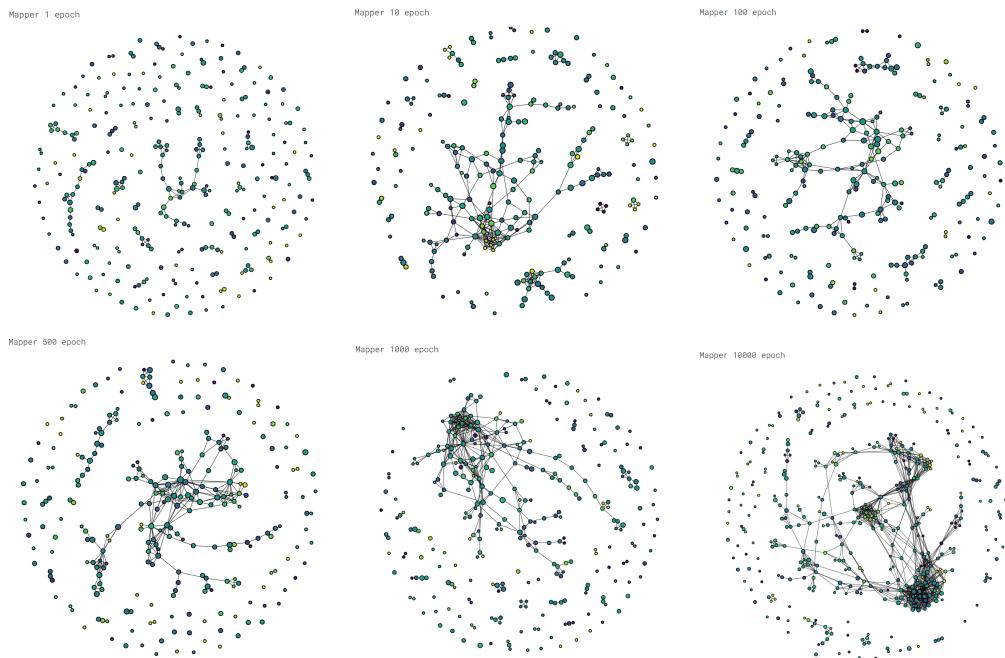


Figura 35: Comparación de los Mappers para cada *epoch*

También se ha probado de hacer estas mismas figuras pero esta vez solo para 10000 *epochs* (ya que en este punto los filtros están más definidos) y reduciendo su dimensionalidad con PCA y UMAP con 2 y 3 componentes cada uno. Se muestran las figuras resultantes a continuación:

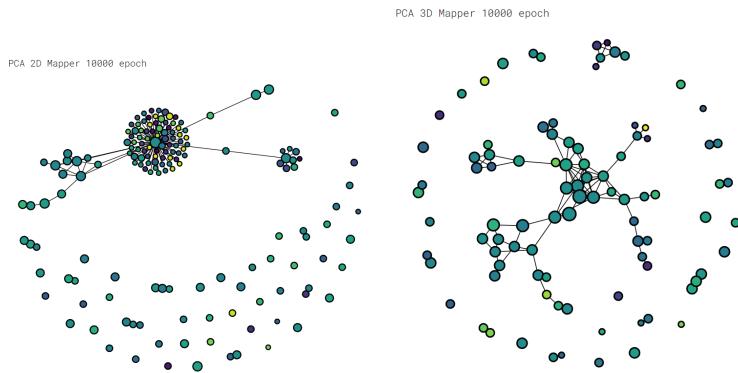


Figura 36: Mapper para el PCA 2D y 3D de 10000 *epochs*

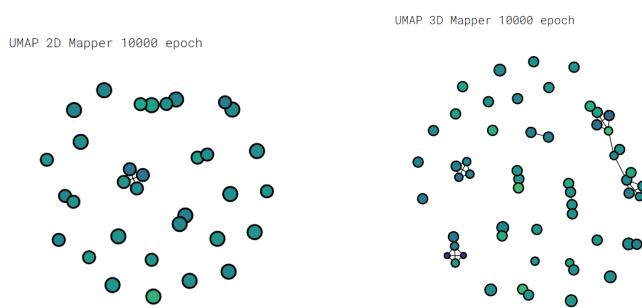


Figura 37: Mapper para el UMAP 2D y 3D de 10000 *epochs*

## 5. Resultados

En este apartado, se verán los resultados obtenidos de la aplicación de las técnicas de análisis topológico sobre los modelos de redes neuronales convolucionales entrenados con el dataset MNIST. Se observa cómo la estructura y evolución de los pesos en diferentes capas de la CNN pueden ser interpretadas desde una perspectiva topológica, para tratar de dar un enfoque diferente al funcionamiento interno y la toma de decisiones de las CNN.

### 5.1. Tiempos de ejecución y precisión de los modelos

En este apartado, se analizan los tiempos de ejecución y la precisión alcanzada por los modelos a lo largo de diferentes números de *epochs*.

	Media de tiempo (s)	Media de precisión (%)
1 epoch	18.19	97.71
10 epochs	183.68	98.96
100 epochs	1858.50	99.26
500 epochs	9290.13	99.22
1000 epochs	1825.07	99.20
10000 epochs	18023.92	99.23

Tabla 9: Tiempos de ejecución y precisión de los modelos

La Tabla 9 presenta datos para 1, 10, 100, 500, 1000 y 10000 *epochs*. Se nota que la media de precisión de los modelos aumenta ligeramente con el incremento en el número de *epochs*. Inicialmente, la precisión es del 97.71 % para 1 *epoch* y asciende hasta el 99.26 % para 100 *epochs*. Más allá de este punto, la precisión se mantiene estable, indicando una mejora gradual en la capacidad del modelo para clasificar las imágenes a medida que se expone repetidamente al mismo conjunto de entrenamiento, hasta alcanzar un límite.

Desde el punto de vista del tiempo de ejecución, se observa que para *epochs* que van de 1 a 500, el tiempo medio de ejecución por *epoch* es de aproximadamente 18 segundos. Sin embargo, para 1000 y 10000 *epochs*, el tiempo promedio disminuye significativamente a 1.8 segundos por *epoch*. Esta reducción es atribuible al alto costo computacional que implicaban estos modelos, por lo tanto se tuvo que cambiar de máquina para estos entrenamientos más costosos.

Las dos máquinas utilizadas tienen las siguientes prestaciones:

	Máquina 1	Máquina 2
Procesador (CPU)	Intel(R) Core(TM) i7-9750H	Intel(R) Xeon(R)
Memoria RAM	16 GB	32 GB
Tarjeta Gráfica (GPU)	RTX 2060	T4 x2

Tabla 10: Características de cada máquina

La Máquina 1 corresponde a un ordenador personal, mientras que la Máquina 2 hace referencia a la máquina virtual provista por Kaggle, utilizada especialmente para los entrenamientos de mayor duración y complejidad debido a sus capacidades computacionales superiores.

### 5.2. Comparación de filtros por etapa de entrenamiento

Al analizar la evolución de los filtros aprendidos durante distintas etapas del entrenamiento de la red neuronal convolucional, se observa cambios significativos que se reflejan tanto visualmente como en nuestras métricas cuantitativas.

