UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Ciencias de la Computación

Métodos de entrenamiento basados en agrupamiento para capas convolucionales en redes neuronales

Tesis presentada para optar al título de Magíster de la Universidad de Buenos Aires en
Explotación de Datos y Descubrimiento de Conocimiento

Autor: Ing. Federico Ezequiel Rabinovich
Director: Dr. Facundo Quiroga
Co-Director: Dr. Franco Ronchetti

Fecha de defensa: 12 de diciembre de 2023

Abstract

Clustering-Based Training Methods for Convolutional Layers in Neural Networks

During the course of the last decade, artificial neural networks have become the forefront of the field of artificial intelligence. Their ability to model highly complex functions has been repeatedly demonstrated through continuous state-of-the-art advancements across virtually all domains of the field. Convolutional networks, in particular, have revolutionized the field of computer vision and have been successfully employed in diverse problem domains such as natural language processing and time series modeling.

However, despite the immense progress made in a short span of time, artificial neural networks remain, in many aspects, mysterious "black boxes." Due to the large number of processing units they consist of, interpreting the results (inferences) they produce is extremely complex. Additionally, they exhibit efficiency issues, particularly during training. The computational resources required to train a network with billions of parameters are immense, resulting in significant economic and ecological impacts. Moreover, this inefficiency acts as a barrier to the widespread accessibility of this technology.

This work aims to understand the encoding of the convolutional filters in the first layer of convolutional networks and propose a novel training method for those filters. It begins with an exploratory data analysis, starting from the filters learned on ImageNet by various high-performance convolutional network models. The characteristics of the solution space reached by different models are investigated, and potential training-induced restrictions on this space are explored.

In the main body of the work, two methodologies for obtaining filters for a first convolutional layer are tested. The K-Means clustering and Principal Component Analysis (PCA) algorithms

are applied to generate the filters of the first convolutional layer of a convolutional network in an unsupervised manner. The scope of this work is limited to finding methods to train only the first layer of the network, leaving the application to subsequent layers for future research.

Two widely-used datasets in the literature, CIFAR10 and Mnist-Fashion, were employed for the experiments. Additionally, two convolutional network models were used throughout all the experiments. One of them, SimpleConv, was ad-hoc constructed by the thesis author. The other model used is the widely accepted and utilized EfficientNet V2 Small.

The results were evaluated comparatively to the traditional training method of Gradient Descent and Backpropagation for the two datasets and the two mentioned architectures. Finally, a series of confirmatory experiments were conducted, analyzing the parameters generated by the two alternative training algorithms. Additionally, the changes experienced by the filters during the Gradient Descent and Backpropagation training process were examined, conditioned by the characteristics of the algorithm used for their initialization.

This work introduces two novel training methods for the first layer filters of a convolutional network. The experiments demonstrate that both methods produce suitable filters for a first layer in the image classification task on the tested configurations. Under these circumstances, both methods achieve superior performance compared to the traditional Gradient Descent training method.

Resumen

Durante el transcurso de la última década, las redes neuronales artificiales se han convertido en la punta de lanza del campo de la inteligencia artificial. Su capacidad para modelar funciones de altísima complejidad ha quedado demostrada una y otra vez con sus permanentes avances del estado del arte en prácticamente todos los dominios del área. Las redes convolucionales en particular, han revolucionado el campo de la visión artificial y han sido empleadas con éxito en problemáticas tan disímiles como el procesamiento de lenguaje natural y el modelado de series temporales.

No obstante los inmensos avances producidos en tan pocos años, las redes neuronales artificiales continúan siendo, en muchos aspectos, misteriosas "cajas negras". Debido a la cantidad de unidades de procesamiento que las conforman, la interpretación de los resultados (inferencias) que producen resulta extremadamente compleja. Adicionalmente, presentan problemas de eficiencia, particularmente durante su entrenamiento. Los recursos computacionales necesarios para entrenar una red de miles de millones de parámetros son inmensos, generando un impacto económico y ecológico significativos. Asimismo, esta ineficiencia actúa como limitante al libre acceso a esta tecnología.

Este trabajo tiene el objetivo de comprender la codificación de los filtros convolucionales de la primera capa de las redes convolucionales, y proponer un nuevo método de entrenamiento para los filtros de esa capa.

Se comenzó por realizar un análisis exploratorio de datos, tomando como punto de partida los filtros aprendidos sobre ImageNet por diversos modelos de redes convolucionales de alto rendimiento. Se indagó sobre las características del espacio de soluciones a las que los diferentes modelos arribaron y se exploraron posibles restricciones sobre este espacio producto del entrenamiento al que fueron expuestos.

En el cuerpo principal del trabajo, se ensayaron dos metodologías para la obtención de filtros para una primera capa convolucional. Se aplicaron los algoritmos de agrupamiento K-Means y de reducción de dimensionalidad PCA para generar los filtros de la primera capa convolucional de una red convolucional, de forma no supervisada. El alcance de este trabajo se limitó a la búsqueda de métodos para entrenar únicamente la primera capa de la red, dejando para futuros trabajos la aplicación a capas subsiguientes.

Para realizar los experimentos, se emplearon dos conjuntos de datos ampliamente utilizados en la literatura: CIFAR10 y Mnist-Fashion. Asimismo, se tomaron, a lo largo de todos los experimentos, dos modelos de redes convolucionales. Uno de ellos, SimpleConv, fue construido de forma ad-hoc por el tesista. El otro modelo utilizado es EfficientNet V2 Small, de amplia aceptación y utilización en la literatura.

Se evaluaron los resultados de forma comparativa al método tradicional de entrenamiento por Descenso por Gradientes y Backpropagation para los dos conjuntos de datos y las dos arquitecturas mencionadas.

Finalmente se realizó una serie de experimentos confirmatorios, llevando a cabo un análisis de los parámetros generados por los dos algoritmos alternativos de entrenamiento. Asimismo, se analizaron los cambios experimentados por los filtros durante el proceso de entrenamiento por Descenso por Gradientes y Backpropagation, condicionados a las características del algoritmo utilizado para su inicialización.

Este trabajo aporta dos métodos novedosos de entrenamiento para la primera capa de filtros de una red convolucional. Los experimentos demuestran que ambos métodos producen filtros adecuados para una primera capa en la tarea de clasificación de imágenes sobre las configuraciones ensayadas. Bajo estas circunstancias, ambos métodos obtienen desempeños superiores a los obtenidos por el método tradicional de entrenamiento de descenso por gradientes.

Agradecimientos

En primer lugar agradezco el infinito apoyo de mi mujer Betina y de mi hijo Santiago a lo largo de todo el proceso de desarrollo de este trabajo. Siempre estuvieron cerca, con amor incondicional.

Adicionalmente, estoy sumamente agradecido a mis directores de tesis Facundo Quiroga y Franco Ronchetti por creer en mí y en el proyecto. Sin su ardua colaboración, ideas, discusiones, propuestas y revisiones, esta tésis nunca hubiera sido.

Finalmente, me gustaría agradecer a los numerosos, conocidos y anónimos investigadores del área de Inteligencia Artificial, que durante más de 70 años hicieron sus aportes para que hoy podamos estar cosechando este increíble estado del arte.

Índice

1. Introducción 10
1.1. Motivación 10
1.2. Objetivos 12
1.3. Organización 12
2. Marco teórico 14
2.1. Técnicas de agrupamiento no supervisado y de validación 15
2.1.1. K-Medias (algoritmo de Lloyd–Forgy 15
2.1.2. DBScan 16
2.1.3. Agrupamiento aglomerativo 17
2.1.4. Métrica de Silhouette 17
2.1.5. DBCV 18
2.2. Redes neuronales 19
2.2.1. Entrenamiento de redes neuronales 21
2.3. Redes convolucionales 24
2.3.1. Operación de convolución 26
2.3.2. Las redes convolucionales y la extracción de información visual 29
2.4. Estrategias de inicialización de redes neuronales 31
2.4.1. Inicialización Glorot 32
2.4.2. Inicialización He 33
2.4.3. Inicialización ortogonal 34
2.5. Métodos supervisados de entrenamiento para CNNs alternativos (a GDBP) 35
2.6. Estrategias de entrenamiento no supervisado de CNNs con empleo de GDBP 36
2.7. Métodos alternativos para el entrenamiento de filtros para capas convolucionales 39
2.7.1. Estrategias de entrenamiento e inicialización basadas en PC 40
2.7.2. Estrategias de entrenamiento e inicialización basadas en K-Medias 43

2.8. Métodos de optimización de CNNs a través de la poda de filtros entrenados 44
2.9. Conjuntos de datos: Mnist-Fashion, CIFAR10, ImageNet 45
2.9.1. CIFAR10 45
2.9.2. Mnist-Fashion 46
2.9.3. ImageNet 47
2.10. Modelos Convolucionales: Efficient Net 48
3. Exploración de filtros convolucionales en arquitecturas de alto desempeño 53
3.1. Métrica de similitud 54
3.1.1. Conjunto de datos empleados 55
3.1.2. Preprocesamiento 58
3.1.3. Experimentos realizados 58
3.1.4. Similitud entre filtros a partir de los mapas de características 60
3.1.5. Similitud directa y similitud de mapas de características 62
3.2. Agrupamiento de filtros 64
3.2.1. Agrupamiento por medio de DBScan 65
3.2.2. Agrupamiento por medio de agrupamiento aglomerativo 69
3.2.3. Observaciones y conclusiones de la sección 71
4. Métodos de obtención de filtros para capas convolucionales 74
4.1. Método de obtención de filtros convolucionales a través de Componentes Principales 77
4.1.1. Preprocesamiento 78
4.1.2. Entrenamiento de los filtros 80
4.1.3. Post Procesamiento 81
4.2. Método de obtención de filtros convolucionales a través de K-Medias 82
4.2.1. Preprocesamiento 83
4.2.2. Entrenamiento de los filtros 83
4.2.3. Post Procesamiento 84
4.3. Método de obtención de filtros convolucionales a través de parches aleatorios 84
4.3.1. Preprocesamiento 84
4.3.2. Entrenamiento de los filtros 85
4.3.3. Post Procesamiento 85
5. Evaluación de los métodos 86
5.1. Objetivos 87
5.2. Criterios para la realización de los experimentos 88
5.3. Criterios de evaluación 89
5.4. Resultados obtenidos para los métodos de entrenamiento de filtros propuestos 90
5.5. Filtros obtenidos por medio de los diferentes métodos propuestos 98

5.6. Resultados comparativos para el entrenamiento de filtros por medio de K-Medias y K-Medoides 102
5.7. Impacto del conjunto de datos en el entrenamiento de filtros 104
5.7.1. Impacto del tamaño de muestra en el entrenamiento de filtros 104
5.7.2. Impacto del balance de clases en el entrenamiento de filtros 109
5.7.3. Impacto del conjunto de datos en el entrenamiento de filtros 111
5.8. Conclusiones 112
6. Conclusiones y futuros trabajos 114
6.1. Análisis exploratorio de filtros convolucionales en modelos pre entrenados 115
6.2. Métodos de entrenamiento de filtros 117
Epílogo 120
Bibliografía 121
Apéndice 126
Arquitectura SimpleConv 126
Arquitectura EfficientNet 128
Hiperparámetros utilizados en los entrenamientos 129

1. Introducción

1.1. Motivación

Desde el año 2012, las arquitecturas de redes neuronales basadas en capas convolucionales han logrado las mejores métricas de rendimiento en diferentes tareas del área de Visión Artificial. Nuevas arquitecturas han ido surgiendo en estos últimos diez años, logrando superar año tras año las métricas establecidas por el estado del arte, alcanzando o superando incluso el desempeño humano en varias tareas [6] [10].

A pesar de los logros obtenidos por estos modelos, en muchas ocasiones se ha puesto el foco desde la comunidad académica sobre su falta de interpretabilidad [47]. Este punto no solo concierne a las redes convolucionales en particular sino que abarca al conjunto de las redes neuronales artificiales. Sus características intrínsecas de capas lineales seguidas por funciones no lineales apiladas las convierte en verdaderas cajas negras, en donde solo resultan directamente interpretables las entradas y salidas del modelo.

El entrenamiento por medio de Descenso por Gradientes y Backpropagation (GDBP, de aquí en adelante) implica la obtención de un conjunto de soluciones (los parámetros aprendidos) de escaso valor interpretativo. El algoritmo persigue la optimización de una función objetivo y para ello actualiza los parámetros del modelo de modo iterativo. De producirse la convergencia, las soluciones encontradas por este mecanismo son simplemente aquellas que permiten minimizar localmente el error de dicha función. De forma directa no aportan ninguna interpretabilidad sobre los mecanismos subyacentes que permiten al modelo discriminar entre una u otra clase. Es por ello que son considerados por gran parte de la comunidad académica como verdaderas cajas negras.

Cabe mencionar que en muchas aplicaciones la interpretabilidad de los resultados obtenidos por modelos estadísticos y de aprendizaje automático puede llegar a ser un factor decisivo al momento de decidir su adopción en ambientes productivos.

Por otra parte, y como prerrequisito para el desarrollo de métodos alternativos de entrenamiento, resulta inevitable, realizar diversos análisis exploratorios sobre modelos ya entrenados por medio tradicionales. Por tal debe aquí entenderse al ampliamente utilizado GDBP para el entrenamiento y al muestreo aleatorio (como lo son los métodos de inicialización Glorot [10] y He [9]) para la inicialización de los parámetros.

El objetivo de esta etapa exploratoria es el de poder extraer conclusiones acerca de los parámetros aprendidos por los modelos por métodos tradicionales de entrenamiento. De este modo, se pretende también establecer puntos de comparación para los métodos de entrenamiento a ensayar. Las métricas de desempeño obtenidas por GDBP permitirán evaluar cada uno de los métodos ensayados, destacando de forma clara los métodos alternativos más promisorios. En definitiva, lo que se intenta es replicar los resultados obtenidos por GDBP, por medios completamente diferentes.

Estos análisis exploratorios permiten ganar en comprensión y en profundidad respecto al funcionamiento interno de las redes convolucionales. Si bien no se trata de un objetivo de impacto inmediato, poder aportar al entendimiento detallado de los mecanismos que gobiernan las redes convolucionales representa un gran aporte potencial para trabajos futuros en el área.

Más allá de facilitar la interpretabilidad de los modelos, existen diversas motivaciones adicionales para encontrar un método alternativo de entrenamiento a GDBP así como prescindir de un conjunto de datos etiquetado.

Por un lado, es significativa la cantidad de recursos computacionales y energéticos que demanda el entrenamiento por GDBP. Los grandes modelos de miles de millones de parámetros que logran avances en el estado del arte son grandes consumidores de energía y poder de procesamiento. Pueden llegar a requerir al momento de entrenamiento, miles de placas de procesamiento gráfico (GPU) y consumir el equivalente a la energía necesaria para una ciudad de medianas dimensiones [48]. Esto no solo genera un impacto ambiental directo sino que también limita el acceso a la investigación y desarrollo a unas pocas compañías que cuentan con la infraestructura y recursos necesarios.

Adicionalmente, la necesidad de contar con instancias de entrenamiento etiquetadas implica altos costos de preprocesamiento manual de los datos y presenta desafíos insuperables en cuanto a la escalabilidad de los modelos. Poder reducir o incluso eliminar la necesidad de conjuntos de datos etiquetados para entrenar un modelo daría acceso al entrenamiento de

nuevos modelos. Permitiría no solo reducir costos de entrenamiento, sino también entrenar modelos sobre una amplísima diversidad de conjuntos de datos que actualmente no están etiquetados.

1.2. Objetivos

El objetivo del presente trabajo es ensayar métodos de entrenamiento e inicialización de parámetros alternativos a GDBP para redes convolucionales. Tomando dos arquitecturas convolucionales, una de ellas denominada "SimpleConv" construida de forma ad-hoc por el tesista y EfficientNet V2 Small, de amplia aceptación en la literatura, se ensayaron diferentes métodos alternativos de entrenamiento. Los métodos se basaron en la distribución de los datos de entrada (conjuntos de imágenes) sin emplear anotaciones. Es decir, se trata de procedimientos de entrenamiento no supervisados. Los recursos computacionales necesarios para la ejecución de estos métodos resultan notoriamente inferiores a los necesarios para el entrenamiento por medio de GDBP. Esto implica la posibilidad de realizar entrenamientos mucho más eficientes.

Alternativamente al objetivo de entrenamiento, se buscó obtener, por medio de los mismos métodos de entrenamiento propuestos, métodos de inicialización para los parámetros de la red que produzcan resultados comparables o superiores a los métodos de inicialización por muestreo aleatorio (Glorot, He).

Adicionalmente, este trabajo hace un aporte a la interpretabilidad de los parámetros aprendidos por los modelos de redes convolucionales. Asimismo, busca aportar a la comprensión de varias particularidades del entrenamiento por GDBP de este tipo de redes artificiales. Estos objetivos adicionales se plantearon como necesarios y accesorios al objetivo principal y permitieron ganar en entendimiento respecto a las características de los filtros y la distribución de los datos de entrada.

1.3. Organización

Este trabajo está organizado en 6 capítulos donde se abordan las siguientes temáticas:

- **Capítulo 1: Introducción.** Planteo de los objetivos del trabajo y la motivación para dichos objetivos, en el marco del estado actual del arte para modelos convolucionales.
- **Capítulo 2: Marco teórico.** En este capítulo se desarrollan los conceptos teóricos fundamentales sobre los que se apoyan los experimentos realizados en este trabajo. Se contextualizan los modelos convolucionales dentro del marco del aprendizaje automático y las redes neuronales artificiales y se profundiza en el funcionamiento y métodos de entrenamiento para redes convolucionales.
- **Capítulo 3: Exploración de filtros convolucionales en arquitecturas de alto desempeño.** Se exploran las características de los parámetros obtenidos para la primera capa de diversos modelos convolucionales entrenados por medio de GDBP. Asimismo, se selecciona una métrica de similitud entre filtros y se la utiliza para generar agrupamientos no supervisados de los filtros de los modelos pre entrenados.
- **Capítulo 4: Métodos de obtención de filtros para capas convolucionales.** Se desarrollan las diferentes metodologías utilizadas para el entrenamiento de filtros convolucionales. Se plantean las premisas que apoyan una u otra propuesta de entrenamiento, así como los objetivos y criterios de evaluación para los experimentos.
- **Capítulo 5: Evaluación de los métodos.** En este capítulo se realiza la evaluación de los diferentes métodos de obtención de filtros, comparándolos en términos de desempeño, con el método tradicional de entrenamiento por GDBP. Se evalúa también el impacto de algunos hiperparámetros de los diferentes métodos en el desempeño general de la red.
- **Capítulo 6: Conclusiones.** En este capítulo se desarrollan las conclusiones extraídas de los experimentos, así como las direcciones más promisorias hacia donde orientar futuros trabajos.

Esta tesis surge de mis intereses en el campo de las redes neuronales artificiales y su aplicabilidad en diferentes áreas de visión artificial. Esperamos que sea una lectura estimulante y que el lector encuentre en estas líneas un aporte original en el desarrollo de nuevas técnicas dentro del campo de la inteligencia artificial.

2. Marco teórico

El presente trabajo se enmarca dentro de la disciplina del aprendizaje automático. Esta disciplina presenta un paradigma completamente diferente al tradicional para lograr el aprendizaje en máquinas. El paradigma tradicional de programación consiste en el establecimiento de reglas de decisión por parte de un operador humano. Luego estas reglas serán aplicadas por el sistema de cómputo a la hora de tomar decisiones en tiempo de ejecución [2].

Contrariamente, en el marco del paradigma de aprendizaje automático, estas reglas de decisión son inferidas de forma automática por el sistema de cómputo, a partir de determinado conjunto de datos con que se lo provee. Este proceso donde la máquina realiza su propio proceso de generación de reglas de decisión a partir de datos, es referido como “aprendizaje automático”.

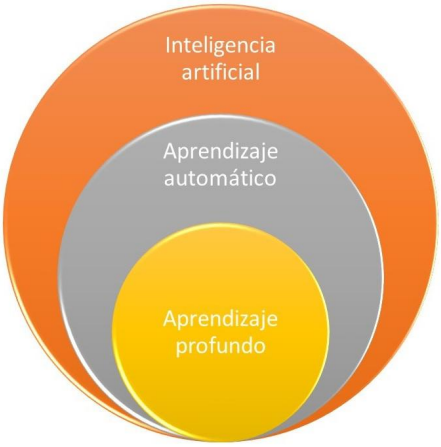


Fig 2.1: Las redes convolucionales son un tipo de redes neuronales artificiales que se inscriben dentro del marco teórico del aprendizaje automático.

En el marco del aprendizaje automático, se entiende que un agente ha logrado “aprender” cuando logra mejorar su desempeño en determinada tarea, a partir de la incorporación de nuevos datos o nuevas experiencias. Es decir, su desempeño sobre determinada tarea no está fijada al momento de creación del agente o del algoritmo, sino que éste logra “aprender” durante el proceso de incorporación de nueva información.

El campo del aprendizaje automático se suele dividir en 3 tipos de aprendizaje: supervisado, no supervisado y por refuerzo. Si bien existen muchas variantes que combinan dos o más de estos tipos (aprendizaje semi-supervisado, auto-supervisado, etc), se utiliza esta categorización por simplicidad didáctica.

Por su parte, el aprendizaje supervisado consiste en la utilización de las etiquetas o anotaciones de las observaciones para lograr el aprendizaje del algoritmo o modelo. Los ejemplos anotados son instancias o puntos de datos que usualmente se corresponden con la entidad que se representa en una fila de una tabla de determinado conjunto de datos. El hecho de estar anotados o etiquetados refiere a la disponibilidad de un valor o conjunto de valores ya definidos previamente (usualmente por un anotador humano) para cada una de las instancias del conjunto de datos. Una vez entrenado el modelo o algoritmo, son estos tipos de datos los que se pretenderá obtener a su salida. Es decir, lograr que el algoritmo genere anotaciones (predicciones) sobre instancias no observadas durante el proceso de entrenamiento.

En el marco del aprendizaje no supervisado, en cambio, se le presentan ejemplos al algoritmo o modelo que no están etiquetados. Es decir, no se le provee un valor esperado de salida para los ejemplos con los que se lo entrena, sino que se pretende encontrar patrones en el conjunto de datos que permitan agruparlos con determinado criterio. Para ello se utilizan una variedad de algoritmos de agrupamiento no supervisado [1].

En el presente trabajo, se hace uso tanto de técnicas de aprendizaje supervisado como no supervisado.

2.1. Técnicas de agrupamiento no supervisado y de validación.

En esta sección se desarrollan brevemente las técnicas de agrupamiento utilizadas en el presente trabajo, así como las técnicas para validar los resultados de agrupamientos no supervisados.

2.1.1. K-Medias (algoritmo de Lloyd–Forgy)

K-Medias es probablemente uno de los algoritmos de agrupamiento más utilizados y se basa en la búsqueda de los centroides que minimizan la distancia euclídea a las observaciones adjudicadas a cada grupo.

El algoritmo requiere la definición previa de la cantidad de grupos que se pretende encontrar. Comienza inicializando en posiciones al azar tantos centroides como grupos se han definido. Adjudica cada una de las observaciones al centroide más cercano y recalcula la posiciones de estos centroides para minimizar la distancia total euclídea de cada uno de ellos con las observaciones que pertenecen a su grupo. Este proceso se repite, reasignando observaciones y recalculando centroides, hasta que los centroides dejan de ser movidos o reubicados, de acuerdo a determinado umbral de tolerancia. Debido a la naturaleza aleatoria de las inicializaciones, se suelen realizar varias corridas del algoritmo, seleccionando aquella en la que los centroides mejor representan cada agrupamiento (con menor distancia total entre cada centroide y las observaciones del grupo).

Existen muchas variaciones de este algoritmo, pero quizá la más conocida y utilizada en el presente trabajo es K-medoides. Esta variación utiliza la mediana como estadístico, en lugar de la media, para realizar el cálculo de posición de cada uno de los centroides (denominados "medoides" en esta variante) [2].

2.1.2. DBScan

DBScan es también probablemente uno de los algoritmos de agrupamiento más utilizados y se diferencia principalmente de K-Medias por la diversidad de formas espaciales que los grupos pueden adoptar.

El tipo de agrupamientos que obtiene DBScan no está basado directamente en la distancia entre observaciones sino en la densidad de éstas en el espacio. A diferencia de K-Medias, DBScan no requiere determinar a priori la cantidad de grupos a encontrar, sino que forma grupos allí donde el espacio está densamente poblado de observaciones.

Otra característica adicional de este algoritmo es que contempla la existencia de ruido, clasificando así a aquellas observaciones que no se encuentran en regiones pobladas del espacio. DBScan categoriza las observaciones de acuerdo a tres clases:

- Puntos nucleares: son aquellos puntos que tienen en su vecindario (definido por un umbral de distancia) al menos una cantidad mínima (también predefinida) de puntos.
- Puntos alcanzables: son aquellos puntos que se encuentran en el vecindario de un punto nuclear, es decir, a una distancia menor al umbral de distancia.

- Puntos inalcanzables o “ruido”: aquellos puntos que se encuentran a una distancia mayor al umbral del punto nuclear más cercano.

Cada agrupamiento contiene al menos un punto nuclear. Si dos o más puntos nucleares son alcanzables entre sí, o se puede formar un recorrido entre puntos nucleares con saltos que nunca excedan el umbral de distancia, estos puntos nucleares conforman un mismo agrupamiento.

Los dos parámetros fundamentales para este algoritmo son la cantidad mínima de observaciones para definir la presencia de puntos nucleares y el umbral de distancia. Adicionalmente y de acuerdo a la implementación específica de que se trate, suelen ofrecerse paŕametros adicionales como la métrica de distancia a utilizar o qué implementación del algoritmo emplear [2].

2.1.3. Agrupamiento aglomerativo

El algoritmo de agrupamiento aglomerativo es de tipo jerárquico. El algoritmo parte de las instancias individuales, donde cada instancia se corresponde con un grupo y fusiona progresivamente los grupos más cercanos. Las observaciones son agrupadas de forma progresiva, generando agrupamientos cada vez más grandes. En cada paso del algoritmo, se realiza una fusión entre los grupos más próximos.

Los parámetros más importantes para este algoritmo son el tipo de enlace, la métrica de distancia a emplear y el umbral de distancia donde el algoritmo debe detenerse.

Existen varias posibilidades para el tipo de enlace a emplear:

- Ward: busca minimizar la suma de los cuadrados de las diferencias dentro de los grupos (similar a K-Medias)
- Enlace máximo o completo: minimiza la distancia máxima entre observaciones de pares de grupos
- Enlace promedio: minimiza el promedio de distancias entre todas las observaciones de pares de grupos
- Enlace único: minimiza la distancia entre las observaciones más cercanas de pares de grupos [2].

2.1.4. Métrica de Silhouette

Silhouette es un método y una métrica para validar la consistencia de los agrupamientos generados, por lo general, por técnicas de agrupamiento no supervisado.

Este método se basa en los conceptos de cohesión y separación de cada observación del conjunto de datos. Obtiene un valor para cada instancia, de acuerdo a qué tan similar es dicho objeto al grupo al que se lo ha asignado y qué tan diferente es al grupo más cercano al que no pertenece. Este valor está comprendido en el rango [-1, 1], donde pueden interpretarse los valores negativos como una mala asignación de grupo (la observación es más similar a una grupo ajeno que al propio). A mayor valor, mejor asignada la observación.

Silhouette obtiene la medida de cohesión de determinada instancia al grupo al que fue asignado por medio del cálculo de la distancia media de dicho punto al resto de los puntos del grupo. En cuanto a la separación, realiza un cálculo idéntico, pero respecto al agrupamiento más cercano (grupo vecino). Es decir, calcula la distancia media de la observación respecto a todas las observaciones del grupo ajeno más cercano.

Probablemente la mayor desventaja de esta métrica es que suele obtener valores mayores para agrupamientos convexos y menores para aquellos basados en densidad, como es el caso de los obtenidos por DBScan [2].

2.1.5. DBCV

DBCV o validación de agrupamientos basados en la densidad, por sus siglas en inglés, es un método y una métrica para validar el resultado de agrupamientos. Se diferencia fundamentalmente de Silhouette por su capacidad para realizar una correcta validación de agrupamientos con formas diversas, sin estar sesgado hacia agrupamientos de tipo convexo.

Permite obtener valores dentro del mismo rango que Silhouette [-1, 1], donde a mayor valor, mejor calidad del agrupamiento. A diferencia de Silhouette, evalúa tanto la cohesión de cada grupo así como la separación entre grupos considerando la densidad de las regiones con puntos y no la distancia entre ellos. El score final es obtenido como la suma ponderada del score de validación de cada agrupamiento.

Es por estos motivos que DBCV resulta más apropiado para evaluar la calidad de agrupamientos realizados por métodos basados en densidad, como es el caso de DBScan. Adicionalmente, DBCV contempla la existencia de ruido en el conjunto de datos. Es decir, observaciones etiquetadas por el algoritmo de agrupamiento como no pertenecientes a ningún grupo. Considera como buenos agrupamientos a aquellos donde la densidad más baja dentro de cada grupo es mayor a la densidad más alta de zonas con observaciones categorizadas como ruido.

Su capacidad para el manejo de instancias catalogadas como ruido así como su valoración no sesgada por la forma que puedan tomar los agrupamientos lo vuelve un candidato ideal para la valoración de agrupamientos realizados por métodos como DBScan [3].

Habiendo repasado los principales métodos de clustering utilizados en esta tesis para agrupar filtros convolucionales, a continuación se presentan los modelos de redes neuronales, y en particular las Convolucionales, para los cuales se propone un método de entrenamiento y se analizan en esta tesis.

2.2. Redes neuronales

Las redes neuronales artificiales se inscriben dentro del paradigma de aprendizaje automático. Son una de las muchas estrategias creadas para lograr el aprendizaje automático. Su origen histórico se remonta a la década del 50 del siglo XX, con la creación del Perceptrón por parte de F. Rosenblatt [4] inspirado en los mecanismos de activación de las células neuronales. Posteriormente esta idea fue ampliada a la del Perceptrón multicapa para poder ser aplicada en conjuntos de datos no linealmente separables. La arquitectura tradicional de un Perceptrón multicapa consta de 2 o más capas, cada una de ellas con 1 o más unidades, y mediando entre ellas una función de activación no lineal.

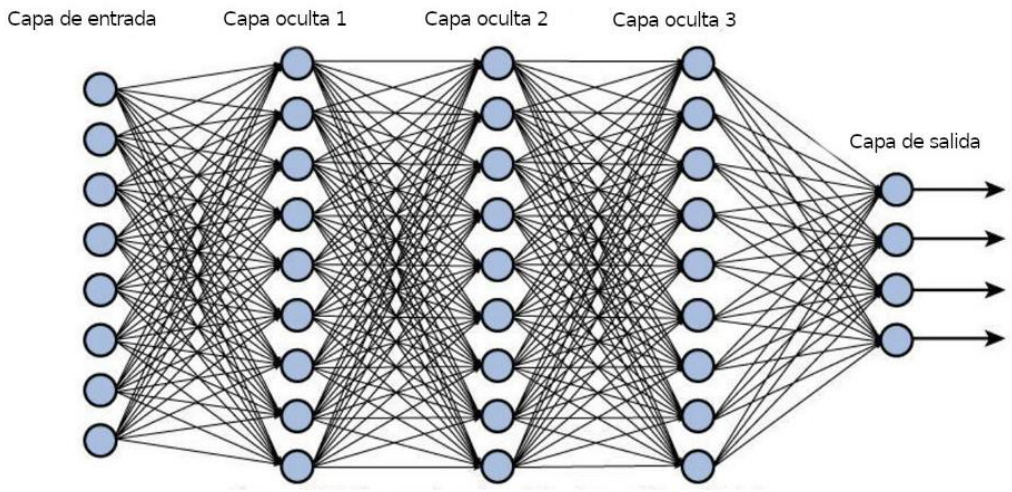


Fig 2.2: Perceptron multicapa.

Cada unidad de cada capa (ilustrados en la figura como círculos celestes) está conectada con las salidas de todas las unidades de la capa anterior. A través de un vector de pesos de

dimensionalidad igual a la cantidad de conexiones entrantes, la unidad pondera cada una de las entradas para luego realizar una sumatoria de todos los productos realizados.

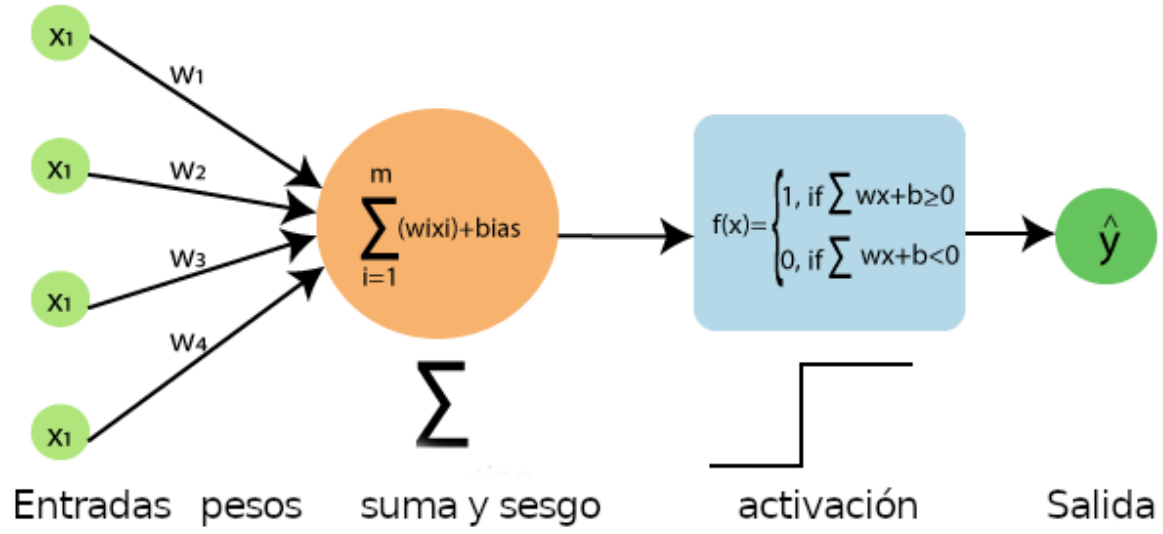


Fig 2.3: Operaciones realizadas internamente por cada unidad de un perceptrón multicapa.

Para cada entrada que recibe de la capa anterior, la unidad posee un peso establecido por el que multiplica el valor de dicha entrada. Cada unidad posee también un valor de sesgo fijo que no depende de ninguna entrada y que también está incluido en la sumatoria. Finalmente, el valor resultante es pasado por una función de activación no lineal y está será la salida que la unidad exponga a las unidades de la siguiente capa.

El rol de la función de activación es el de permitir a la red en su conjunto, construir una función compleja que proyecte el espacio de entrada en un espacio de salida de forma no lineal. Sin la presencia de funciones de activación no lineales entre capas, el apilado de capas lineales puede ser reducido a una mera transformación lineal. La complejidad de la función que puede adoptar un perceptrón multicapa está fuertemente condicionada por la cantidad de parámetros o pesos que posea, es decir, por la cantidad de unidades y capas que lo conforman.

La función de activación más utilizada en la mayoría de las arquitecturas es la función ReLU, pero puede utilizarse cualquier función que cumpla con las condiciones de ser no lineal y derivable. Esto último resulta fundamental para poder realizar el entrenamiento de la red por medio de los algoritmos de Backpropagation y Descenso por Gradientes.

A la arquitectura ilustrada en la figura 2.2 actualmente se la denomina también red neuronal artificial y es la estructura básica sobre la que se ha desarrollado la disciplina de Aprendizaje

Profundo o *Deep Learning*. El Aprendizaje Profundo es una más de las técnicas y algoritmos que componen el corpus teórico del Aprendizaje Automático, tal como se ilustra en la figura 2.1.

Desde las últimas décadas del siglo XX, se han ido desarrollando nuevas y diversas tipologías de redes neuronales artificiales, cada una con sus particularidades. Existen tipologías de conjuntos de datos bien diferenciadas, con los que se pretende alimentar a los modelos de Aprendizaje Automático. Un conjunto de datos puede consistir en un conjunto de imágenes, en datos tabulares o en una serie temporal, por ejemplo. Cada una de estos tipos de datos posee sus particularidades y las diversas arquitecturas de redes neuronales que han ido surgiendo pretenden explotar dichas particularidades.

Para el caso de series temporales, por ejemplo, se han desarrollado diversas arquitecturas de redes neuronales recurrentes o RNN por sus siglas en inglés, que intentan capturar las dependencias temporales de un conjunto de datos. A través de mecanismos de memoria de corto plazo, explotan la autocorrelación presente en este tipo de conjunto de datos.

De forma análoga, las redes neuronales convolucionales o CNN, se han desarrollado para explotar las regularidades espaciales que presentan los conjuntos de datos de imágenes. A partir del trabajo seminal de Yann LeCun en 1989 [5], se han desarrollado diversas estrategias y variaciones sobre este tipo de redes neuronales artificiales. Especialmente durante la última década, se han alcanzado niveles de desempeño similares al humano [6] en diversas tareas relacionadas con la Visión Artificial.

2.2.1. Entrenamiento de redes neuronales

Para los modelos de redes neuronales pueden establecerse dos momentos o tiempos claramente diferenciados: el entrenamiento y la inferencia.