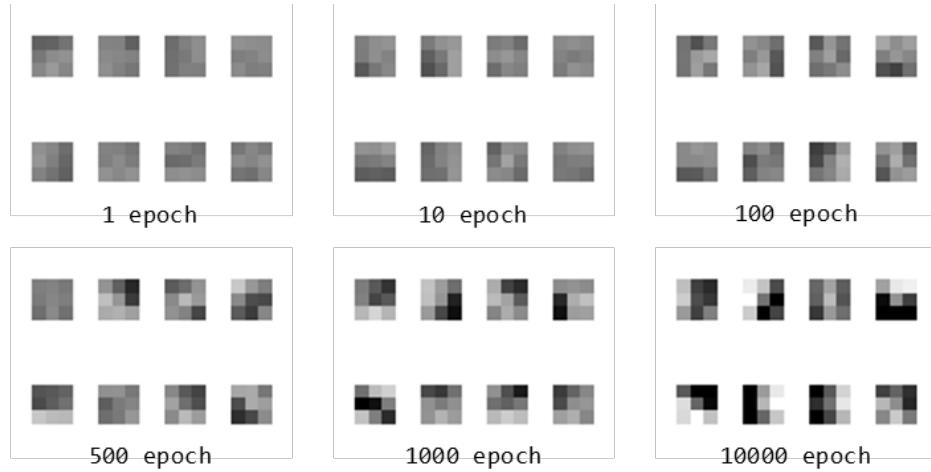


Figura 38: Comparativa 8 filtros aleatorios de cada una de las etapas del entrenamiento

En la Figura 38 se muestra claramente cómo el contraste en los filtros aumenta a medida que se incrementa el número de *epochs*. Este comportamiento, previamente cuantificado en la Tabla 4, revela que tanto los valores máximos como los mínimos de los filtros tienden a incrementar y disminuir respectivamente con el aumento de *epochs*, lo que también lleva a un aumento en la desviación estándar de estos valores.

Observando la Tabla 5, se nota que en los modelos entrenados con una sola *epoch*, la mayoría de los filtros se clasifican como los tipos 8-11, que corresponden a filtros generados aleatoriamente. La tasa de éxito en esta etapa es de apenas el 0.46 %. Sin embargo, a medida que el entrenamiento avanza y se incrementa el número de *epochs*, se empieza a ver una mejora significativa en la clasificación de los filtros aprendidos como teóricos, llegando hasta a un 39.22 % de los filtros clasificados como teóricos en los modelos entrenados con 10000 *epochs*.

La evolución de la clasificación de estos filtros a lo largo de diferentes *epochs* se visualiza en la Figura 28, donde se muestran las medias de los filtros clasificados en cada tipo de filtro por cada *epoch*.

Con este análisis se observa cómo la red aprende a enfocarse en características más definidas y relevantes conforme se expone más tiempo al entrenamiento.

## 5.3. Resultados de las técnicas de análisis topológico aplicadas

### 5.3.1. Resultados del PCA

En la Tabla 6 se observa cómo la varianza explicada para 2 y 3 componentes aumenta al pasar de 1 a 10 *epochs*, pero una vez alcanzado este punto, se mantiene estable e incluso disminuye ligeramente. Se puede notar que para 2 y 3 componentes, la varianza media es de 0.6 y 0.75 respectivamente, lo cual indica que estamos perdiendo bastante varianza al aplicar el PCA.

Visualmente, se observa estos resultados para el PCA según el tiempo de entrenamiento en las siguientes gráficas:

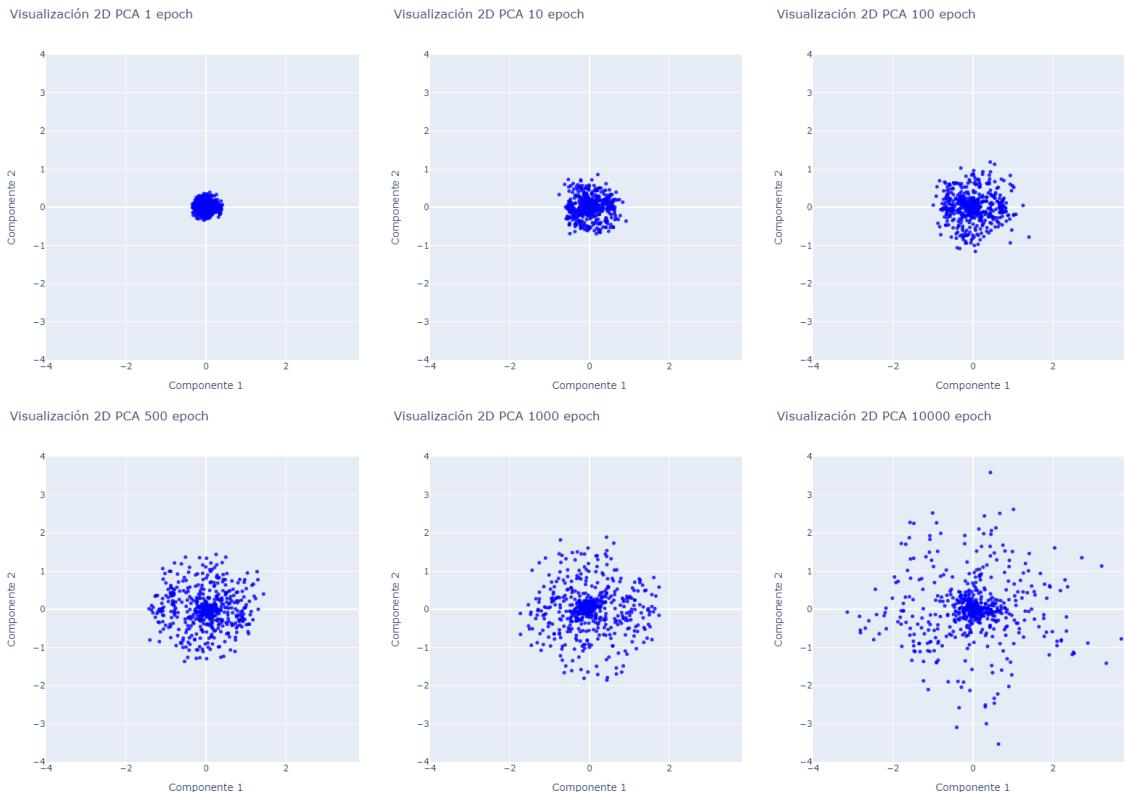


Figura 39: Evolución del PCA con 2 componentes respecto las *epochs*

A medida que avanza el entrenamiento, los filtros se van separando más del centro, es decir, se diferencian más del punto donde están aquellos que son prácticamente grises.

También se puede comparar el PCA de 3 componentes en base al tiempo de entrenamiento.