Para que un modelo de redes neuronales sea útil a la tarea por la que fue diseñado, es necesario primero entrenarlo. Su utilidad estará en función de sus capacidades para obtener predicciones certeras, luego, en tiempo de inferencia. Entrenar un modelo de redes neuronales equivale a buscar los valores óptimos para todos sus parámetros y para ello pueden seguirse varias estrategias. La más habitual por amplísimo margen es la ya detallada de GDBP.

Durante el tiempo de entrenamiento de una red, el objetivo es el de encontrar un conjunto de parámetros para el modelo que minimicen una determinada función de error. Esta función de error dependerá de la tarea para la que se quiera entrenar el modelo. Para el caso específico de las redes convolucionales para clasificación de imágenes suele emplearse la función de entropía cruzada:

$$\text{Loss} = -\sum_{i=1} y_i \cdot \log \hat{y}_i$$

Fig 2.4: Entropía cruzada.

Donde y_i es la clase a la que pertenece la *i* imagen de entrada e $\hat{y}_i$ es la clase predicha por el modelo para la imagen *i*. Para calcular el valor de esta función de error, se realiza una inferencia o *Forward Pass* sobre todas las imágenes que componen el conjunto de datos de entrenamiento para obtener las predicciones de clase del modelo para cada una de las instancias. El valor de la función de error será tanto más bajo cuantas más coincidencias haya entre la clase predicha por el modelo y la clase verdadera de la imagen. Esto implica, por supuesto, contar con un conjunto de imágenes de entrenamiento etiquetadas. Es decir, la clase a la que pertenece cada imágen es conocida previamente y producto, en muchas ocasiones, del etiquetado manual.

Al comenzar el entrenamiento de una red neuronal, sus parámetros son inicializados al azar y por ende, las predicciones que el modelo pueda realizar tendrán el mismo carácter meramente aleatorio. En este estado, el valor de la función de error será muy elevado, ya que las predicciones que realice el modelo no coinicidirán, más que por chance pura, con las clases reales de las imágenes.

A partir del valor de la función de error, se calculan los gradientes respecto a este error para todos los parámetros de la red. Esto se realiza por medio del algoritmo de Backpropagation que consiste en propagar las derivadas en el sentido inverso al del gráfico computacional utilizado para realizar la inferencia o *Forward Pass*. Por medio de la regla de la cadena, se van obteniendo las derivadas parciales, comenzando por la última capa de la red y finalizando con la primera.

Los gradientes obtenidos por medio de Backpropagation se utilizan de forma inmediata para actualizar los parámetros del modelo en la dirección opuesta.

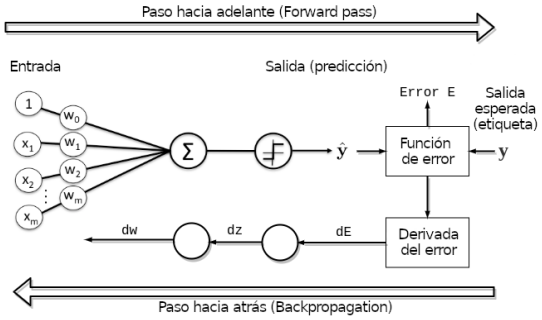


Fig 2.5: Pasos hacia adelante y hacia atrás para la actualización de parámetros en función del error obtenido.

Los gradientes calculados respecto a la función de error indican la dirección de crecimiento de la función de error para cada uno de los parámetros. El objetivo de la fase de entrenamiento de la red es el de minimizar dicha función de error, lo que equivale a obtener predicciones cada vez mas certeras. Para minimizar el valor de la función de error, entonces, los parámetros de la red se actualizan en la dirección opuesta a la de crecimiento. Es decir, se busca reducir el error que tendrá el modelo en inferencia al modificar sus parámetros.

Este proceso de inferencia, Backpropagation y actualización de parámetros se realiza de forma iterativa. Cada una de estas iteraciones es denominada época y todo el proceso en sí es denominado Descenso por Gradientes, ya que se busca "descender" por la topografía de la función de error de manera iterativa a través del cálculo de gradientes respecto a dicha función. Con cada iteración se realiza un pequeño paso, actualizando los parámetros del modelo en la dirección en la que permiten una reducción del error.

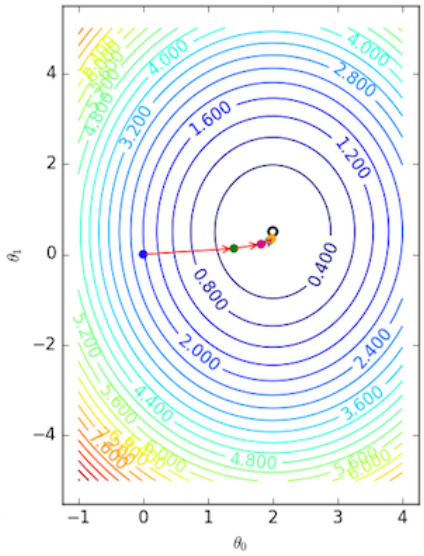


Fig 2.6: Topografía de la función de error y búsqueda de un mínimo local de forma iterativa. Cada flecha roja representa una iteración. En eje x parámetro 0 (θ_0), y en eje y, parámetro 1 (θ_1).

La topografía de la función de error suele contener innumerables mínimos locales. Cuando se alcanza alguno de estos mínimos el algoritmo de Descenso por Gradientes ha convergido y ya no será posible seguir reduciendo el valor del error, al menos sin escapar previamente de ese mínimo local.

Las redes neuronales vistas hasta ahora son las comunes o más simples que se pueden utilizar. A continuación, presentamos las redes convolucionales, que aumentan las capacidades de las redes neuronales básicas.

2.3. Redes convolucionales

Las redes neuronales convolucionales o CNN son un caso particular de redes neuronales. Están especialmente diseñadas para capturar regularidades espaciales del conjunto de datos de entrada. A diferencia de una red neuronal tradicional o densamente conectada, dónde cada unidad establece pesos diferenciados para cada salida de la capa precedente, las CNN aplican un mismo conjunto de parámetros a diferentes entradas de la red. De este modo, explotan las características propias de tipos de datos como las imágenes, donde existe alta correlación entre diversas áreas o segmentos de imagen.

Las CNNs se han desarrollado inicialmente para ser aplicadas en la tarea específica de clasificación de imágenes. Dada determinada imagen, identificar a qué clase pertenece dentro de un conjunto finito, e inicialmente muy limitado de clases. Obtuvieron su primer éxito en la tarea de clasificación de dígitos manuscritos. Es decir, a partir de la imagen de un determinado dígito, identificar dicho dígito.

A lo largo de la historia de las CNNs, se han ido incorporando nuevas tareas como la segmentación semántica, la localización de objetos y la detección de objetos, por mencionar solo algunas de ellas. Si bien la topología de la red está en función de la tarea para la cual fue designada, la estructura básica a partir de la que se han desarrollado las diversas aplicaciones es la misma y es la que aquí se describe con mayor detalle.

Los componentes centrales de toda red convolucional son cada una de las capas convolucionales donde, por medio de los filtros aprendidos durante el entrenamiento de la red, se realiza la operación de convolución. El hecho de utilizar un mismo conjunto de parámetros, es decir, un mismo filtro, para operar sobre los diferentes segmentos o parches de la imagen, persigue dos objetivos: por un lado, explotar las regularidades espaciales presentes en las imágenes, tal como se mencionaba anteriormente, a la vez que reducir la cantidad de parámetros que la red debe aprender durante el proceso de entrenamiento.

Cada unidad en una red neuronal totalmente conectada posee tantos parámetros como entradas, más un valor de sesgo adicional. Es decir, la cantidad de parámetros de una capa totalmente conectada es igual al producto de la cantidad de activaciones en la capa anterior $|a^{l-1}|$ más 1, con la cantidad de unidades en capa n_u^l:

$$n_w^l = (|a^{l-1}| + 1) \times n_u^l$$

Donde n_w^l es la cantidad de parámetros en capa *l,* $|a^{l-1}|$ es la cantidad de activaciones en capa *l-1* y n_u^l es la cantidad de unidades en capa *l.*

Si se considera, por ejemplo, el caso de una imagen de alta resolución, de 1080 x 1920 píxeles, una primera capa de una red densa o totalmente conectada, debería tener (1080 x 1920 + 1) parámetros x la cantidad de unidades de dicha capa.

Si se tratase de 64 unidades, por ejemplo, para esa primera capa, el modelo ya estaría destinando más de 132 millones de parámetros. Al utilizar capas convolucionales en lugar de capas densamente conectadas, para cada segmento de las imágenes de entrada se utilizan los mismos parámetros. Siguiendo el caso del ejemplo propuesto, con 64 unidades en una primera capa convolucional, solo se estaría requiriendo (3x3x3 +1) x 64 parámetros. Es decir, solamente 1792. Esto sería así para el caso particular, pero muy habitual, de filtros de 3x3, con 3 canales de entrada (RGB) y zancada de 1. En capas intermedias de la red la cantidad de canales de entrada suele ser mucho más elevada. Por ejemplo, con 128 canales de entrada y 256 unidades, se requieren (3x3x128 + 1) x 256 parámetros, es decir, 295.168.

Las redes convolucionales actuales están típicamente formadas por varias decenas de capas convolucionales, para culminar en una o dos capas totalmente conectadas. A la salida de cada capa convolucional se aplica una función de activación no lineal. Probablemente la más utilizada es la función ReLU o unidad lineal rectificada.

Suele emplearse también en muchos modelos de CNNs capas intercaladas de *Max Pooling*. Estas capas realizan una operación muy sencilla consistente en la extracción del valor máximo dentro de determinada ventana espacial. Operan de modo similar a las convoluciones, utilizando un filtro que se desplaza a través de todo el mapa de características. Es común encontrar tamaños de ventana de 2x2 píxeles con zancada de 2. En este caso, la operación consiste en ofrecer como salida el valor máximo presente en cada segmento de entrada de 2x2 píxeles. El objetivo de esta operación es el de reducir las dimensiones de la entrada, sin perder información significativa. Luego de una, dos o más capas convolucionales puede intercalarse una capa de *Max Pooling*.

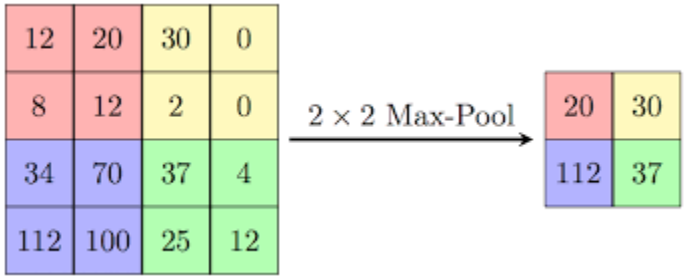


Fig 2.7: Operación de Max Pooling con un filtro de 2x2 y zancada de 2.

Habitualmente las redes convolucionales están diseñadas bajo una clara política de reducción progresiva de las dimensiones de la imagen de entrada. A medida que determinada imagen de

entrada es procesada capa a capa, se va reduciendo su alto y ancho y por el contrario, se incrementa la cantidad de canales.

Inicialmente, si se trata de un conjunto de imágenes en color, la primera capa convolucional de la red opera sobre 3 canales (RGB). La cantidad de canales de salida de esta primera capa será equivalente a la cantidad de unidades o filtros que esta capa posea. Es decir, la cantidad de canales de salida de una capa convolucional está determinada por la cantidad de unidades presentes en esa capa. A medida que se progresa en profundidad, las políticas de diseño de CNNs suelen establecer un incremento progresivamente la cantidad de unidades/canales.

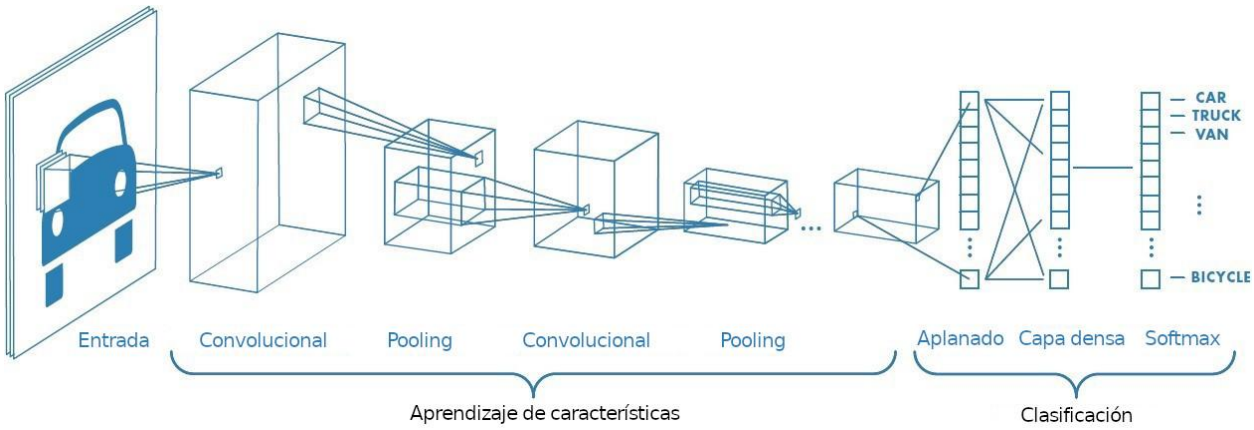


Fig 2.8: Red neuronal convolucional aplicada a la tarea de clasificación de imágenes.

Existen diversas variantes a esta tipología estándar o "vainilla", como por ejemplo las redes totalmente convolucionales, donde se prescinde de las capas de *Max Pooling* y totalmente conectadas, realizando las reducciones en alto y ancho a través directamente de la zancada de las convoluciones. Asimismo, se han desarrollado diversas variantes en cuanto a la implementación de la operación de convolución, como por ejemplo, las convoluciones espacialmente separables, que buscan reducir el costo computacional y la cantidad de parámetros de la red. Otra variante destacable la constituyen la implementación de conexiones residuales, que ofrecen un camino alternativo de flujo de información, facilitando la propagación del gradiente hacia las primeras capas. Este es un punto de consideración en el entrenamiento de redes neuronales, tal como se verá en las siguientes secciones.

2.3.1. Operación de convolución

La operación distintiva y más importante de toda red convolucional es la operación de convolución. Esta operación, aplicada sucesivamente a través de las diversas capas que

conforman una red convolucional, permite la transformación progresiva de la información de entrada, obteniendo un vector de salida, generalmente de menor dimensionalidad.

En una operación de convolución intervienen dos elementos de entrada y se obtiene como resultado un tercero. Los elementos de entrada o factores de la operación son, por un lado, una observación, ejemplo o instancia de determinado conjunto de datos y un filtro o *kernel*. Tanto la observación como el filtro están conformados por una arreglo de escalares de dimensionalidad fija e igual para ambos elementos. Pueden realizarse convoluciones de cualquier dimensionalidad *d*, siendo *d* cualquier valor entero positivo. En el caso particular de interés para el presente trabajo, la dimensionalidad de las convoluciones es 2, y será este caso particular el que se desarrollará a continuación. En el dominio del procesamiento de imágenes y Visión Artificial, estas 2 dimensiones corresponden al ancho y alto de las imágenes.

Más allá de particularidades en la implementación que más adelante se describirán, la operación de convolución puede definirse concretamente como el producto punto entre el filtro y segmentos o parches sucesivos de la imagen-instancia de entrada. Estos parches deben ser del mismo ancho y alto que el filtro para poder efectuar el producto punto. El producto punto en sí, se define como la sumatoria del producto elemento a elemento entre ambas matrices, la del filtro y la del segmento de imagen [7].

Como resultado de la aplicación del producto punto entre filtro y parche, se obtiene un escalar de salida. El conjunto de estos escalares producidos por el paso del filtro sobre los diferentes segmentos o parches de la imagen de entrada constituyen la salida de la operación de convolución. Cada uno de estos escalares de salida posee un par de coordenadas que permiten identificar qué segmento o parche de la imagen de entrada lo ha producido. En el dominio del procesamiento de imágenes y Visión Artificial, esta salida conjunta de todos estos escalares es denominada mapa de características y posee la misma dimensionalidad (2) que el filtro y la imagen de entrada. En este mapa de características, cada escalar de salida es ubicado según estas coordenadas espaciales mencionadas. Cada valor obtenido será ubicado en el mapa de características de salida en la misma posición relativa a la del filtro sobre la imagen de entrada.

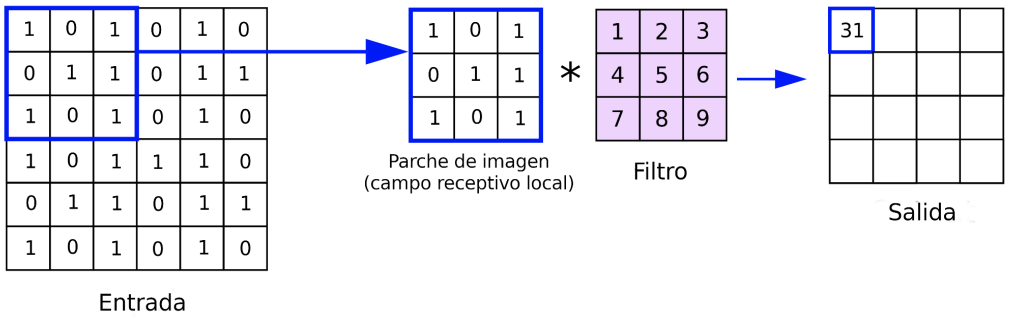


Fig 2.9: Operación de convolución entre imagen de entrada y filtro y la ubicación del escalar resultante del producto punto sobre el mapa de características de salida.

Tal como se observa en la figura 2.9, luego de realizar los productos punto por todos los posibles segmentos de imagen, el mapa de características resultante tendrá un tamaño menor en ancho y alto respecto a la imagen de entrada. El mapa de características tendrá tantos píxeles de ancho y de alto como posibles ubicaciones del filtro sobre la imagen de entrada existan. En el caso de un filtro de 3x3 píxeles y una imagen de entrada de 6x6, como es el caso ilustrado en la figura 2.9, la matriz de salida será de 4x4. En el caso general, el ancho y alto del mapa de características será:

$$a_w = i_w - f_w + 1$$
$$a_h = i_h - f_h + 1$$

Ecuación 2.1.

Donde a_w y a_h son el ancho y alto respectivamente del mapa de características, i_w y i_h son el ancho y alto de la imagen y f_w y f_h son el ancho y alto del filtro.

Más allá de esta definición básica de la operación de convolución en 2 dimensiones, existen varias particularidades en las posibles implementaciones en el contexto de una red convolucional. Las más importantes a considerar son, probablemente, las siguientes:

- **Tamaño del filtro.** En la amplia mayoría de aplicaciones en redes convolucionales, el ancho y alto de los filtros es idéntico, es decir, se trata de filtros cuadrados. Suelen utilizarse habitualmente filtros de ancho y alto impar, siendo los más habituales en las arquitecturas actuales los filtros de 3x3, 5x5 y 7x7. Son también comunes los filtros de 1x1 para la aplicación de convoluciones de ciertas variantes específicas.

- **Relleno (*padding*).** Con el objetivo de mitigar o eliminar la reducción en las dimensiones que se producen al realizar la operación de convolución, se puede agregar un borde de relleno al mapa de características. Se suele optar por el cero como valor de relleno. El ancho de este borde perimetral en todo el contorno de la imagen de entrada estará en función de las dimensiones del filtro. En caso de que se desee obtener un mapa de características de iguales dimensiones a las de la imagen de entrada, se colocarán tantos píxeles de relleno como sean necesarios para que esto suceda. De acuerdo a la ecuación 2.1, para un filtro de 3x3, se utilizará un relleno de 1 y para uno de 5x5 un relleno de 2. Generalizando, la cantidad de píxeles de relleno necesarios será:

$$p_w = (f_w - 1) / 2$$
$$p_h = (f_h - 1) / 2$$

Ecuación 2.2

Donde p_w y p_h son la cantidad de píxeles de relleno para el ancho y alto respectivamente.

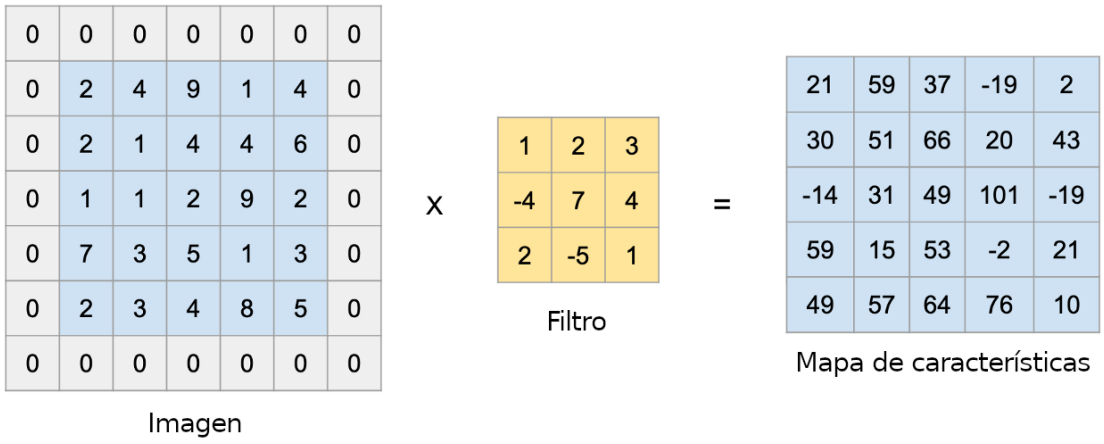


Fig 2.10: Relleno con ceros de 1 píxel de ancho perimetral en la imagen de entrada de 5x5 para generar un mapa de características que conserve las mismas dimensiones.

- **Zancada (stride)**. Una de las estrategias habituales para realizar una reducción en las dimensiones del mapa de características es utilizar zancadas mayores a 1, al desplazar el filtro sobre la imagen de entrada. En lugar de efectuar un producto punto entre el filtro y cada uno de los posibles segmentos de la imagen de entrada (zancada = 1), pueden saltarse tantos segmentos por vez como se desee. La motivación detrás de esta estrategia es la de reducir el costo computacional y/o reducir progresivamente, capa a capa, las dimensiones de la imagen [7].

2.3.2. Las redes convolucionales y la extracción de información visual

Para lograr un buen desempeño en la tarea de clasificación de imágenes sobre la que fueron entrenados, los modelos requieren condensar la información de entrada de forma significativa, en un espacio dimensional reducido. Esta condensación es de alrededor de tres órdenes de magnitud, si se considera que las imágenes de entrada están en el orden del millón de píxeles y el vector de salida de la última capa convolucional puede ser unas mil veces menor.

Para condensar la información de entrada de un modo significativo a la tarea de clasificación, estos modelos deben encontrar parámetros para los filtros convolucionales que permitan lograr abstracciones de modo incremental sobre la información de entrada.

Por la propia arquitectura de las redes convolucionales, cada capa subsiguiente posee un campo receptivo mayor a la capa anterior. Es decir, a mayor profundidad dentro de la red, cada operación de convolución que realice determinado filtro se verá afectada por un área mayor de la imagen original de entrada. Este hecho fundamental de las redes convolucionales opera como condicionante en cuanto al tipo de información que cada capa es capaz de sumarizar.

Mientras que las primeras capas de la red, con un campo receptivo menor, son capaces de detectar características de bajo nivel en la imagen, como bordes y contornos, las capas medias y más profundas logran detectar características de alto nivel y mayor grado de abstracción.
Bien podría pensarse que cada capa convolucional realiza una operación de sumarización progresiva de la información visual, partiendo de la sumarización realizada por la capa anterior y obteniendo una condensación progresiva de dicha información de entrada.

De acuerdo a esta mecánica progresiva en la sumarización de la información, al finalizar el entrenamiento, los filtros de las primeras capas terminan obteniendo parámetros que los convierten en buenos detectores de características básicas en la imágen, como bordes y contornos. Por su parte, las capas subsiguientes terminan configurando parámetros para sus filtros que los hacen buenos detectores de características de mayor y progresivo nivel de abstracción, a partir de la combinación de los mapas de características obtenidos por los filtros de las capas anteriores.

Como resultante de esta característica de las redes convolucionales, donde surge naturalmente una progresiva especialización de sus capas, se obtienen filtros más específicos al conjunto de datos, cuanto más profunda sea la capa. Este fenómeno queda reflejado en el éxito de técnicas como la transferencia de aprendizaje (*Transfer Learning*), donde una red convolucional entrenada sobre determinado conjunto de datos puede ser utilizada para realizar clasificaciones sobre un segundo conjunto de imágenes de otro dominio y con otras clases.

Para las primeras capas, los filtros aprendidos por el modelo suelen ser directamente transferibles a un segundo modelo, mientras que la práctica habitual es la de reentrenar las capas más profundas sobre el nuevo conjunto de datos. La experiencia ha demostrado buenos resultados por medio de este procedimiento, lo cual refuerza la hipótesis acerca de la generalidad de aplicación de los filtros de las primeras capas.

El aprendizaje de los pesos y filtros de una red depende de los valores de los mismos al comienzo del entrenamiento; es decir, de la inicialización. Para ellos, existen varias estrategias que detallamos a continuación.

2.4. Estrategias de inicialización de redes neuronales

No solo el método de entrenamiento tiene impacto sobre los parámetros obtenidos para el modelo y por ende su desempeño final en tiempo de inferencia. El modo en que se inicializan estos parámetros puede también tener un rol muy importante en el modelo resultante, una vez finalizado el entrenamiento. Existen varias causas que permiten comprender el impacto del método de inicialización de los parámetros de la red en el resultado final obtenido [49].

Primeramente, cabe destacar dentro de las propiedades de la función de error, su falta de convexidad. Partiendo de un punto cualquiera de la superficie de dicha función y por medio del método de Descenso por Gradientes, se arribará con certeza a uno de los tantos mínimos locales. Cada uno de estos mínimos locales presentará diferentes valores para la función de error y siendo el objetivo del entrenamiento minimizar dicho error, la elección del punto de partida desde el cual efectuar el descenso no resulta trivial.

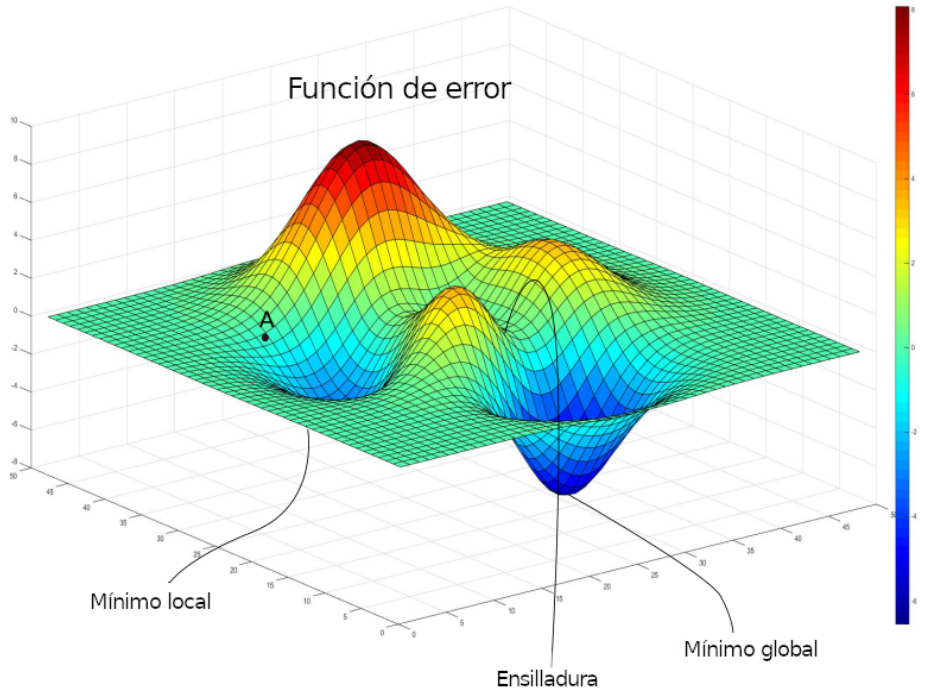


Fig 2.11: Topográfia de la función de error. De acuerdo al punto de inicialización para los parámetros de la red y del algoritmo de optimización utilizado se arribará a diferentes mínimos locales.

En la figura 2.11 se observa la topografía de la función de error en función de solamente 2 parámetros, por motivos ilustrativos. Una red neuronal artificial puede contener millones y hasta miles de millones de parámetros de los cuales será dependiente la función de error. La inicialización de dichos parámetros en diferentes puntos de la topografía hará que el proceso de optimización converja en diferentes mínimos locales de dicha función.

La búsqueda de un método óptimo de inicialización de los parámetros de la red redunda entonces en la posibilidad tanto de obtener redes mejor entrenadas como de reducir los tiempos de entrenamiento, permitiendo que la red converja hacia un buen mínimo local con menor número de iteraciones.

Adicionalmente, existen tres potenciales problemas que deben prevenirse al momento de inicializar los parámetros y que guían las diferentes estrategias de inicialización:

- Inicialización de los parámetros de diferentes unidades con valores idénticos: en caso de inicializar dos o más unidades con los mismos valores, los gradientes serán idénticos y por ende también las actualizaciones que se realicen en cada iteración. Esto provoca que terminen obteniéndose unidades repetidas, con idénticos valores, luego del entrenamiento, reduciendo la potencial capacidad de la red.
- Inicialización con valores demasiado pequeños: esto puede provocar potencialmente el fenómeno conocido como gradientes desvanecientes dificultando el correcto entrenamiento de la red.
- Inicialización con valores demasiado grandes: contrariamente al caso anterior, pero con las mismas potenciales consecuencias, valores demasiado grandes de inicialización pueden generar el fenómeno conocido como gradientes explosivos [8].

Como parte del presente trabajo, se exploraron métodos alternativos para la inicialización de los parámetros de las redes convolucionales, buscando tanto una mejora en el desempeño final de la red, como una reducción en el número de iteraciones para lograr la convergencia. En el capítulo 4 se desarrollan estos métodos y en el capítulo 5 se presentan los resultados obtenidos.

A continuación se detallan algunos de los métodos de inicialización más ampliamente utilizados en la literatura.

2.4.1. Inicialización Glorot

En base a los trabajos de Xavier Glorot et al [9], se desarrollan dos métodos de inicialización aleatorios para los parámetros de redes neuronales de amplia utilización: Glorot Uniforme y Glorot Normal. Ambos toman valores aleatorios para inicializar los pesos de las neuronas. Mientras que Glorot Uniforme toma valores a partir de una distribución uniforme, Glorot Normal los toma de una distribución normal o gaussiana. En ambos casos, las distribuciones se centran en cero y los sesgos de las unidades se fijan en cero.

Para el caso de Glorot Uniforme, se toma un rango de posibles valores que se determina de acuerdo a la ecuación 2.3:

$$\sqrt{\frac{6}{fan{-}in + fan{-}out}}$$

Ecuación 2.3: Fórmula para el cálculo de los límites de muestreo para Glorot Uniforme.

Dónde *fan-in* es la cantidad de entradas de la capa y *fan-out* la cantidad de unidades de la capa y por ende, la cantidad de salidas. Es decir, en función de la cantidad de unidades de la capa actual y de la anterior, se determina el rango de muestreo de una distribución uniforme desde la cual se obtendrán los pesos de las unidades de forma aleatoria.

Este es el método de inicialización que se ha utilizado a través de todos los experimentos realizados en este trabajo para determinar la línea base de comparación con el resto de los métodos de inicialización y entrenamiento planteados.

El caso de la inicialización por medio de Glorot Normal, se trata igualmente de tomar valores aleatorios para los pesos de la capa, en este caso, a partir de una distribución normal centrada en cero y con desvío estándar calculado de acuerdo a la ecuación 2.4:

$$W \approx N(\mu, \sigma)$$

$$\mu = 0 \qquad \sigma = \sqrt{\frac{2}{fan{-}in + fan{-}ou}}$$

Ecuación 2.4: Fórmula de inicialización por medio de Glorot Normal.

Dónde, nuevamente *fan-in* y *fan-out* representan la cantidad de unidades de entrada y salida. Para el caso particular de las redes convolucionales, donde se aplican filtros que se desplazan sobre los mapas de características, la cantidad de unidades de entrada y de salida se pueden calcular mediante la ecuación 2.5:

$$fan\text{-}in = f_w \times f_h \times c_{in}$$
$$fan\text{-}out = f_w \times f_h \times c_{out}$$

Ecuación 2.5: Cálculo de fan-in y fan-out para el caso particular de capas convolucionales.

Donde f_w y f_h son el ancho y alto del filtro, c_{in} la cantidad de canales de entrada y c_{out} la cantidad de canales de salida.

2.4.2. Inicialización He

Este método de inicialización de parámetros desarrollado por He at al [10] se centra en la búsqueda de un método óptimo para el caso específico de redes convolucionales con funciones de activación ReLU.

Desarrolla también dos estrategias, una basada en distribución normal (He Normal) y una en distribución uniforme (He Uniforme), ambas centradas también en cero. Para el caso de la uniforme, los límites inferior y superior se calculan de acuerdo a la ecuación 2.6:

$$\sqrt{\frac{6}{fan-in}}$$

Ecuación 2.6: Fórmula para el cálculo de los límites de muestreo para He Uniforme.

Dónde, *fan-in* nuevamente representan la cantidad de unidades de entrada. Para el caso de la inicialización He Normal, la fórmula es la desarrollada en la ecuación 2.7:

$$W \approx N(\mu, \sigma)$$

$$\mu = 0 \qquad \sigma = \sqrt{\frac{2}{fan-in}}$$

Ecuación 2.7: Fórmula de inicialización por medio de He Normal.

Como puede observarse, a diferencia de Glorot, la inicialización He solo contempla la cantidad de unidades de entrada, sin considerar la de unidades de salida. Todos los sesgos son inicializados en cero, de forma idéntica a la inicialización Glorot.

2.4.3. Inicialización ortogonal

En el trabajo de Saxe et al. [11] se desarrolla una propuesta para la inicialización de los parámetros de redes neuronales basada en la generación de matrices ortogonales. En particular, los autores proponen seleccionar como matriz inicial de pesos una matriz cuadrada ortogonal al azar tal que $W^T W = I$ para incrementar la velocidad de convergencia de la red.

Utilizando una matriz ortogonal, se obtienen varias propiedades deseables en relación a la convergencia de la red durante el entrenamiento. Por un lado, preservan la norma del vector de entradas al realizar el producto, es decir: $||Wx||2 = ||x||2$, desfavoreciendo la aparición de los fenómenos de gradiente explosivo y gradiente desvaneciente. Adicionalmente, al estar

compuesta filas ortogonales, se fomenta el desarrollo, durante el entrenamiento, de unidades especializadas en diferentes características del conjunto de entrada.

Para el caso particular de las redes convolucionales, se busca que los pesos para cada canal de determinada capa sean ortogonales, fomentando la especialización diferenciada entre canales, de modo análogo al caso más general planteado anteriormente.

Respecto a la implementación de este método, puede utilizarse la Descomposición de Valores Singulares (SVD por sus siglas en inglés) de una matriz generada al azar con distribución normal para obtener la matriz de pesos. [12]

Los métodos de inicialización aquí descriptos son los empleados con mayor frecuencia en el entrenamiento de redes neuronales artificiales por medio de GDBP, siendo este último el algoritmo más ampliamente utilizado. Sin embargo, se han propuesto varias alternativas para el entrenamiento de redes convolucionales que se discuten a continuación.

2.5. Métodos supervisados de entrenamiento para CNNs alternativos (a GDBP)

Se han realizado varios intentos y se han propuesto varios algoritmos para el entrenamiento de redes neuronales con métodos alternativos a GDBP. Por lo general, el objetivo primordial ha sido reducir las necesidades de recursos computacionales. La mayoría de ellos datan ya de algunas décadas, donde el estado del arte, en cuanto a capacidad de cómputo, era significativamente menor a la actual.

En relación a trabajos más actuales, en Ma et al.(2019) [13] se presenta el algoritmo “HSIC bottleneck” (criterio de independencia de Hilbert-Schmidt) con ventajas sobre GDBP como el hecho de facilitar el procesamiento paralelo, requerir menor cantidad de operaciones y no presentar los problemas de gradientes desvanecientes o explosivos.

Este algoritmo no utiliza Backpropagation y ni entropía cruzada como función de error. Por el contrario, se basa en la desigualdad de Fano y el criterio de información mutua. En lugar de condicionar la clase de determinada instancia a las características de dicha instancia de modo indirecto, a través de Backpropagation, utiliza la entropía condicionada directamente sobre las características de las instancias. Es decir, H(Y|X). Cuando las probabilidades de clasificar incorrectamente determinada instancia son bajas, su entropía condicional también lo es,

resultando en un valor de información mutua alta, ya que la entropía de clase H(Y) permanece siempre constante para determinado conjunto de datos.

De cualquier forma, los resultados obtenidos por "HSIC bottleneck" en tareas de clasificación de imágenes no llegan a alcanzar los obtenidos por métodos tradicionales que utilizan el algoritmo de Backpropagation. En definitiva, los avances producidos en la última década respecto al estado del arte en diversas tareas del área de Visión Artificial (Computer Vision) son todos producto del empleo de GDBP como método de entrenamiento sobre diferentes arquitecturas convolucionales.