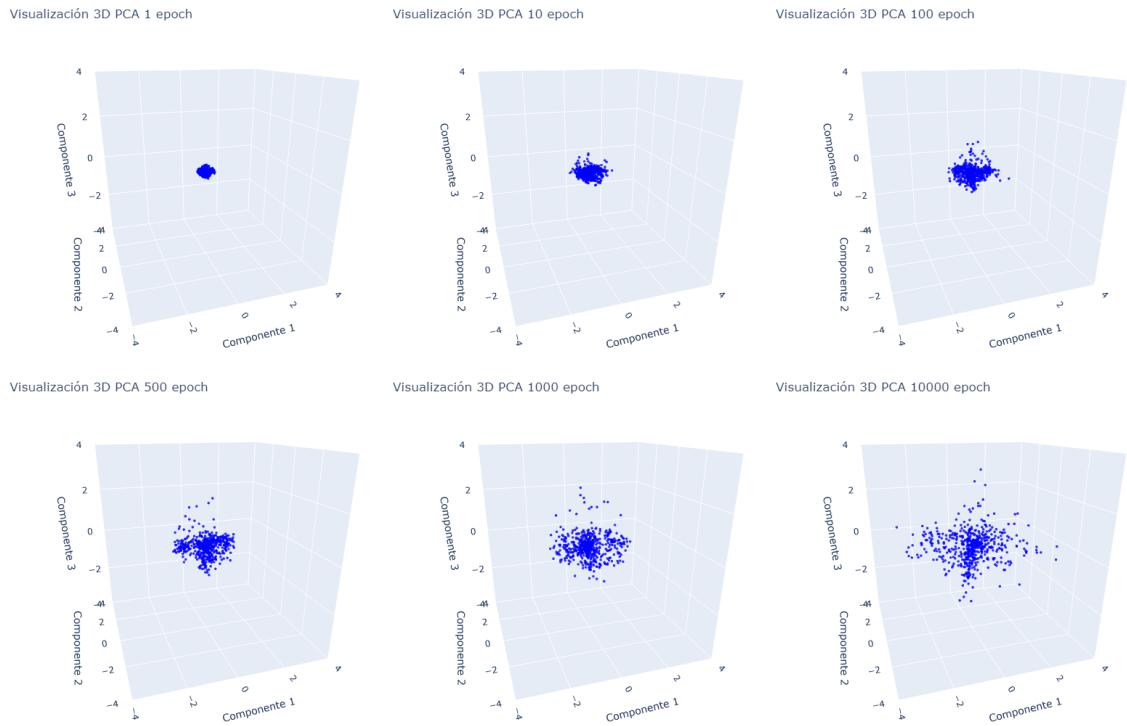


Figura 40: Evolución del PCA con 3 componentes respecto las *epochs*

En los gráficos 3D se observa un comportamiento similar al del PCA con dos componentes, aunque aquí se puede ver cómo, a medida que avanzan las *epochs*, se expande en los ejes de las componentes 1-2 y 2-3.

Si se analiza más en profundidad el último de los gráficos (Figura 41) se puede ver cómo se está formando claramente un círculo principal (círculo azul) y dos posibles círculos secundarios, aunque no muy definidos (círculos naranja y verde).

Visualización 3D PCA 10000 epoch

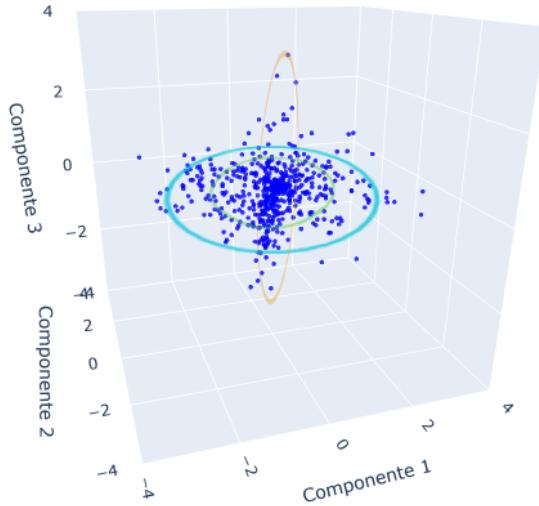


Figura 41: Círculos primario y secundarios del PCA 3D

Se analiza el círculo azul, ya que es el que ha tenido una evolución más clara a lo largo de las *epochs*. Para ello, se realiza una proyección sobre los ejes de las componentes 1-2 del gráfico tridimensional de 10000 *epochs*. Como en el centro la mayoría de los filtros son grises, se pueden obviar para esta visualización. A continuación, se separa el resto de los puntos en 8 cuadrantes con el centro en (0,0), lo que deja el siguiente gráfico como resultado:

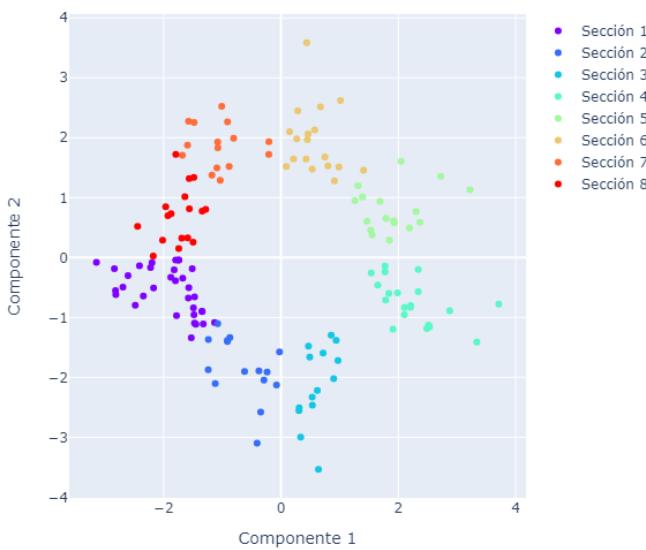


Figura 42: Proyección 2D del PCA 3D en los componentes 1 y 2

Se calcula la media de los filtros en cada sección para ver cómo se comportan estos en el espacio y se obtiene la siguiente gráfica:

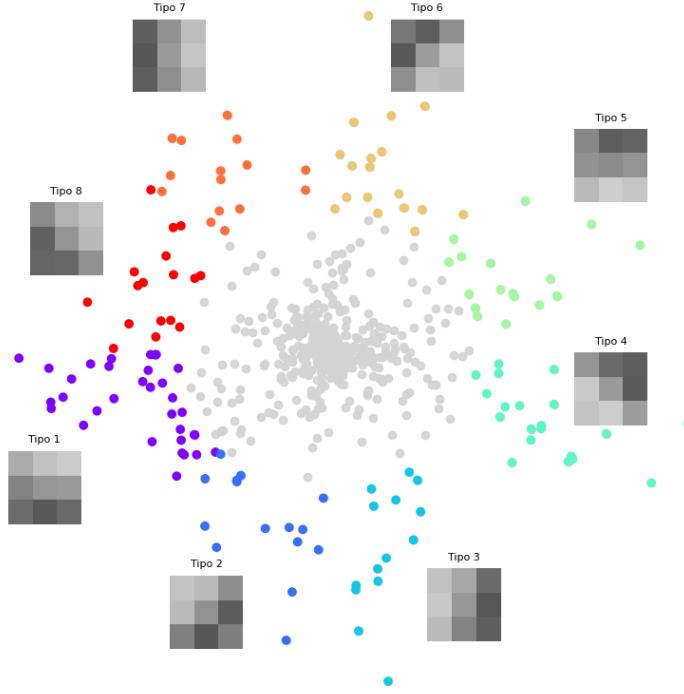


Figura 43: Proyección 2D del PCA 3D con medias por sección

Se puede observar en este gráfico la formación del círculo principal visto en la Figura 1 en las redes entrenadas con 1000 *epochs*. Si se exploran los puntos, se puede ver cómo los filtros se transforman y cómo cambian de uno a otro a medida que se acercan a los extremos de la nube de puntos.

### 5.3.2. Resultados del UMAP

Se observa en la Tabla 8 cómo tanto la *trustworthiness* como la *continuity* para el UMAP con 2 y 3 componentes aumentan levemente al pasar de 1 a 10 *epochs*, pero después de alcanzar este punto, ambas métricas se mantienen estables e incluso disminuyen ligeramente. La *trustworthiness* es alta en todos los casos teniendo de media un 0.97, lo que indica una buena preservación de la estructura original. En cambio, la media de la *continuity* es 0.47, este valor señala que algunos puntos o *clusters* que estaban agrupados correctamente en el espacio de alta dimensión no se están agrupando de manera adecuada en el espacio de baja dimensión. Se puede ver el resultado del UMAP con dos componentes según el tiempo de entrenamiento:

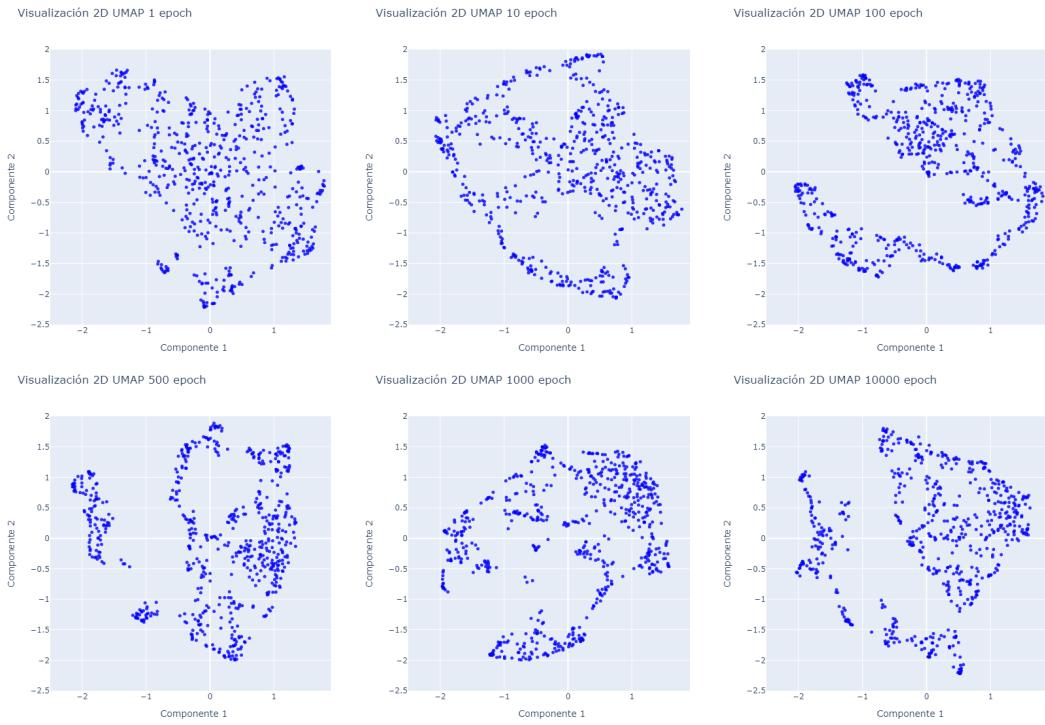


Figura 44: Evolución del UMAP con 2 componentes respecto las *epochs*

Se observa que visualmente no hay una evolución clara, ya que las gráficas son diferentes entre sí y no presentan una estructura claramente definida a simple vista.

También se puede comparar el UMAP de 3 componentes en base al tiempo de entrenamiento.

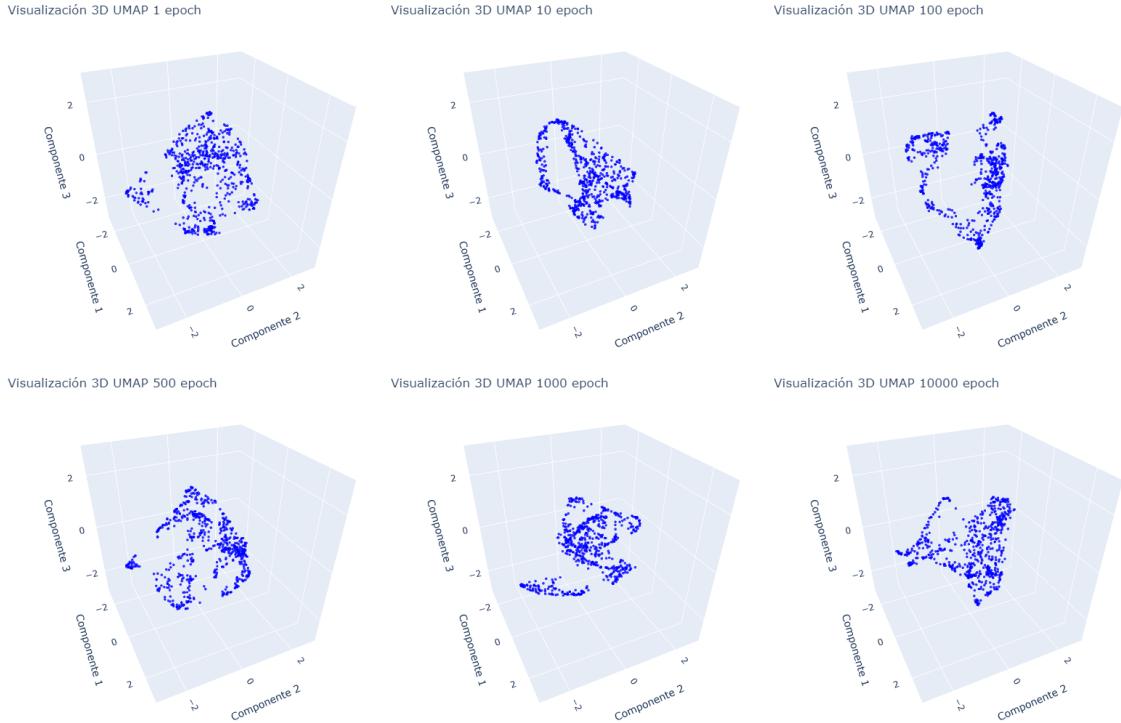


Figura 45: Evolución del UMAP con 3 componentes respecto las *epochs*

Notamos que en todos estos gráficos se forma un círculo en algún momento.

Se analiza con más profundidad la gráfica de los modelos entrenados con 10000 *epochs*, ya que con estos se observan mejor las diferencias de sus filtros, al estar más definidos.

Visualización 3D UMAP 10000 epoch

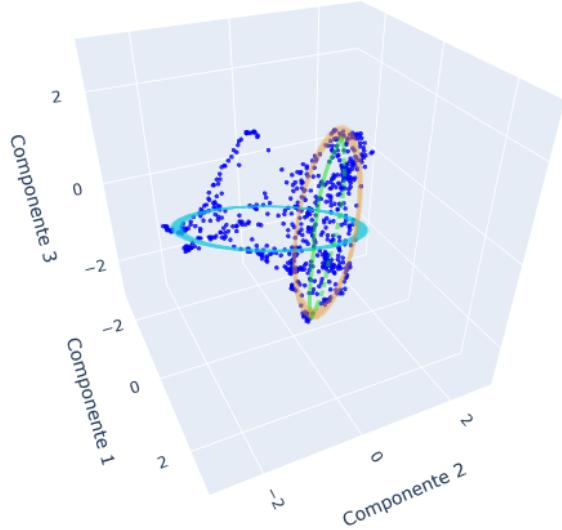


Figura 46: Círculos primario y secundarios del UMAP 3D

En este caso, se observa que el círculo principal (círculo azul) está menos definido. Sin embargo, sí se aprecia cómo se está formando lo que podría ser una esfera, representada por los círculos naranja y verde. Por lo tanto se examinará este mismo gráfico haciendo diferentes proyecciones para observar mejor estas figuras.