Así como se han desarrollado métodos alternativos a GDBP en el marco específico de las redes convolucionales, también se han propuesto varios enfoques para lograr su entrenamiento de modo no supervisado, pero empleando GDBP.

2.6. Estrategias de entrenamiento no supervisado de CNNs con empleo de GDBP

Las diferentes metodologías de obtención de filtros para capas convolucionales ensayadas en la presente tesis, exploran en definitiva las posibilidades de realizar el entrenamiento de una red convolucional, de forma no supervisada. Es por ello que resulta relevante para el presente trabajo, presentar otros trabajos previos que comparten este mismo objetivo.

En cuanto a métodos no supervisados de entrenamiento, existen varios enfoques y una importante cantidad de trabajos que han surgido a partir de las arquitecturas de Auto Codificadores (*Autoencoders* en inglés) [14].

Los Auto Codificadores son modelos compuestos por dos módulos, un codificador y un decodificador encadenados. El codificador es alimentado con instancias del conjunto de datos de alta dimensionalidad y genera vectores de salida de menor dimensionalidad. Estos vectores son luego inyectados en el decodificador, cuya tarea es reconstruir la instancia con la dimensionalidad original. Para ello, suele utilizarse como función de error, alguna métrica de distancia entre la instancia original inyectada a la entrada y la obtenida a la salida.

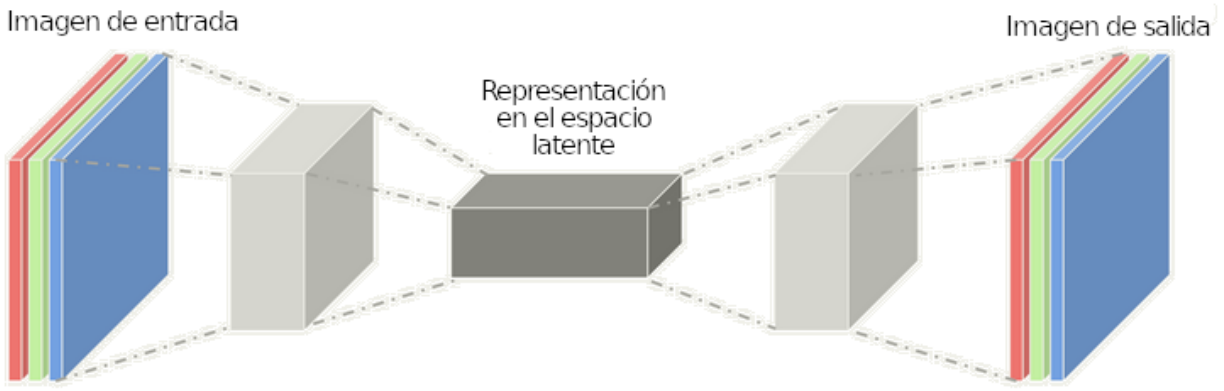


Fig 2.12: Arquitectura de un Auto Codificador convolucional.

Las arquitecturas de los codificadores y decodificadores suelen construirse de forma complementaria. Para el caso específico de Auto Codificadores convolucionales, suelen aplicarse operaciones de convolución y convolución transpuesta en codificador y decodificador respectivamente.

En el contexto del entrenamiento de los Auto Codificadores, no existe una etiqueta de clase como tal. Todo el entrenamiento es realizado directamente a partir de las instancias del conjunto de datos sin etiquetar. Se trata por ende, de un tipo de aprendizaje no supervisado.

El objetivo del entrenamiento es, por lo general, obtener un modelo de codificador capaz de condensar la información semántica de un conjunto de imágenes en un vector de menor dimensionalidad. Es decir, un extractor de características.

Un caso particular de Auto Codificadores son los modelos MAE [14] (por sus siglas en inglés: *Masked Autoencoders*). En este tipo de modelos, se realizan oclusiones parciales de la imágen de entrada y se entrenan ambos módulos (codificador y decodificador) para reconstruirla, de modo muy similar a las técnicas de entrenamiento para Auto Codificadores de tipo de supresión de ruido o *denoising*.

En todos los enfoques no supervisados basados en arquitecturas de Auto Codificadores se utiliza GDBP para actualizar los pesos de ambos modelos en conjunto y el objetivo es obtener un codificador que pueda lograr una buena representación y abstracción del contenido de las imágenes a través de su vector de características de salida.

Alternativamente a los enfoques basados en Auto Codificadores, existen varios trabajos que logran entrenamientos no supervisados o auto supervisados, empleando métodos de aprendizaje por contraste como [15, 16, 17]. En Dosovitskiy et al [17] se entrena un modelo para discriminar entre lo que los autores denominan, clases subrogantes. Cada clase subrogante es el

conjunto de imágenes producidas por medio de aumentaciones aleatorias a un único parche de imágen original. De este modo, al generar clases de forma auto supervisada, se evita la necesidad de contar con muestras etiquetadas que agrupen las imágenes por clases predefinidas de forma manual. Adicionalmente, las representaciones aprendidas por el modelo mediante esta tarea, resultan invariantes a las transformaciones que se hayan empleado en el proceso de aumentación.

Otro trabajo de interés utilizando un enfoque no supervisado por medio del aprendizaje por contraste (*contrastive learning*) es SimCLR [15]. El método de entrenamiento se basa en la aumentación de datos (*data augmentation*) y una función de error de contraste. Se generan nuevas imágenes con modificaciones aleatorias a partir de cada ejemplo original y se las codifica a través de una red convolucional (ResNet). Luego una pequeña cabeza de proyección, proyecta las salida de la red hacia un espacio vectorial para realizar las comparaciones. A partir de una función de error de contraste, el algoritmo intenta maximizar la similitud entre las aumentaciones (*data augmentation*) realizadas sobre una misma imagen original. Finalmente realizan un ajuste fino de la red de modo supervisado (fine tuning) utilizando solo el 1% de las etiquetas disponibles en el conjunto de datos (ImageNet [18]). A través de este método de entrenamiento, los autores han logrado obtener resultados de desempeño superiores a los de AlexNet [19] utilizando 100 veces menos datos etiquetados.

Otro trabajo basado en entrenamiento no supervisado o auto supervisado con GDBP, también utilizando el principio de aprendizaje por contraste, es SwAV [16]. Basados en este principio de entrenamiento, los autores construyen un algoritmo más eficiente en términos computacionales y de uso de memoria, al evitar las comparaciones directas de a pares de imágenes aumentadas. Para ello, en lugar de buscar minimizar las diferencias entre 2 aumentaciones de una misma imagen, realizan agrupamientos en línea no supervisados de dichas aumentaciones y emplean una función de error para maximizar la consistencia de estos agrupamientos. De este modo, evitan la comparación directa entre pares de aumentaciones. El método emplea específicamente como tarea de excusa un algoritmo que intenta predecir el código (la asignación de grupo) de una aumentación, a partir de la representación de otra aumentación o "vista" de la misma instancia de entrenamiento.

Adicionalmente, específicamente para el campo del diagnóstico médico por imágenes, donde los conjuntos de datos etiquetados suelen ser particularmente reducidos y por ende especialmente necesaria la aplicación de métodos no supervisados, es destacable el trabajo de Ahn et al [20]. En este trabajo se utiliza un ensamble de dos redes convolucionales (AlexNet y VGG19), seguidas por el algoritmo de agrupamiento k medias. Las redes generan un vector de representación de 4096 dimensiones con el que se alimenta la entrada de k medias. El algoritmo se entrena de extremo a extremo, es decir, ajustando los parámetros de las redes a partir de la

función de error calculada al final de todo el procesamiento. Por medio del algoritmo de agrupamiento se generan clases subrogantes (los grupos) para cada representación vectorial de las imágenes.

El empleo de k-medias como técnica de agrupamiento no supervisado es también empleado en esta tésis para el agrupamiento de parches de imágenes con el objetivo de facilitar la separación de clases. A diferencia del trabajo de Ahn et al., el agrupamiento se realiza sobre el conjunto de datos de entrada, pero con el objetivo similar final de obtener una separación de clases por medios no supervisados.

2.7. Métodos alternativos para el entrenamiento de filtros para capas convolucionales

Adicionalmente, podemos encontrar todo un conjunto de trabajos que, si bien utilizan GDBP como estrategia de optimización general, introducen también el empleo de filtros calculados de forma análitica para reducir tanto las necesidades de cómputo como el espacio de soluciones a explorar.

Estos trabajos parten del amplio conocimiento previo que se tiene en la comunidad de Visión Artificial sobre diferentes tipos de filtros y su aplicación. Específicamente los filtros Gabor se han empleado en varios trabajos como [21, 22], utilizándolo para la primera o primeras capas convolucionales. Los filtros Gabor están basados en una onda plana sinusoidal de determinada frecuencia y orientación, siendo éstos los parámetros que permiten definir un filtro Gabor en particular.

Existen varios justificativos para el empleo de filtros Gabor específicamente en la primera capa de una red convolucional. Por un lado, existe evidencia de la existencia de células simples en la primera sección del córtex visual (V1) en mamíferos que responden fundamentalmente a la orientación y frecuencia espacial [23] y pueden ser muy bien modeladas por los filtros Gabor [22]. Adicionalmente, parte de la comunidad de Visión Artificial reconoce que los modelos de redes convolucionales entrenados de modo tradicional, sobre un corpus de imágenes naturales, terminan aprendiendo una serie de filtros Gabor para su primera capa [24].

Los modelos entonces son entrenados con GDBP pero con la salvedad de que no se busca la optimización directa de los parámetros de los filtros dentro del espacio completo de soluciones sino que se los restringe al espacio de funciones Gabor [21]. Es decir, se introduce un sesgo

inductivo al modelo, permitiéndole adoptar para los parámetros de los filtros de la primera o primera capas solamente funciones Gabor. Para ello y por medio del descenso por gradientes se busca optimizar, no ya libremente los parámetros de los filtros, sino los parámetros de la función Gabor para cada uno de ellos. Esto por supuesto redunda en una reducción en la cantidad de parámetros que el modelo debe aprender y por ende repercute positivamente tanto en las necesidad de recursos computacionales, tiempo requerido para la convergencia y el tamaño del conjunto de datos necesario para evitar el sobreajuste (overfitting), así como en la interpretabilidad de los mismos.

2.7.1. Estrategias de entrenamiento e inicialización basadas en CP

Una de las estrategias para el entrenamiento de filtros de capas convolucionales explorada en el presente trabajo es el de la descomposición del conjunto de datos por medio de Componentes Principales. Resulta entonces central para el desarrollo de los siguientes capítulos poder conocer los trabajos previos que se han realizado utilizando esta misma técnica.

La descomposición por medio de Componentes Principales para el entrenamiento de filtros convolucionales ha suscitado el interés de varios investigadores [25, 26, 27, 28, 29, 30, 31]. Los componentes principales de determinado conjunto de datos son un conjunto de vectores ortogonales entre sí con algunas particularidades. Es un conjunto ordenado de vectores, tantos como dimensiones posee el conjunto de datos. El orden de estos vectores es el de importancia decreciente en cuanto a varianza de los datos explicada. El primer vector del conjunto será el que mayor variabilidad de los datos posea. Es decir, define la dirección donde el conjunto de datos posee mayor variabilidad. El segundo de estos vectores, ortogonal al primero y a todo el resto, constituye la segunda dirección de mayor varianza explicada. Cada vector subsiguiente configura una nueva dirección, con cada vez menor varianza explicada del conjunto de datos.

Estos vectores son denominados autovectores o *eigenvectors* y son obtenidos a partir del proceso de determinación de los componentes principales de determinado conjunto de datos. Las dos particularidades que los hace buenos candidatos para el entrenamiento de filtros convolucionales en el contexto de CNNs son su mutua ortogonalidad y el hecho de que definan las direcciones de mayor variabilidad de los datos. Esto último permite capturar con eficiencia el espacio de las entradas a la capa, mientras que la ortogonalidad asegura la nula correlación entre los filtros.

El primer aporte realizado en cuanto a entrenamiento de filtros convolucionales utilizando la descomposición en componentes principales, es el realizado por Chan et al [25] del año 2014. En este trabajo, los autores plantean 3 estrategias para el entrenamiento de filtros para 2 capas

convolucionales en cascada. Por una parte, la construcción de filtros por medio del algoritmo PCA, tomando los parches de entrada, tanto de la primera como de la segunda capa, como conjunto de entrenamiento del algoritmo. A través de la descomposición matricial, generan los filtros a partir de los autovectores encontrados. A esta red la denominan PCANet. Adicionalmente y de modo de poder comparar los resultados obtenidos, realizan el entrenamiento a partir de LDA y de forma aleatoria. Agregan luego etapas adicionales de binarización y hassing que no son de relevancia para la presente tesis.

Otro aporte es el de Ren et al [26] donde se utiliza PCA para la extracción de los autovectores del conjunto de imágenes de entrada como método de inicialización de parámetros para los filtros convolucionales. Este trabajo data del año 2016, y su principal interés para la aplicación de PCA es la de facilitar la convergencia posterior del entrenamiento por Backpropagation y Descenso por Gradientes y evitar la degradación de los gradientes que se produce en su paso capa a capa, desde la función de error hasta la primera capa convolucional. La técnica de cálculo de autovectores se utiliza exclusivamente con el propósito de inicializar los parámetros de los filtros, que luego serán ajustados de forma tradicional durante el entrenamiento. La motivación del empleo de PCA radica en la identificación de las direcciones de mayor variabilidad de los datos de entrada, con la intuición por parte de los autores, de que la transformación lineal del conjunto de entrenamiento hacia estas direcciones de máxima variabilidad permitirían luego una convergencia más rápida al reducir los problemas de entrenamiento asociados con los gradientes desvanecientes. Cabe destacar que la implementación de está técnica está limitada a la inicialización de los pesos de la primera capa convolucional de una red con arquitectura LeNet-5.

Otro trabajo relevante en esta dirección, utilizando el algoritmo de PCA como técnica no supervisada en conjunto con el entrenamiento por Backpropagation y Descenso por Gradientes para redes convolucionales es el de Soon et al [27]. Si bien este trabajo está enfocado puntualmente en el dominio de la clasificación de tipos de vehículos para uso en sistemas de vigilancia, su aporte resulta sumamente relevante en el contexto de la presente tesis.

Con la motivación de obtener una mejora considerable en los tiempos de entrenamiento y particularmente eliminar la necesidad del empleo de hardware GPU, este trabajo del 2020 utiliza el algoritmo de PCA para el entrenamiento de filtros convolucionales. Es particularmente destacable la aplicación de esta metodología a lo largo de todas las capas convolucionales de la red, y no ya como mera inicialización de parámetros sino como paso único de entrenamiento.

En este enfoque, los autores utilizan Backpropagation y Descenso por Gradientes exclusivamente para el entrenamiento final de la capa densamente conectada.

El método de entrenamiento de filtros, capa a capa, está basado en el trabajo de Chan et al de 2015 [25] y su principal aporte radica en la utilización de funciones de activación ReLU y una capa intermedia de MaxPooling de acuerdo a estándares más actuales para el entrenamiento de redes convolucionales.

Por otra parte, en el aporte realizado por Zhang et al [28] los autores combinan PCA con LDA en una metodología que denominan PLDANet. Si bien Chan et al [25] ya habían utilizado estas técnicas en su trabajo del año 2015 para el entrenamiento de filtros para capas convolucionales, los autores de PLDANet combinan ambas técnicas en una única metodología de entrenamiento. Su motivación, muy similar a la de otros autores que han tomado enfoques similares, radica tanto en la minimización de costos computacionales asociados al entrenamiento por Backpropagation y Descenso por Gradientes, como poder ganar interpretabilidad en los parámetros utilizados por la red al momento de inferencia.

La metodología de entrenamiento empleada por los autores utiliza PCA para las *n* primeras capas de una red convolucional, a partir de los autovectores calculados. La cantidad de capas convolucionales sobre las que se implementa PCA se calculan en función del tamaño de la imagen de entrada y del tamaño de cada uno de los filtros de las capas, de modo tal que luego de todas las capas generadas por PCA se aplique una última capa generada por medio de LDA con un campo receptivo que abarque la totalidad del área de la imagen de entrada. Detrás de esta metodología, yace la hipótesis por parte de los autores, de que solo es posible utilizar un método supervisado con separación por clases, una vez que que los filtros operan sobre un campo receptivo que contempla la totalidad de la imagen.

Otro aporte en la utilización de PCA para el entrenamiento de filtros convolucionales es el de Sun et al [29]. En este trabajo los autores utilizan dos capas convolucionales generados por medio de los autovectores de la información de entrada, aplicando luego etapas de hashing y de clasificación por medio de SVM. El trabajo está enfocado en la detección temprana de signos de enfermedad en cerdos. El empleo de PCA en este caso no está motivado por la búsqueda de una mejora en el desempeño final del modelo sino en la obtención de tiempos de inferencia menores al trabajar con una arquitectura poco profunda. Es de particular interés para la presente tesis la búsqueda que realizan los autores del tamaño óptimo de filtro en función de maximizar el desempeño del clasificador. Para un tamaño de ventana de 64 x 64 píxeles, los filtros de 5 x 5 han resultado óptimos, en términos de accuracy.

Otro trabajo de relevancia es el de Tian et al [30], donde los autores también utilizan una metodología similar a PCANet que denominan Stacked PCA Network, para la construcción de filtros convolucionales para la tarea específica de reconocimiento facial. Probablemente el aporte más relevante para la presente tesis resida en los experimentos que los autores realizan respecto a las funciones de activación (no lineales) entre capas convolucionales generadas por

PCA. Luego de numerosos experimentos determinan que el desempeño de la red para la tarea de reconocimiento facial empeora con la utilización de funciones no lineales de activación entre capas convolucionales, por lo que deciden removerlas y acoplarlas de forma directa. Es decir, luego de una capa convolucional, aplican una segunda capa convolucional, sin aplicar ninguna función de activación ni operación de Pooling.

En el dominio de la clasificación de ruido en imágenes, el trabajo de Khaw et al [31] propone una implementación para el entrenamiento de filtros convolucionales a partir de los autovectores de PCA, muy similar a la realizada en PCANet [25]. Sin embargo, es remarcable el logro de los autores al alcanzar reducir los tiempos de entrenamiento significativamente al utilizar Backpropagation y Descenso por Gradientes para entrenar solamente la última capa totalmente conectada de la red. Los filtros entrenados por medio de PCA no se utilizan como inicializadores sino que permanecen fijos durante el posterior entrenamiento de la capa totalmente conectada.

2.7.2. Estrategias de entrenamiento e inicialización basadas en K-Medias

El algoritmo de K-Medias es probablemente uno de los algoritmos de agrupamiento no supervisado más empleados en Aprendizaje Automático por su simplicidad y aplicabilidad. Se ha utilizado también en el presente trabajo como estrategia para el entrenamiento de filtros para capas convolucionales, por lo que su abordaje dentro del contexto del marco teórico resulta de vital importancia.