En cada una de las proyecciones se aplica un algoritmo DBSCAN para agrupar los filtros más cercanos y así poder calcular la media de estos, para ver gráficamente cómo están distribuidos.

La proyección sobre las componentes 1 y 2 se muestra en la siguiente figura:

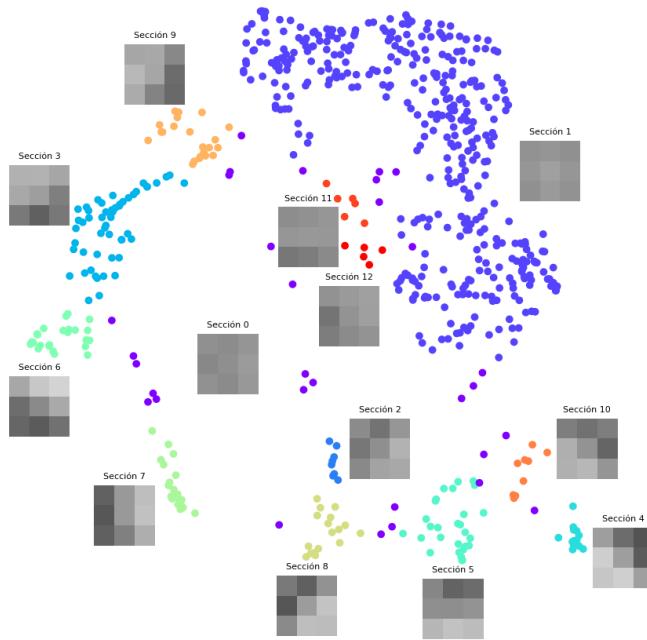


Figura 47: Proyección en los componentes 1 y 2 del UMAP 3D con 10000 epochs

En esta figura, se puede notar que en la esquina superior derecha (sección 1) los filtros son prácticamente grises, mientras que en la parte inferior derecha (secciones 2 a 10) se ven los filtros del círculo principal (Figura 1) haciendo un semicírculo en este caso.

Al hacer una proyección sobre las componentes 1 y 3 se observa el siguiente gráfico:

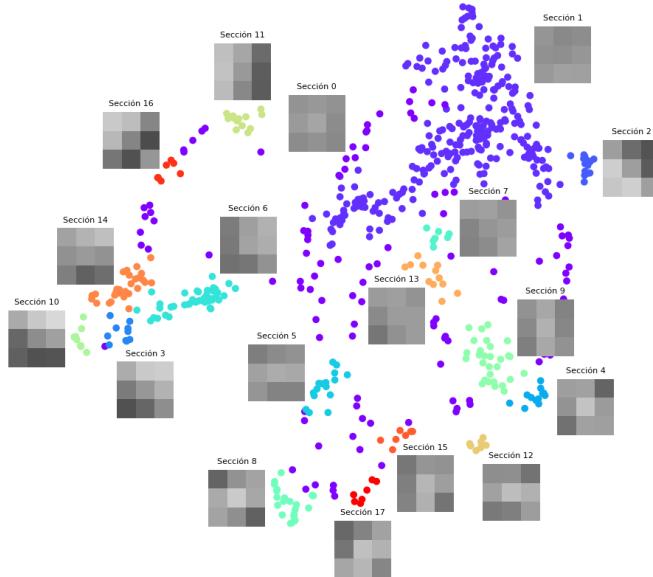


Figura 48: Proyección en los componentes 1 y 3 del UMAP 3D con 10000 epochs

En este caso, se vuelve a ver la esquina superior derecha con filtros que son prácticamente grises. También se observa de nuevo la distribución semicircular de los filtros del círculo principal (Figura 1), aunque en esta ocasión también se puede observar algunos de los filtros de los círculos secundarios en las secciones 5, 9, 15 y 12. Sin embargo, debido a la escasez de filtros, no se puede discernir

claramente ningún círculo entre estos.

Para la proyección en las componentes 2 y 3, se obtiene el gráfico:

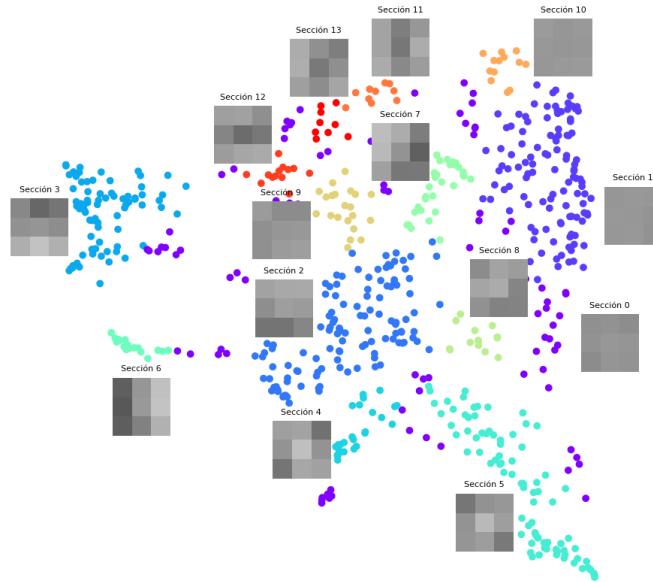


Figura 49: Proyección en los componentes 2 y 3 del UMAP 3D con 10000 *epochs*

En esta proyección se observa más grupos de filtros grises en las secciones 0, 1, 9 y 10. Además, los filtros del círculo principal no se ven tan claramente, pero en cambio sí se pueden llegar a observar otros filtros diferentes a los vistos en las otras proyecciones de los círculos secundarios en las secciones 11, 12 y 13 (Figura 1).

### 5.3.3. Resultados del Vietoris-Rips

Se observa en la Figura 32 la evolución de los filtros de una red convolucional en diferentes *epochs* durante entrenamiento. Las dimensiones H0, H1 y H2 representan, respectivamente, componentes conexas, agujeros y cavidades dentro de la estructura de datos.

En la fase inicial de entrenamiento, con 1 y 10 *epochs*, se observa una concentración de puntos cercanos al origen en H0, estos puntos representan componentes conexas de corta vida. También se observa que las dimensiones H1 y H2 muestran poca actividad, lo que indica que aún no se han formado estructuras significativas en los filtros, lo cual es esperado ya que los filtros aprendidos por la red al principio son básicos y uniformes.

A medida que se alcanzan las 100 y 500 *epochs*, se observa una tendencia en H0 hacia una mayor longevidad, llegando el punto más alto en las 500 *epochs* a (0, 1.22). H1 y H2 muestran un incremento en la cantidad y persistencia de círculos y cavidades, respectivamente, tal y como se puede observar

para las 500 *epochs* en el punto (1.015, 1.181) para H1.

Para las 1000 *epochs*, la topología de los filtros parece estabilizarse, aunque en la dimensión H0 se nota una disminución respecto al entrenamiento con 500 *epochs*. H1 demuestra más persistencia, ya que los ciclos que se crean tardan más en desaparecer, y H2 comienza a aparecer esporádicamente.