Inicializando k centroides de forma aleatoria, K-Medias realiza varias iteraciones para llevar a estos centroides hacia las posiciones que mejor representen los k agrupamientos predefinidos. Para hacer esto, en cada iteración asigna el centroide más cercano a cada uno de los puntos del conjunto de datos y luego recalcula la posición de cada centroide como el promedio de todos los puntos que le fueron asignados. Este procedimiento se repite durante varias iteraciones para obtener, finalmente, un conjunto de k centroides, uno para cada uno de los grupos.

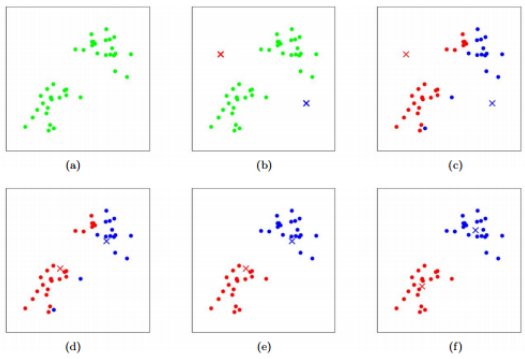


Fig 2.13: K-Medias paso a paso. (a y b) Inicialización de 2 centroides al azar. (c y e) asignación de puntos. (d y f) recalculo de centroides.

Las principales particularidades de K-Medias son la necesidad de definir con antelación el número de agrupamientos que se desea obtener y la utilización de la distancia euclidiana como métrica para determinar la pertenencia de cada punto a uno u otro agrupamiento.

No existen prácticamente antecedentes respecto al empleo de este algoritmo para el entrenamiento de filtros convolucionales de modo no supervisado.

El único trabajo estrechamente relacionado con esta estrategia de que el autor tenga conocimiento es el realizado por Coates et al [32] donde se utiliza el algoritmo K-Medias para el agrupamiento no supervisado de parches (patches) de imágenes para el entrenamiento de filtros visuales. El empleo del algoritmo de K-Medias en el trabajo mencionado, intenta aliviar los costos computacionales asociados tradicionalmente al entrenamiento de varias capas de características (features) de forma jerárquica en una red neuronal. En lugar de alimentar la red directamente con un conjunto de imágenes de entrada, plantean la utilización de este algoritmo para la construcción de filtros visuales. Por medio de estos, se logra un preprocesamiento inicial de las imágenes, aliviando el trabajo de entrenamiento posterior de la red, al alimentarla directamente con los mapas de características extraídos por medio de dichos filtros.

2.8. Métodos de optimización de CNNs a través de la poda de filtros entrenados

Relacionados con el presente trabajo, existen numerosas investigaciones respecto al proceso de poda de filtros convolucionales que persiguen el objeto de optimizar una red convolucional. Estos métodos de optimización buscan obtener modelos más livianos y eficientes, tanto en términos de cantidad de parámetros como de empleo de recursos computacionales en tiempo de inferencia.

En la mayoría de estos trabajos, el método de entrenamiento de la red es el tradicional, es decir, supervisado y por medio de GDBP, pero una vez entrenada la red se procede a realizar una selección de los filtros más relevantes de acuerdo a diferentes criterios.

En el caso de RoyChowdhury et al [33] los autores seleccionan filtros que consideran redundantes y los eliminan por medio de agrupamiento no supervisado. Utilizan una matriz de

similitud entre pares de filtros basada en la norma L2 y confeccionan un grafo a partir de ella. Para cada componente conexa, conservan un único filtro y eliminan el resto.

Por medio de un procedimiento semejante, en Son et al [34] se realiza un agrupamiento no supervisado de los filtros por medio del algoritmo de K-Medias. En este caso, todos los filtros obtenidos por medio del entrenamiento por GDBP son reemplazados por los centroides hallados durante el proceso de agrupamiento, permitiendo reducir las dimensiones de la red.

Utilizando también K-Medias y adicionalmente agrupamiento aglomerativo con distancia euclideana, Wu et al [35] realiza un agrupamiento de los filtros de modo indirecto, reduciendo previamente su dimensionalidad por medio de PCA. Por medio de la métrica de Silhouette determinan el número óptimo de grupos. Finalmente seleccionan un único filtro representativo por grupo y eliminan el resto.

De modo muy similar, Ghosh et al [36] utiliza también K-Medias y agrupamiento aglomerativo, pero de modo directo sobre los filtros. A diferencia de Wu et al [35] se utiliza la correlación como métrica de similitud y se descartan los filtros redundantes.

Particularmente distintivo es el método propuesto por Din et al [37] donde no se recurre a la poda de filtros posterior al entrenamiento de la red sino que se la incorpora como parte del entrenamiento. En cada iteración del descenso por gradientes, se realiza un agrupamiento de filtros para cada capa por medio de K-Medias. Adicionando un término a la función de error, se induce a la red a incrementar la cohesión interna de los grupos. Una vez finalizado el entrenamiento, se obtienen una cantidad de filtros F_o por capa, inferior a la cantidad inicial de filtros F_i. Todos los filtros pertenecientes a un mismo grupo resultan idénticos, por lo que la poda de los filtros redundantes resulta completamente trivial.

2.9. Conjuntos de datos: Mnist-Fashion, CIFAR10, ImageNet

A lo largo de este trabajo se han empleado tres conjuntos de datos de amplia utilización en la literatura: CIFAR10 [38], Mnist-Fashion [39] e ImageNet [18]. A continuación se describen brevemente cada uno de ellos:

2.9.1. CIFAR10

CIFAR10 [38] consta de 50.000 imágenes para entrenamiento y otras 10.000 para evaluación. Se trata de un conjunto de imágenes de 32x32 píxeles en color (3 canales: RGB) pertenecientes a 10 clases o categorías predefinidas: automóviles, aviones, aves, gatos, venados, perros, ranas, caballos, barcos y camiones. El conjunto de datos está perfectamente balanceado, proporcionando 5.000 ejemplos de entrenamiento y 1.000 ejemplos de evaluación para cada clase. CIFAR10 fue creado por Alex Krizhevsky, Vinod Nair, y Geoffrey Hinton.

A continuación se visualizan algunos ejemplos del conjunto de datos:

Fig 2.14: Algunos ejemplos del conjunto de datos CIFAR10 y sus clases de pertenencia.

2.9.2. Mnist-Fashion

Mnist-Fashion [39] es también uno de los conjuntos de imágenes más utilizados en los trabajos relacionados con Visión Artificial. Consta de 60.000 ejemplos de entrenamiento y otros 10.000 de evaluación, con imágenes de 28x28 píxeles en escala de grises (1 canal). Se trata de un conjunto de imágenes de productos de moda que pertenecen a una de las siguientes 10 clases: remera, pantalón, pullover, vestido, saco, sandalias, camisa, zapatillas, bolso, botines. El conjunto de datos se encuentra perfectamente balanceado, con 6.000 ejemplos de entrenamiento y 1.000 de evaluación por clase.

Algunos ejemplos de imágenes que componen Mnist-Fashion:

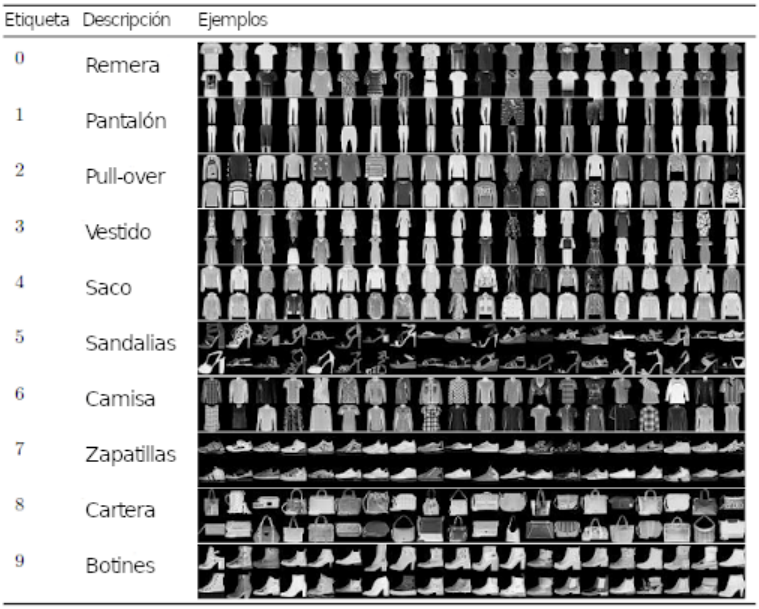


Fig 2.15: Ejemplos de imágenes de Mnist-Fashion y sus clases de pertenencia.

2.9.3. ImageNet

Prácticamente desde su introducción, ImageNet [18] se ha convertido en el conjunto de datos más empleado en los trabajos relacionados con Visión Artificial. Su contribución al avance del campo de Visión Artificial es invaluable y ha sido utilizado como punto de referencia para la comparación de diferentes algoritmos por gran parte de la comunidad.

Existen muchas versiones de este conjunto de datos pero probablemente la más empleada y referenciada en la literatura sea ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Este subconjunto específico de ImageNet consta de 1.281.167 imágenes de entrenamiento, 50.000 de validación y 100.000 de evaluación. Comprende un total de 1000 clases diferentes para la tarea de clasificación de imágenes. Adicionalmente a las etiquetas de clase, incluye las coordenadas del objeto principal de cada imagen para ser utilizadas en modelos aplicados a la tarea de localización de objetos.

Las imágenes son en color (3 canales: RGB) y tienen un formato cuadrado estandarizado de 224 píxeles de lado. Las clases del conjunto de datos se han tomado a partir de una estructura jerárquica de sustantivos definida en el proyecto WordNet [40].

Algunas imágenes contenidas en el conjunto de datos:

Fig 2.16: Algunas imágenes del conjunto ImageNet ILSVRC.

2.10. Modelos Convolucionales: Efficient Net

Desde el trabajo seminal de Yann LeCun [5] en redes convolucionales han surgido numerosos modelos de CNN de diversas características. Muchos de ellos han aportado e implementado nuevas ideas que han logrado avances considerables en el estado del arte. Uno de estos avances en Visión Artificial y particularmente en el diseño de arquitecturas convolucionales es el de Efficient Net.

El aporte realizado por Tan et al [41] puede resumirse en dos puntos fundamentales: por un lado, el diseño de una arquitectura base eficiente denominada B0 y por el otro, el desarrollo de políticas de escalamiento para redes convolucionales.

El diseño de la arquitectura base para la versión más pequeña de EfficientNet (B0), a partir de la cual fueron desarrolladas todas las versiones, se realizó a través de la "Búsqueda de Arquitecturas Neuronales" o Neural Architecture Search (NAS), tomando el trabajo previo realizado por Zoph y Le en 2017 [42].

El incentivo central del desarrollo de EfficientNet es el de poder obtener un modelo base y una política para escalar dicho modelo de acuerdo a los recursos computacionales disponibles para cada caso de uso. EfficientNet busca y logra obtener avances en el estado del arte en cuanto a desempeño en clasificación de imágenes sobre ImageNet ILSVRC, pero tomando un enfoque completamente diferente a la mayoría de los avances previos basados en el incremento en profundidad y/o ancho del modelo.

En lugar de incrementar las dimensiones de modelos existentes y destinar mayor presupuesto computacional a su entrenamiento, el enfoque de los autores se centra en la búsqueda de una familia de modelos eficientes. La premisa que guía el trabajo es la de buscar la arquitectura con mejor desempeño en función de un determinado presupuesto computacional, medido específicamente en FLOPS (cantidad de operaciones de punto flotante).

Uno de los aportes más importantes del trabajo de Tan et al [41] es el replanteo del modo óptimo de escalar los modelos convolucionales. Los autores definen tres dimensiones de escalamiento de modelos: profundidad (d), ancho (w) y resolución (r). A través de la experimentación, llegan a la conclusión de que un modelo puede ser escalado en alguna de estas tres dimensiones logrando un incremento en su desempeño, pero con retornos decrecientes.

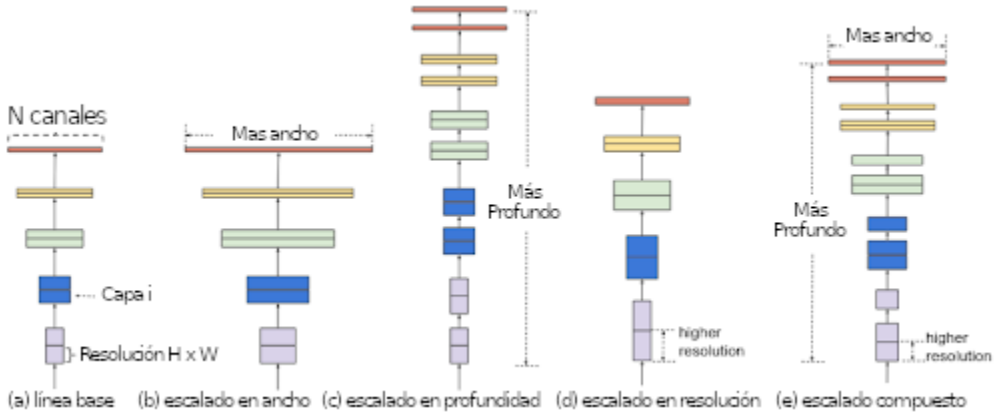


Fig 2.17: Diferentes estrategias de escalamiento de un modelo convolucional. (a) modelo de base de referencia. (b,c,d) escalamiento en ancho, profundidad y resolución. (e) escalamiento compuesto.

Es decir, los incrementos de escala de determinado modelo, si se realizan sobre una sola de las dimensiones, saturan muy rápidamente sus posibilidades de incremento de desempeño. La ganancia marginal que aporta el escalamiento es decreciente.

Por otra parte, si se escalan las tres dimensiones (*d, w, r*) de un modelo de forma coordinada, se logran incrementos de desempeño mucho más eficientes. El retorno es mucho mayor, en

términos del incremento de desempeño obtenido en función del presupuesto en FLOPS destinado.

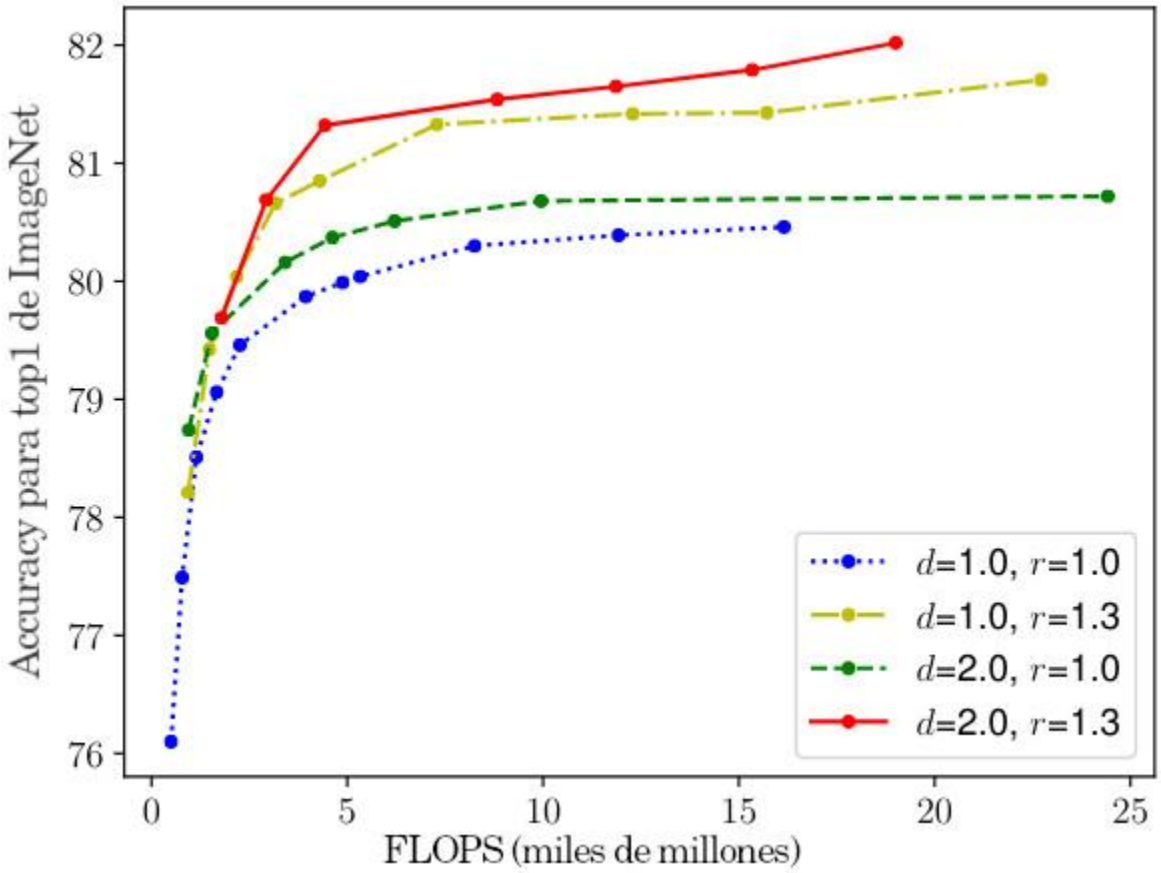


Fig 2.18: Desempeño obtenido en función del presupuesto en FLOPS para diferentes escalamientos del modelo base en profundidad (d) y resolución.

Tal como se aprecia en la figura 2.18, escalando solamente en profundidad (curva verde) o solo en resolución (curva roja) el escalamiento es menos eficiente en términos de costo-beneficio (FLOPS-Accuracy) que si se escala el modelo base en ambas dimensiones de forma coordinada.

Más allá de las métricas de desempeño y eficiencia, los Mapas de Activación de Clase CAM (*Class Activation Map*) de EfficientNet proveen una excelente referencia del desempeño de esta arquitectura. Los Mapas de Activación de Clase son una técnica desarrollada por Zhou et al [43] que permiten visualizar mediante un mapa de calor, qué regiones de determinada imagen de entrada son más relevantes para determinado modelo convolucional para generar el vector de predicciones de clase a la salida.

Tal como se puede apreciar en la figura 2.18, el escalamiento en una sola dimensión (profundidad, ancho o resolución) no logra capturar con precisión las regiones de mayor

importancia de la imagen para realizar la clasificación. El escalamiento coordinado o compuesto, por otra parte, parece lograrlo con mucha mayor precisión.