Finalmente, con 10000 *epochs*, se observa toda una línea vertical de puntos para H0, lo que indica una alta persistencia de componentes conexas, sugiriendo que la mayoría de los componentes detectados en los filtros tienen una vida útil prolongada. En H1 se observa cuatro puntos que están más alejados que los demás de la recta, siendo estos (0.9, 1.4), (1.2, 1.5), (1.4, 1.8) y (1.7, 2.3), los cuales representan los cuatro ciclos que han tenido mayor persistencia en los datos. Mientras que en H2, hay un punto que destaca del resto, siendo este (1.8, 2.0), el cual representa una cavidad que persiste en los datos.

En las dos primeras gráficas de la Figura 33, se observa la comparación entre el PCA 2D y el UMAP 2D. En el caso del PCA de dos dimensiones, se nota que los puntos correspondientes a H0 y H1 están muy agrupados cerca del origen, lo que sugiere que ni las componentes conexas ni los ciclos tienen mucha persistencia. En cambio, en el UMAP de dos dimensiones, se comienza a ver que estas formas son más evidentes en los datos.

Continuando con el análisis de las dos gráficas restantes de la misma figura, se nota que el comportamiento del PCA en dimensión tres es similar al observado en el PCA de dos dimensiones, donde los puntos se agrupan cerca del origen. Sin embargo, en el UMAP de dimensión tres, estos puntos se dispersan más respecto a la línea discontinua. Específicamente, se destacan dos puntos etiquetados como H1 que se encuentran muy alejados de este eje. Esto sugiere que mediante el uso de UMAP de dimensión tres sobre los pesos aprendidos en solo una época, ya es posible identificar dos ciclos que persisten en los datos.

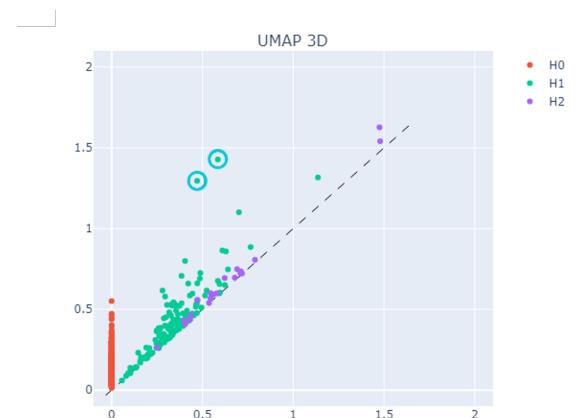


Figura 50: Vietoris-Rips para 1 *epoch* de UMAP 3D

En las dos primeras gráficas de la Figura 34, se observa que con 10000 *epochs*, el comportamiento del PCA y el UMAP se ha vuelto más similar en comparación con el análisis realizado con solo una

época. Sin embargo, el UMAP destaca por revelar un único ciclo que persiste notablemente en los datos, mientras que el PCA identifica múltiples ciclos, aunque estos no muestran una persistencia tan marcada.

En las dos últimas gráficas de la misma figura, se nota una mayor similitud entre ambas, donde cada una muestra tres ciclos que persisten claramente, como se puede observar en las representaciones gráficas correspondientes.

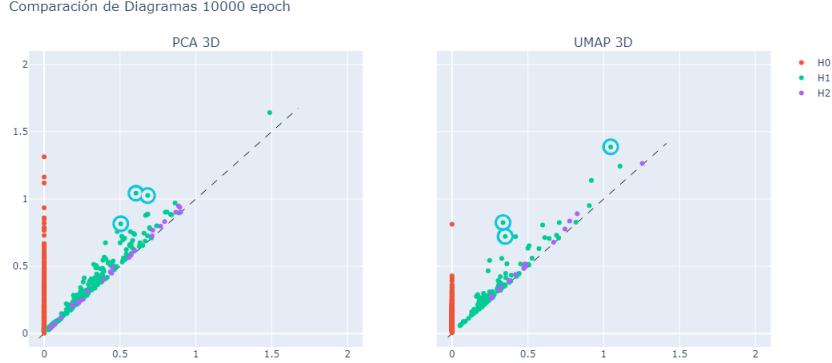


Figura 51: Vietoris-Rips para 10000 *epochs* de PCA y UMAP 3D

Adicionalmente, observamos que ninguno de los puntos etiquetados como H2 se aleja significativamente de la recta. Esto indica que no se están formando cavidades que persistan en los datos analizados.

#### 5.3.4. Resultados Mapper

Como se puede observar en la Figura 35, a medida que avanza el entrenamiento, se forman más nodos y enlaces entre ellos, dando lugar a estructuras más complejas, como se muestra en la siguiente tabla:

Epoch	Nodos	Conexiones	Muestras totales	Muestras únicas
1	286	144	740	596
10	240	332	854	628
100	253	252	849	626
500	266	312	854	617
1000	304	456	903	616
10000	427	1088	1247	630

Tabla 11: Métricas del Mapper sin reducir dimensionalidad

Analizando con más detalle el caso de las 10000 *epochs*, que presenta la estructura más compleja, se observa que no sigue ninguna tendencia general, como se puede apreciar en el siguiente gráfico:

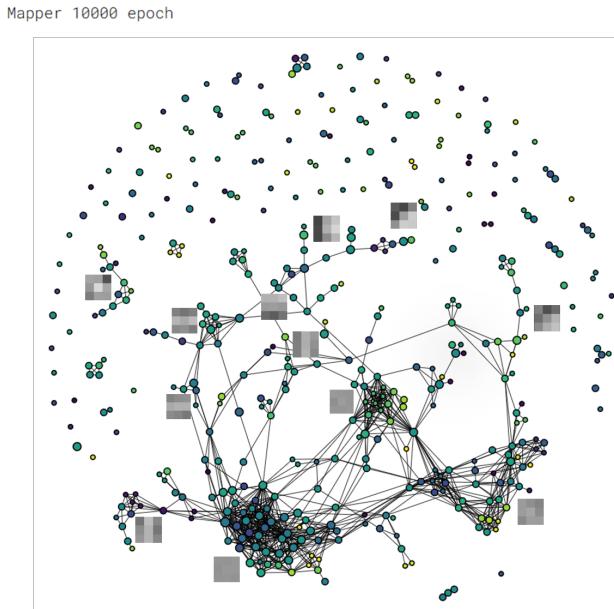


Figura 52: Mapper de 10000 *epochs* con filtros representativos

En las zonas con más nodos conectados, estos están formados por filtros prácticamente grises. En la parte superior de los nodos conectados se observa la continuidad de algunos filtros del círculo principal, mientras que más abajo a la izquierda se encuentran filtros del círculo secundario. Finalmente, el resto de los nodos son agrupaciones de filtros que pertenecen al círculo principal o secundario, pero no siguen ninguna estructura destacable.

Para el caso de la reducción de dimensionalidad con PCA de 2 componentes, se observa que a las 10000 *epochs* se forma una estructura circular (círculo rojo) donde todos los filtros que forman los nodos son prácticamente grises. Además, hay dos ramas más alejadas de este centro cuyos nodos están formados mayoritariamente por filtros de los círculos secundarios (círculos azules). En el resto de puntos se puede ver como se agrupan algunos filtros del círculo principal y secundario, aunque de nuevo, no siguen ninguna estructura destacable.