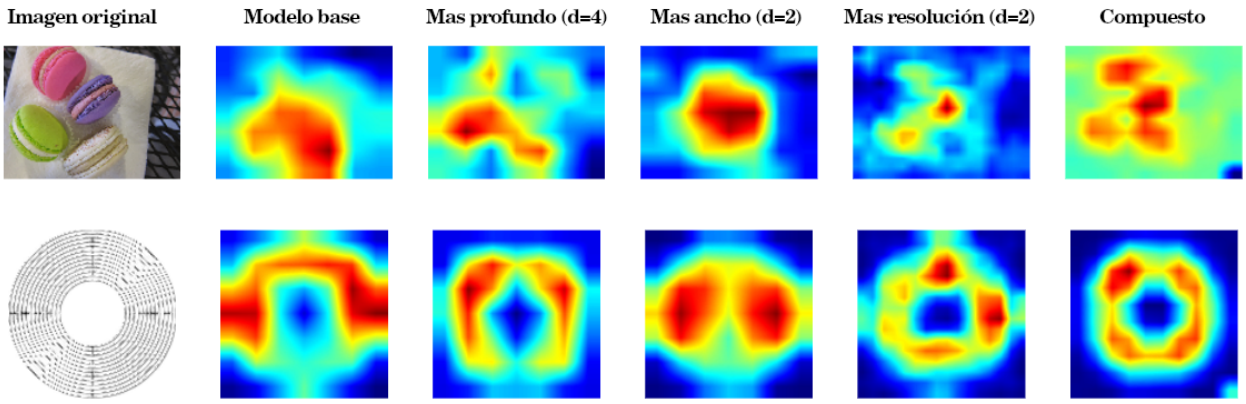


Fig 2.19: Mapas de activación de clase para modelo base, modelo escalado en profundidad, en ancho, en resolución y escalamiento compuesto.

En cuanto a la arquitectura del modelo base presentada por los autores, ésta consta de 8 bloques convolucionales y un último bloque de salida con la capa densa o totalmente conectada. A excepción del primer bloque, los bloques 2 a 8 utilizan una estrategia de "cuello de botella invertido" o *MBConv* basado en el trabajo previo Sandler et al. [44].

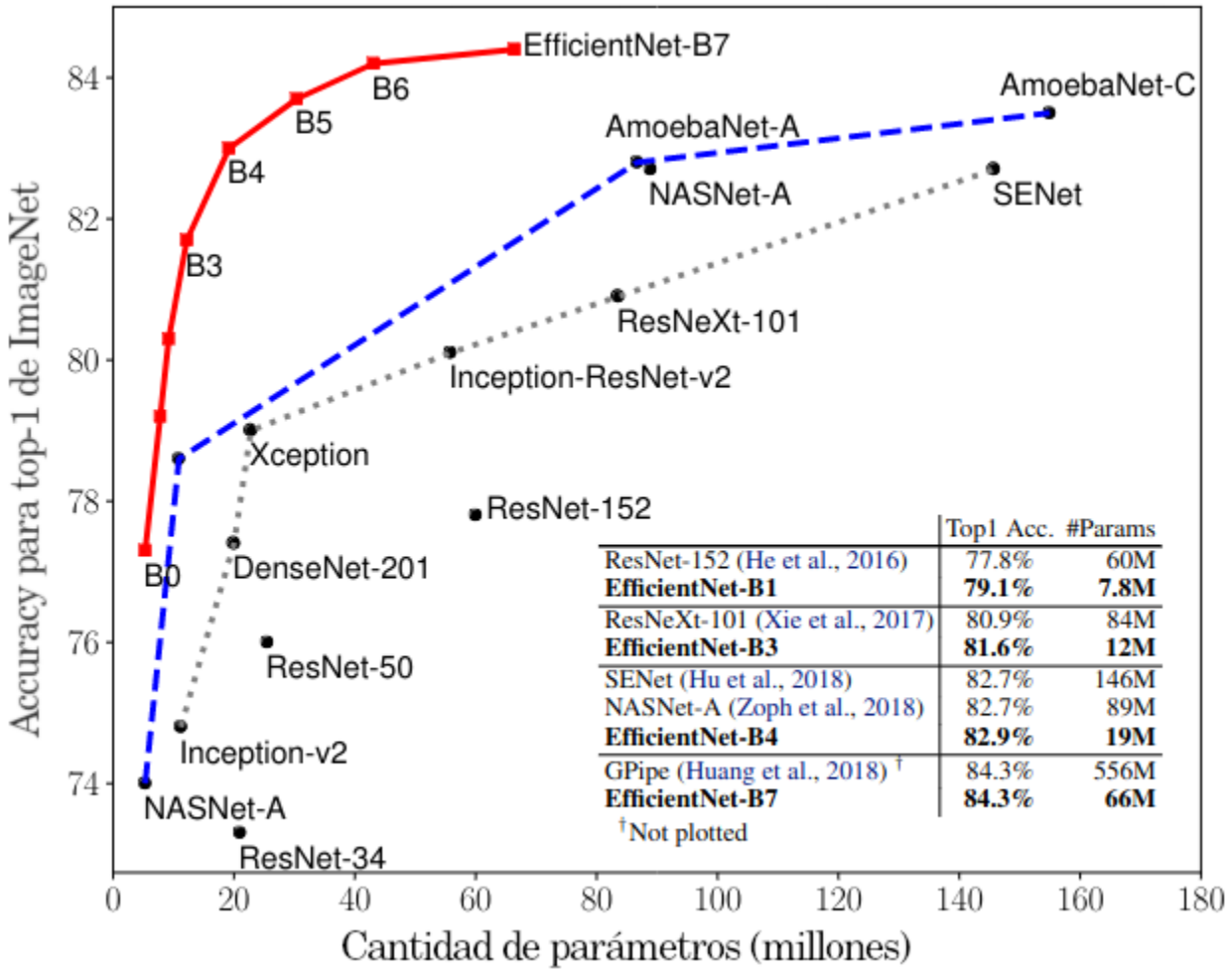
Los bloques MBConv utilizan conexiones residuales entre la entrada y la salida del bloque. Esta estrategia ha resultado muy exitosa para facilitar el entrenamiento de redes neuronales de muchas capas, logrando mitigar los efectos del fenómeno denominado gradiente desvaneciente (*vanishing gradient*) producto de realizar Backpropagation a lo largo de muchas capas.

Etapa i	Operador $\hat{\mathcal{F}}_i$	Resolución $\hat{H}_i \times \hat{W}_i$	N Canales $\hat{C}_i$	N Capas $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Fig 2.20: Arquitectura base de EfficientNet, compuesta de 9 etapas o bloques.

Asimismo, cada bloque está compuesto de convoluciones separables en profundidad (*depth-wise separable convolutions*) que permiten realizar una drástica reducción en la cantidad de paŕametros empleados. A diferencia de las convoluciones estándar, las convoluciones separables en profundidad realizan el proceso en 2 etapas. Primero se efectúa una operación de convolución para cada canal de forma independiente y luego una segunda operación de convolución punto a punto (*point-wise convolution*) para capturar la correlación entre canales.

EfficientNet logra un nuevo avance en el estado del arte a través de una nueva arquitectura base y una familia de modelos producto de la política de escalamiento desarrollada. Los resultados obtenidos en cuanto a eficiencia, medida en términos de accuracy en función de la cantidad de parámetros del modelo, pueden visualizarse en la figura 2.21.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Fig 2.21: Eficiencia de la familia de modelos EfficientNet en comparación con otros modelos de alto rendimiento.

3. Exploración de filtros convolucionales en arquitecturas de alto desempeño

La etapa inicial del presente trabajo consistió en la exploración de filtros convolucionales ya entrenados por el método tradicional de GDBP. Se tomaron varios modelos de alto rendimiento pre-entrenados sobre el conjunto de imágenes de ImageNet y se analizaron los filtros generados para la primera capa de cada uno de ellos.

El objetivo detrás de estos experimentos fue obtener una mayor comprensión respecto a las características del espacio de soluciones al que arriban diferentes arquitecturas convolucionales por medio de GDBP. La detección de regularidades en este espacio puede aportar al entendimiento de las características que hacen a la utilidad de un filtro convolucional en el contexto de la clasificación de imágenes.

En la sección 3.1 del presente capítulo se desarrollan diferentes experimentos con la finalidad de determinar una métrica de similitud entre filtros convolucionales adecuada. Es decir, aquella que mejor refleje la similitud entre mapas de características obtenidos por los diferentes filtros.

Por supuesto, la métrica de similitud óptima para comparar filtros es, justamente, la comparación de los mapas de características que se obtienen de ellos. Sin embargo, resulta de particular interés encontrar una métrica de distancia directa entre filtros que refleje la similitud entre los mapas de características que producen. Esto permite tanto profundizar en el estudio de los filtros en el contexto de la operación de convolución, como eliminar las dependencias respecto a un conjunto de datos en particular. Adicionalmente, tiene la ventaja de reducir los costos computacionales.

Seguidamente, en la sección 3.2 se realizan diferentes agrupaciones de filtros de modelos pre entrenados, tomando como métrica de similitud para realizar dichos agrupamientos, la métrica

más adecuada, de acuerdo a los experimentos realizados en el apartado 3.1. La finalidad de dichos agrupamientos es la de poder validar las hipótesis planteadas respecto a dos puntos. Por un lado, la presencia de filtros redundantes en un mismo modelo, lo que avalaría la hipótesis del carácter sub óptimo de las soluciones generadas por GDBP. Por otro lado, la presencia de filtros similares en diferentes modelos. Es decir, la tendencia a encontrar soluciones con filtros coincidentes en diferentes arquitecturas, con diferentes hiperparámetros de entrenamiento. Esto avalaría la hipótesis respecto a las restricciones en el espacio de soluciones posibles.

3.1. Métrica de similitud

Para poder definir una métrica de similitud directa entre pares de filtros, se compararon varias métricas candidatas contra un valor de referencia. Este valor de referencia es la similitud entre los mapas de características producidos por los filtros al realizar la operación de convolución.

Para poder obtener este valor de referencia, debió considerarse, en primera instancia, la selección de una métrica de similitud entre mapas de características adecuada. Si bien los mapas de características no son estrictamente imágenes, resulta razonable considerarlos de este modo a los fines de la selección de la métrica de distancia más apropiada.

Existen numerosos métodos y métricas para realizar la comparación entre imágenes. Dependerá del caso de uso y del objetivo de la comparación, cuál de ellos resulta más adecuado. Para este caso particular, donde no se busca evaluar la similitud semántica de un par de imágenes sino su contenido literal, una de las métricas más empleadas es la distancia euclídea. No existe un consenso absoluto al respecto en la literatura, pero es esta métrica la que se encuentra con cierta frecuencia en numerosos trabajos como los de RoyChowdhury et al [33] y Zagoruyko et al [45].

A partir de lo considerado en el párrafo anterior, se utilizó la distancia euclídea como métrica de evaluación de similitud entre mapas de características. A partir de este valor de referencia para los mapas de características producidos se compararon diferentes métricas de similitud directa entre filtros: la similitud coseno, el complemento de la distancia euclidiana y el producto escalar.

La elección de estas métricas se basa en los siguientes criterios: tanto la similitud coseno como el producto escalar están estrechamente relacionados con la operación de convolución, siendo la primera, la versión normalizada de la segunda. En cuanto a la distancia euclidiana, es un

método directo de medir la diferencia entre imágenes, píxel a píxel, penalizando especialmente las diferencias más importantes en la comparación a nivel píxel.

Se realizaron experimentos para determinar cuál de las tres métricas mencionadas refleja de modo más consistente la real similitud entre filtros. Es decir, cuál de ellas se asemeja en mayor grado al valor de referencia establecido por la comparación de mapas de características.

Se entenderá por real similitud entre filtros lo expresado por la siguiente definición:

> *Dos filtros son similares si producen el mismo escalar de activación al convolucionarse con el mismo segmento de imagen. Llevado esto a un dominio continuo, dos filtros (o más) serán tan similares entre sí como lo sean sus respectivas activaciones al realizar la convolución con un mismo conjunto de segmentos o parches (patches) de imagen. Lo inverso también debe darse. Es decir, dos filtros muy disímiles producirán dos salidas muy disímiles entre sí.*

Para realizar los experimentos se utilizaron los conjuntos de datos que se detallan a continuación.

3.1.1. Conjunto de datos empleados

Los datos que se utilizaron para este análisis exploratorio corresponden a los parámetros aprendidos por la primera capa convolucional de diferentes modelos de redes convolucionales durante su entrenamiento en el corpus ImageNet.

En una red convolucional, cada capa está compuesta por un número N de filtros o kernels. La cantidad N de filtros utilizados en la primera capa para cada modelo es variable, pero suelen ser los más habituales 32 y 64. Por su parte, los filtros que se utilizan en cada capa pueden tener diferentes dimensiones, hasta cierto punto arbitrarias.

Un filtro en una red convolucional posee tres dimensiones: un ancho, un alto y una profundidad o número de canales. En todos las redes que se incluirán en este trabajo, el ancho y el alto de los filtros es el mismo, es decir, son filtros cuadrados. Para la primera capa, la mayoría de los modelos pre entrenados para imágenes a color implementa filtros de tamaño 3x3x3. La cantidad de canales, es siempre 3 para la primera capa, en concordancia con los 3 canales que poseen las imágenes de entrada (rojo, verde y azul). En el presente trabajo se utilizaron solamente los filtros de aquellas redes pre entrenadas que poseen dimensiones de 3x3x3. Con esta limitación se mantiene fija esta variable y se despeja la variabilidad que este factor pudiera introducir en los experimentos.

Se utilizaron entonces los filtros de la primera capa aprendidos sobre ImageNet para los siguientes modelos:

- EfficientNetV2L
- InceptionV3
- MobileNetV3Large
- NASNetLarge
- VGG19
- Xception

Para cada uno de estos modelos, se visualizan a continuación sus filtros para la primera capa convolucional, de acuerdo a su entrenamiento sobre el corpus ImageNet.

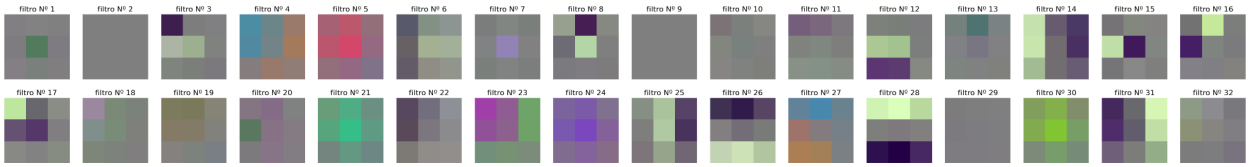


Fig 3.1: Filtros de la primera capa convolucional de EfficientNet V2L pre entrenada en ImageNet.

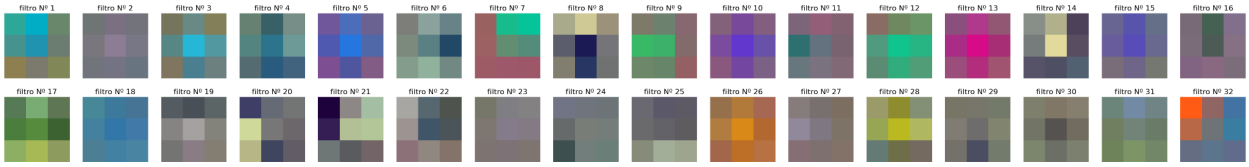


Fig 3.2: Filtros de la primera capa convolucional de Inception V3 pre entrenada en ImageNet.

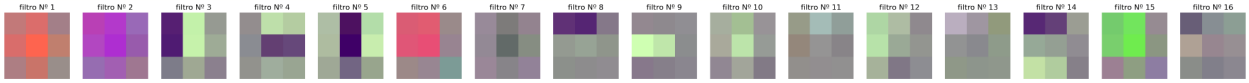


Fig 3.3: Filtros de la primera capa convolucional de MobileNet V3L pre entrenada en ImageNet.

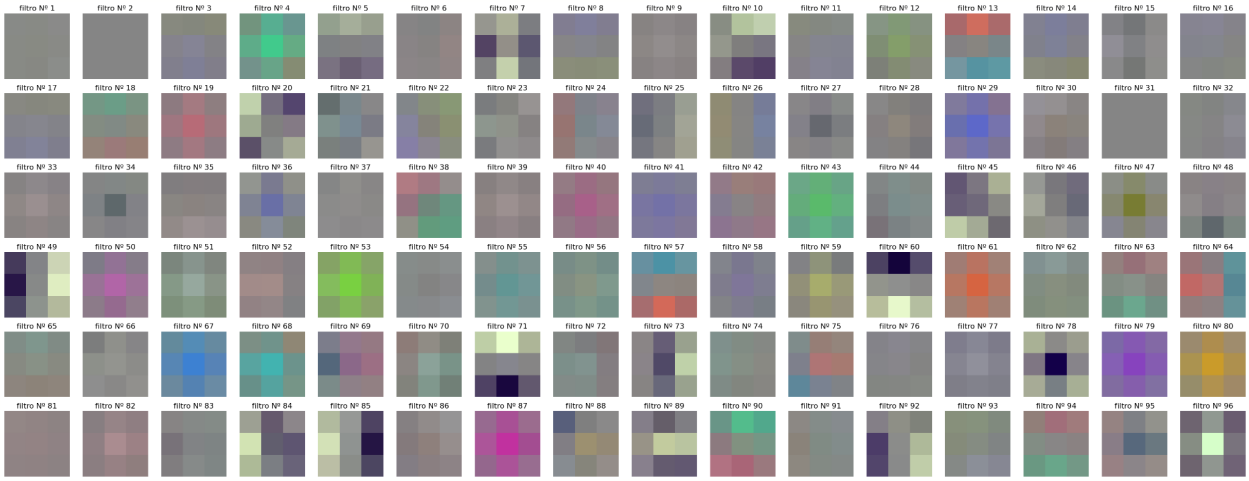


Fig 3.4: Filtros de la primera capa convolucional de NasNet L pre entrenada en ImageNet.

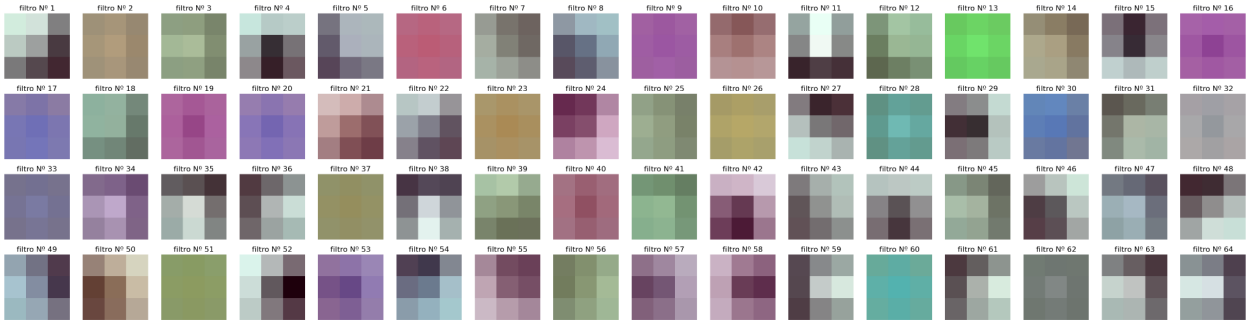


Fig 3.5: Filtros de la primera capa convolucional de VGG19 V3L pre entrenada en ImageNet.