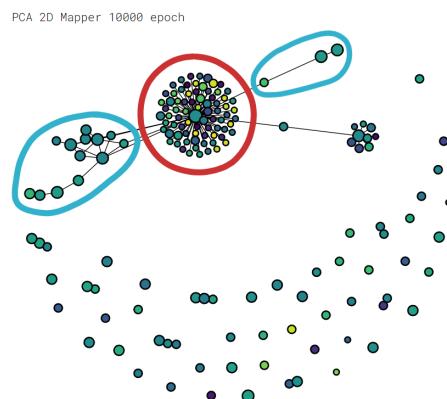


Figura 53: Mapper para el PCA 2D de 10000 *epochs* marcado

En la salida del PCA con 3 componentes, la estructura es más compleja. El centro está compuesto por los filtros más grises, mientras que las ramas que emergen de este centro no siguen una tendencia concreta, excepto las marcadas en morado y azul, donde se observa que siguen las estructuras del círculo principal y secundario, respectivamente.

PCA 3D Mapper 10000 epoch

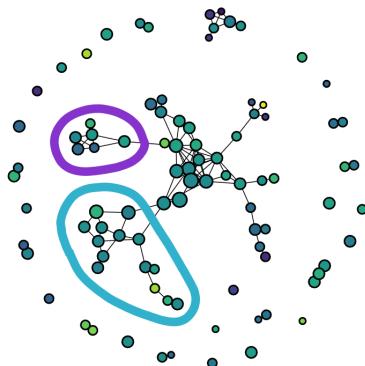


Figura 54: Mapper para el PCA 3D de 10000 epoch marcado

En los casos del UMAP con 2 y 3 componentes, se observa que las estructuras son mucho más simples, consistiendo en unos pocos nodos conectados entre sí. En ambos casos, la mayoría de los nodos están formados por filtros del círculo principal o secundario, siendo los del círculo principal mucho más visibles.

## 6. Conclusiones

En resumen, este estudio ha explorado con profundidad la evolución y estructura de los filtros en redes neuronales convolucionales utilizando técnicas como el PCA, UMAP, Vietoris-Rips y Mapper. Los filtros mostraron un aumento en definición y especificidad conforme incrementaron los *epochs*, lo que refleja una mejora en la capacidad de la red para enfocarse en características relevantes.

Algo a destacar es que en el PCA se observó claramente un círculo principal, señalando una optimización significativa en cómo los filtros procesan características visuales importantes. En contraste, aunque UMAP no mostró tan claramente el círculo principal, sí permitió vislumbrar círculos secundarios, evidenciando patrones emergentes durante el entrenamiento, siendo estos respaldados por el análisis Vietoris-Rips.

A pesar de los avances proporcionados por el PCA, UMAP y Vietoris-Rips, el análisis con Mapper enfrentó limitaciones, ofreciendo una visión más limitada y ambigua en la organización de los filtros, especialmente en la visualización clara de ciclos primarios y secundarios.

Integrar estas herramientas nos permite avanzar significativamente en nuestra comprensión de cómo las CNN procesan y clasifican eficazmente la información visual.

Ahora, para una exploración más detallada de los puntos mencionados en el resumen, abordaremos cada técnica utilizada, su aplicación específica y los resultados obtenidos a lo largo del estudio.

Este estudio detalla cómo los tiempos de ejecución y la precisión de los modelos CNN varían con el número de *epochs*. Se encuentra que la precisión mejora notablemente hasta un 99.26 % al alcanzar 100 *epochs*, y luego se estabiliza, sugiriendo que entrenar más allá de cierto punto no necesariamente resulta en modelos mejores. Paralelamente, aunque los tiempos de ejecución aumentan con más *epochs*, el uso de hardware más avanzado mejora significativamente la eficiencia, destacando la importancia de recursos computacionales adecuados para entrenamientos extensos.

En el apartado 5.2, se muestran cambios notables en los filtros aprendidos a medida que aumentaba el número de *epochs*. Los filtros se vuelven más definidos y específicos, lo que refleja una evolución en la capacidad de la red para capturar y enfocarse en características relevantes de los datos. Visualmente, los filtros muestran mayor contraste y detalle con más *epochs*, lo que sugiere que la red se adapta y mejora su enfoque en patrones esenciales para la clasificación a medida que se aumenta el tiempo de entrenamiento.

En el apartado 5.3.1, se analiza cómo el PCA refleja los cambios en la estructura de los datos de los filtros a medida que se incrementan los *epochs* de entrenamiento. Se observa que la varianza

explicada aumenta al principio, señalando una mayor dispersión en los datos con 10 *epochs*, pero luego se estabiliza y disminuye ligeramente, indicando que la mayoría de las variaciones significativas en los filtros se capturan en las primeras etapas del entrenamiento.

Visualmente, los resultados muestran que los filtros comienzan a separarse y a diferenciarse más claramente los unos de los otros a medida que aumentan los *epochs*, lo cual es consistente con la mejora en la precisión del modelo. Esto sugiere que el PCA es efectivo para capturar las diferencias fundamentales en cómo los filtros procesan y representan las características visuales en las imágenes.

Un descubrimiento notable en los análisis es la aparición de un círculo principal en los resultados de PCA, visible claramente en la Figura 43. Este círculo refleja hallazgos de estudios anteriores, como los de Gabrielsson y Carlsson en 2019 [4], quienes observaron patrones circulares similares en las redes neuronales. En este estudio, este círculo principal indica que los filtros de la red están optimizando la forma en que capturan y procesan características visuales clave. Esta formación circular puede interpretarse como una señal de que la red ha alcanzado un nivel de madurez y eficiencia, ajustando sus filtros para identificar de manera efectiva y equilibrada un rango completo de características importantes. Este comportamiento muestra cómo las redes neuronales pueden desarrollar estructuras internas complejas y significativas para mejorar su rendimiento.

El uso de UMAP para analizar la distribución espacial de los filtros en redes neuronales convolucionales a lo largo de varias etapas de entrenamiento ha revelado patrones de evolución significativos. Este método ha mantenido eficazmente las relaciones locales entre los filtros, como lo indican las consistentes métricas de *trustworthiness*. Sin embargo, la *continuity*, que mide la preservación de relaciones globales, mostró variabilidad, lo que sugiere que mientras UMAP es robusto para captar proximidades locales, podría no ser tan confiable para representar conexiones globales con la misma precisión a lo largo del tiempo.

Aunque UMAP es una herramienta potente para visualizar cómo evoluciona y se adapta el aprendizaje, los cambios en función del tiempo de entrenamiento no son tan marcados como los que se observan con el PCA. En particular, en las Figuras 46 y 47, el círculo primario es más difícil de distinguir que en los resultados obtenidos mediante PCA.

Por otro lado, las Figuras 48 y 49 revelan una mejora en la visualización de los círculos secundarios, lo que indica que en este caso UMAP ha sido efectivo para identificar patrones y estructuras menos predominantes que se desarrollan durante el entrenamiento.

El análisis mediante la homología persistente de Vietoris-Rips ofrece una visión detallada sobre la estructura topológica de los filtros en las CNN a lo largo de su entrenamiento, revelando cómo evolucionan las componentes conexas e identificando la presencia de ciclos.

A lo largo del entrenamiento, se observa que la cantidad de cavidades persistentes es limitada, lo que indica simplicidad en algunas dimensiones topológicas, mientras que los ciclos muestran una notable persistencia. En las etapas iniciales, las componentes conexas dominan, reflejando estructuras simples. Con el avance del entrenamiento, la persistencia de los ciclos aumenta, indicando un desarrollo en la complejidad de la estructura de los filtros. Este aumento sugiere que la red está refinando su capacidad para capturar y procesar relaciones más complejas y sutiles en los datos.

Es relevante destacar que, hacia el final del entrenamiento, dos o tres ciclos principales se destacan entre los demás, demostrando la formación de estructuras robustas y significativas dentro de la red, las cuales pueden tener una relación directa con los filtros de la Figura 1.

El propósito de usar la técnica de Mapper en este análisis era mostrar claramente cómo se organizan los filtros en las redes neuronales convolucionales durante su entrenamiento, enfocándonos especialmente en la identificación de ciclos primarios y secundarios. Lamentablemente, los resultados no cumplieron con lo que esperábamos.

Aunque teóricamente Mapper puede desglosar las estructuras de los datos, en la práctica solo proporcionó una idea limitada de cómo se agrupan algunos filtros. No se lograron ver estructuras claras y definidas, y las configuraciones de los ciclos primarios y secundarios terminaron siendo ambiguas y difusas, tal y como se puede notar en la Figura 52. Este resultado muestra las dificultades para configurar y usar el Mapper de manera efectiva para analizar topologías complejas en datos de alta dimensionalidad.

Comparando esto con lo que se logró con técnicas más tradicionales como PCA y UMAP, también se llevó una decepción. Aunque estas técnicas mostraron algunos patrones, la claridad y definición en los ciclos que detectó Mapper no fueron suficientes.

## 7. Recomendaciones para futuras investigaciones

A lo largo de este trabajo, se han identificado varias áreas que podrían beneficiarse de investigaciones adicionales. A continuación, se presentan algunas recomendaciones para futuros estudios:

1. **Análisis de la segunda capa convolucional:** En futuras investigaciones, se propone estudiar los pesos de la segunda capa convolucional, replicando el procedimiento empleado en los apartados anteriores. Esto permitirá profundizar en la comprensión de cómo las capas sucesivas de la red contribuyen al procesamiento y clasificación de las características visuales, y cómo estos elementos interactúan para formar representaciones más complejas.
2. **Exploración de técnicas analíticas alternativas:** Para superar las limitaciones encontradas con técnicas como Mapper, se recomienda la exploración y aplicación de nuevas herramientas analíticas que puedan ofrecer una visualización más clara y detallada de las estructuras topológicas complejas. Esto podría incluir la combinación de métodos existentes con algoritmos de aprendizaje automático avanzados para mejorar la precisión y la interpretación de los datos.
3. **Ampliación del conjunto de datos:** Para validar aún más los resultados obtenidos y generalizar las conclusiones, sería beneficioso aplicar estos métodos de análisis a conjuntos de datos más amplios y variados. Esto ayudará a determinar la robustez de los modelos y técnicas empleadas frente a variaciones más extensas en los datos de entrada.
4. **Implementación de optimizaciones en el entrenamiento:** Se sugiere investigar técnicas de optimización del entrenamiento que puedan reducir los tiempos de ejecución sin comprometer la precisión del modelo. Esto podría incluir el ajuste de parámetros como la tasa de aprendizaje o la exploración de nuevos algoritmos de optimización que puedan acelerar la convergencia.

Estas áreas no solo ampliarán el conocimiento actual sobre las redes neuronales convolucionales, sino que también potenciarán el desarrollo de modelos más eficientes y efectivos para aplicaciones prácticas en el campo del aprendizaje profundo y la visión por computadora.

## Referencias

- [1] Dot CSV. *¡Redes Neuronales CONVOLUCIONALES! ¿Cómo funcionan?* 2021. URL: [www.youtube.com/watch?v=V8j1oENVz00](https://www.youtube.com/watch?v=V8j1oENVz00).
- [2] Scikit-Learn Developers. *sklearn.manifold.SpectralEmbedding*. <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.SpectralEmbedding.html>. Documentación de Scikit-Learn. 2024.
- [3] Errodringer. *Understanding Deep Learning Techniques*. YouTube video. Video disponible en YouTube. 2023. URL: <https://www.youtube.com/watch?v=k3CWA2GBb8o>.
- [4] Rickard Brüel Gabrielsson y Gunnar Carlsson. “Exposition and Interpretation of the Topology of Neural Networks”. En: *arXiv preprint arXiv:1810.03234v3* (oct. de 2019). URL: <https://arxiv.org/pdf/1810.03234.pdf>.
- [5] *giotto-tda documentation*. <https://giotto-ai.github.io/gtda-docs/latest/index.html>.
- [6] *giotto-tda Mapper documentation*. <https://giotto-ai.github.io/gtda-docs/latest/modules/mapper.html>.
- [7] *giotto-tda Mapper quickstart notebook*. [https://github.com/giotto-ai/giotto-tda/blob/master/examples/mapper\\_quickstart.ipynb](https://github.com/giotto-ai/giotto-tda/blob/master/examples/mapper_quickstart.ipynb).
- [8] Juan Camilo Gutiérrez Díaz y Jairo Andrés Ángel Cárdenas. *Introducción al análisis topológico de datos*. Trabajo de grado de Pregrado en Matemáticas, Universidad de Los Andes, Facultad de Ciencias, Departamento de Matemáticas, Bogotá, Colombia. 2023. URL: <https://repositorio.uniandes.edu.co/server/api/core/bitstreams/b576ee28-ca67-49af-abe0-8b9dd6ad9a23/content>.
- [9] Allen Hatcher. *Algebraic Topology*. 2002. URL: <https://pi.math.cornell.edu/~hatcher/AT/AT.pdf>.
- [10] Leland McInnes, John Healy y James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. Turing Institute for Mathematics and Computing. 2020. URL: <https://arxiv.org/abs/1802.03426>.
- [11] Joan Porti. *Anàlisi Topològica de Dades*. Notes de Curs 23-24, Versió de: December 10, 2023. 2023.
- [12] Gurjeet Singh, Facundo Mémoli y Gunnar Carlsson. “Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition”. En: *Eurographics Symposium on Point-Based Graphics*. Supported by DARPA grant HR0011-05-1-0007 and NSF DMS 0354543. The Eurographics Association. Stanford University, California, USA, 2007.

- [13] Josh Starmer. *UMAP Dimension Reduction, Main Ideas!!!* YouTube video. StatQuest with Josh Starmer. 2022. URL: [www.youtube.com/watch?v=eN0wFzBA4Sc&t=916s](https://www.youtube.com/watch?v=eN0wFzBA4Sc&t=916s).
- [14] Josh Starmer. *UMAP: Mathematical Details (clearly explained!!!)* YouTube video. StatQuest with Josh Starmer. 2022. URL: <https://www.youtube.com/watch?v=jth4kEvJ3P8>.
- [15] Spyridon Stasis, Ryan Stables y Jason Hockman. “Semantically Controlled Adaptive Equa-  
lisation in Reduced Dimensionality Parameter Space”. En: *Applied Sciences* 6.116 (2016),  
págs. 1-19. DOI: 10.3390/app6040116. URL: <https://www.mdpi.com/2076-3417/6/4/116>.

## Anexos

### Anexo A: Código Fuente del Proyecto

En este anexo se proporciona el enlace al repositorio de GitHub donde se aloja el código fuente desarrollado para el proyecto del Trabajo de Fin de Grado. El repositorio incluye todos los scripts, datos y archivos de configuración utilizados.

**Enlace al Repositorio:** <https://github.com/AlbertoRQ/TFG-Analisis-topologico-para-CNN>