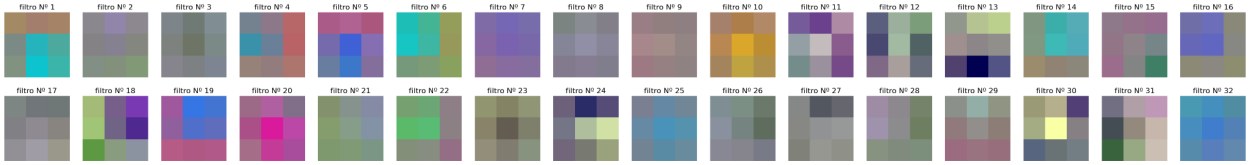


Fig 3.6: Filtros de la primera capa convolucional de Xception pre entrenada en ImageNet.

Los modelos son de público uso y acceso. Si bien existen varios modos de obtenerlos, el más conveniente, en caso de que se esté trabajando con la API de Tensorflow-Keras, es directamente descargar los modelos pre entrenados utilizando esta misma API. Una vez en poder de los modelos, basta con eliminar las capas restantes y conservar únicamente la primera de cada uno de ellos.

Adicionalmente se utilizó el conjunto de datos CIFAR10. Este conjunto de datos es de amplia utilización en la literatura de redes convolucionales. Por la dimensión reducida de sus imágenes resulta muy adecuado para realizar experimentos de forma ágil, sobre infraestructura reducida. Es por ello que se ha utilizado en los experimentos de entrenamiento en capítulos posteriores y se emplea aquí también, para realizar la exploración de filtros convolucionales sobre modelos pre entrenados.

3.1.2. Preprocesamiento

Como se mencionó en el apartado anterior, todos los filtros utilizados en este trabajo presentan las mismas dimensiones, lo que evita la necesidad de realizar una adecuación de los datos en este sentido. Todos ellos son de dimensiones 3x3x3 por lo que pueden compararse de forma directa, tanto en su forma tridimensional como unidimensional, como vectores de 27 dimensiones.

Más allá de la descarga de los modelos individualmente y la extracción de la primera capa de cada uno de ellos, el preprocesamiento realizado es mínimo. Simplemente se apilan todos ellos en una misma lista unificada que los contiene. Adicionalmente se calculan y conservan los valores máximos y mínimos que toman sus parámetros, para cada modelo de forma independiente. Es decir, el cálculo de estos dos estadísticos se realiza para cada capa y se registra en una lista independiente. La finalidad de este procedimiento es la de facilitar la visualización adecuada de los filtros y poder calcular distancias entre pares de filtros que pueden pertenecer a diferentes modelos. Más allá de estas consideraciones, los filtros son utilizados de forma directa al momento de realizar las operaciones de convolución.

3.1.3. Experimentos realizados

Para llevar a cabo la selección de la métrica más adecuada, se tomaron todos los filtros provenientes de la primera capa de los modelos pre-entrenados mencionados (272) y considerando la definición de similitud entre filtros convolucionales propuesta, se procedió de la siguiente forma:

1. Se construyó una red neuronal con una única capa convolucional con los 272 filtros de dimensión 3x3x3 apilados provenientes de la primera capa de todos los modelos. Se utilizó zancada (*stride*) de 1 y sin adicionar márgenes (*padding* "*valid*" o cero). Finalmente se aplicó la función de activación ReLU.

2. Se ejecutó el modelo en modo inferencia o *Forward Pass* sobre el conjunto de datos CIFAR10. Se utilizó este conjunto de datos por sus reducidas dimensiones (32x32 píxeles a 3 canales) que permiten acelerar los tiempos de inferencia. Es decir, se realizó la operación de convolución sobre todas las imágenes con los 272 filtros. Se obtuvo así un tensor de salida con los mapas de características o *feature maps* de todos y cada uno de los 272 filtros sobre todas y cada una de las imágenes.

3. A partir de este tensor de salida, se calculó la norma euclidiana para la diferencia pixel a pixel, entre todos los mapas de características (feature maps) producidos por dos filtros diferentes. Esto se realiza para cada combinación de filtros. De este modo, se obtuvo una matriz de distancias de dimensión 272 x 272. Cada entrada de esta matriz corresponde a la distancia euclidiana entre los mapas de característica producidos por el filtro de la columna seleccionada respecto a los producidos por la fila seleccionada.

4. De forma independiente a lo trabajado en los 3 primeros puntos, se generaron 3 matrices de similitud, una para cada una de las métricas candidatas seleccionadas:

    - Producto escalar
    - Similitud coseno
    - Complemento a la distancia Euclídea.

    Cada una de estas matrices presenta las mismas dimensiones: 272 x 272, donde cada entrada corresponde a la similitud dada por la métrica en cuestión para cada combinación de par de filtros.

5. Finalmente, se comparó la matriz producida en el punto 3, con cada una de las matrices producidas en 4.

    La matriz producida en 3 podría considerarse como el valor de referencia en cuanto a similitud entre mapas de características. Es decir, esta matriz representa la similitud entre pares de filtros, tomando como similitud la definición presentada en el apartado anterior: dos o más filtros serán tan similares entre sí, cuanto más similares sean los mapas de características que producen sobre un mismo conjunto de imágenes.

    Por su parte, las matrices producidas en el punto 4 se obtuvieron directamente de la comparación entre pares de filtros. Al comparar cada una de estas matrices

con la verdad de campo, se busca determinar cuál de ellas resulta más adecuada para establecer la similitud entre dos o más filtros.

Para realizar esta comparación se utilizó la distancia euclidiana y la distancia Manhattan sobre la diferencia punto a punto entre cada par de matrices. Es decir, la verdad de campo comparada con cada una de las 3 matrices producidas en el punto 4.

3.1.4. Similitud entre filtros a partir de los mapas de característica

A partir del procedimiento mencionado en el apartado anterior, se obtuvo la siguiente matriz de distancias, correspondiente a la comparación de las salidas producidas por cada par de filtros. Se visualiza la matriz en forma de complemento, con los valores invertidos, de tal modo de observar similitudes en lugar de distancias. Un tono más claro equivale a mayor similitud entre un par de filtros. La máxima similitud, por supuesto, se observa en la diagonal de la matriz, donde se compara cada filtro consigo mismo.

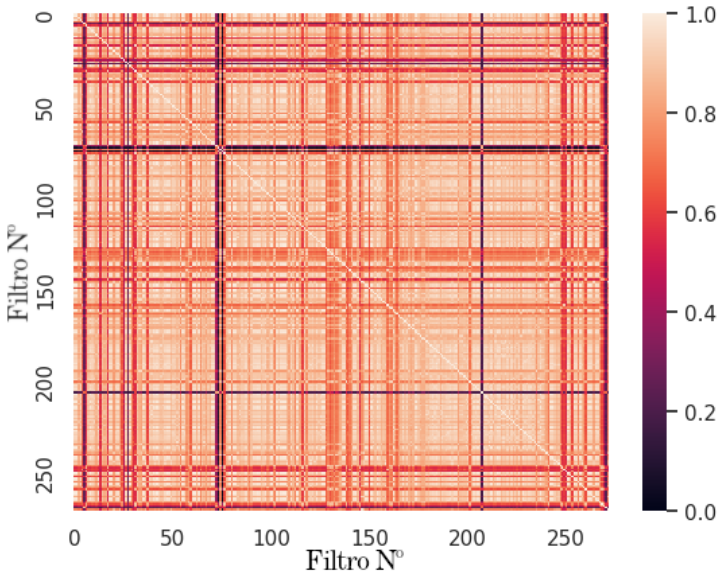


Fig 3.7: Matriz de similitud euclidiana entre mapas de características producidas por los 272 filtros.

Adicionalmente se muestran a continuación algunos de los pares de filtros que han producido los mapas de características con menores distancias, así como los mapas de características producidos por ambos para una misma imágen de referencia.

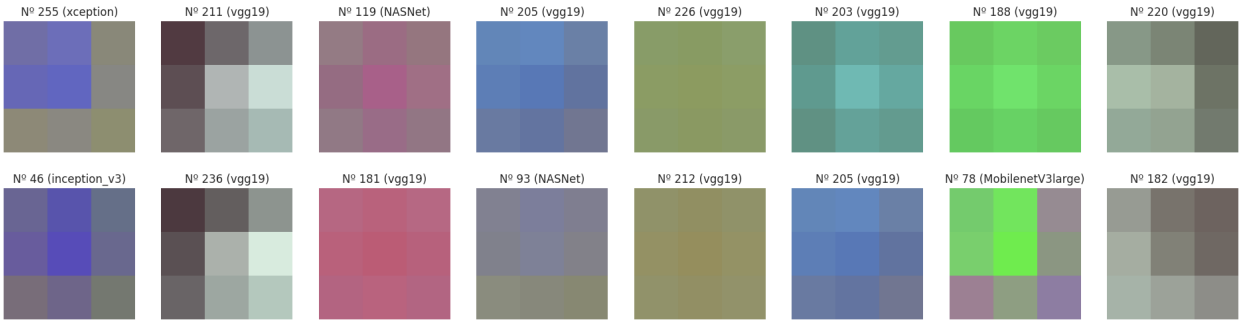


Fig 3.8: Pares de filtros convolucionales provenientes del conjunto de 272 filtros con alta similitud.

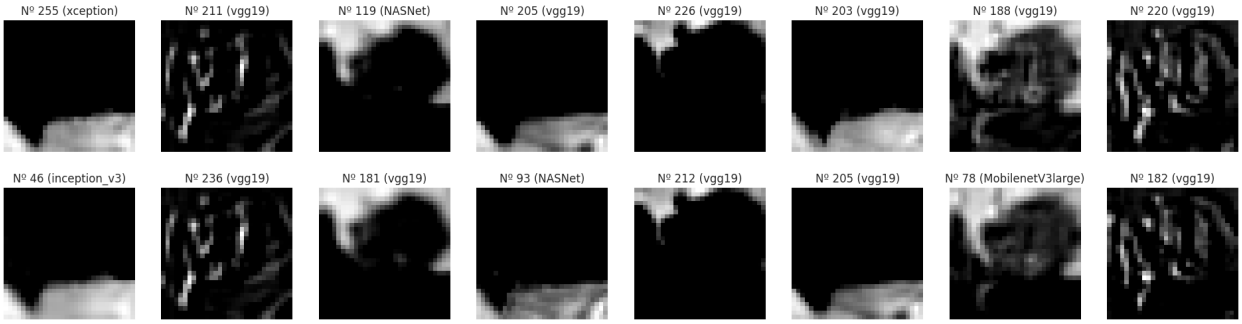


Fig 3.9: Mapas de características generados para una imagen de referencia. En la fila superior e inferior se muestran los mapas de características generados por los pares de filtros visualizados en la figura anterior.

Como se puede apreciar, existe cierta correlación entre la semejanza visual de un par de filtros y los mapas de características que producen al realizar la operación de convolución. Sin embargo, a juzgar por la inspección visual de las similitudes entre pares de filtros y pares de mapas de características, no parece tratarse de una correlación perfecta.

Otro punto a resaltar es la omnipresencia del filtro Nº 75 (perteneciente a Mobilenet V3 Large) como aquel que genera mapas de características con la máxima distancia con respecto a todos los filtros seleccionados para visualización. A simple vista, nada parece particularmente remarcable o evidente acerca del filtro 75 y el por qué de la generación de mapas de características con máxima distancia respecto a los mapas generados por otros filtros:

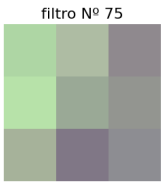


Fig 3.10: Filtro Nº 75 del modelo Mobilenet V3 large entrenado con ImageNet. Este filtro es el que mayor distancia presenta con respecto a todos los pares ilustrados en figura 3.8.

3.1.5. Similitud directa y similitud de mapas de características

A continuación se pueden visualizar las 3 matrices de similitud directa entre filtros, utilizando las 3 métricas propuestas para este trabajo. Cada punto de estas matrices representa en escala de grises, el grado de similitud entre el número de filtro correspondiente a su columna y el correspondiente a su fila:

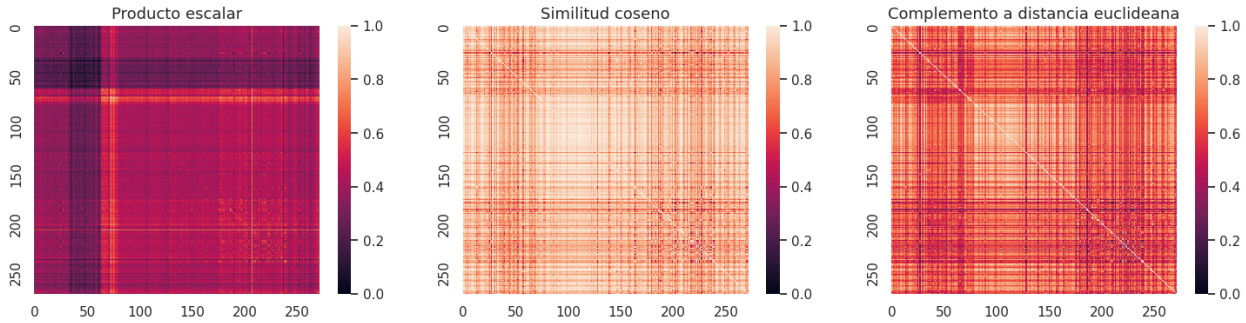


Fig 3.11: Matrices de similitud directa entre filtros con métricas de producto escalar, similitud coseno y similitud euclidiana.

Tanto en el caso de la matriz de similitud coseno como en el del complemento a la distancia euclidiana se observa con claridad la diagonal de máxima similitud, trazada por la comparación de dos filtros idénticos. En el caso del producto escalar, esta diagonal es prácticamente indistinguible, lo que aporta un argumento fuerte en contra de la calidad de esta métrica. Dos filtros idénticos, por supuesto, deberían presentar una similitud máxima. Este punto presenta evidencia de la importancia del ángulo del vector que forma un filtro al ser vectorizado, por sobre su magnitud, que es fácilmente escalable por la red en capas subsiguientes.

Finalmente, se comparan estas tres matrices obtenidas de forma directa sobre los filtros, con la matriz obtenida en la comparación de mapas de características. La distribución de valores de similitud para las 4 matrices es la siguiente:

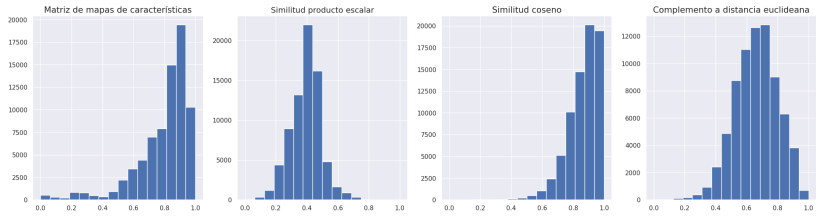


Fig 3.12: Distribución de valores de similitud para las matrices de mapas de características, producto escalar entre filtros, similitud coseno entre filtros y similitud euclidiana entre filtros.

Se observa una mayor similitud en la distribución de la matriz de mapas de características con la generada por la comparación de filtros utilizando la distancia coseno.

En la siguiente figura se observa el resultado de la diferencia absoluta entre cada una de las tres matrices de similitud directa sobre filtros y la matriz de similitud de mapas de características.

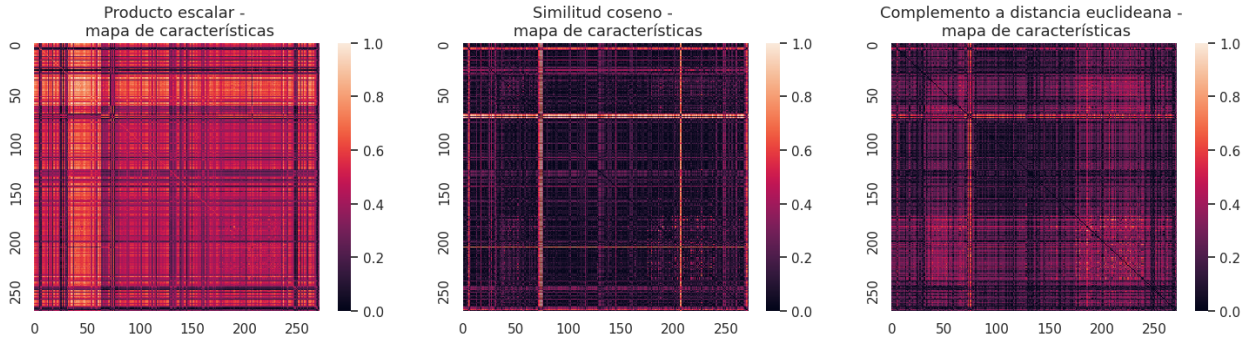


Fig 3.13: Restas entre las tres matrices de similitud directa entre filtros y la matriz de similitud de mapas de características.

Al tratarse de una resta, cuanto más oscura resulta la imagen, mayor similitud representa entre los elementos. La inspección visual permite apreciar claramente que la similitud coseno entre filtros refleja de forma más adecuada la similitud entre los mapas de características generados por dichos filtros al convolucionarse con las imágenes del conjunto de datos.

En los siguientes histogramas se corrobora el mismo resultado, observando como el total de discrepancias en las comparaciones es mucho menor para el caso de la similitud coseno.

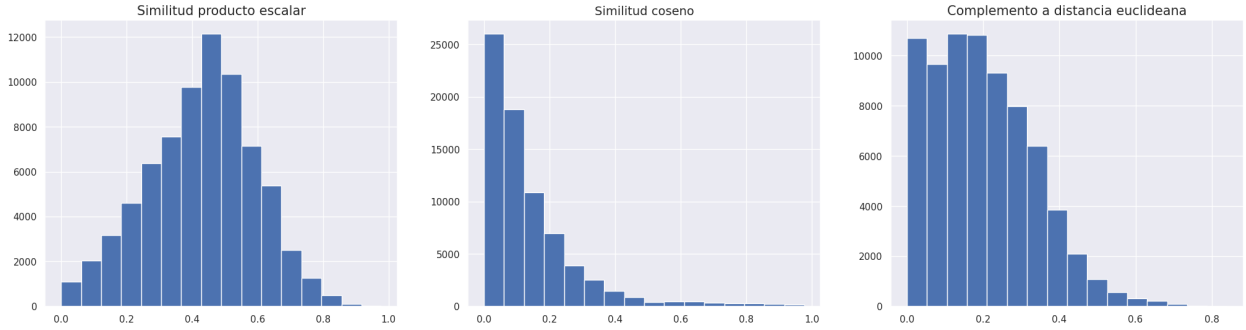


Fig 3.14: distribución de valores de diferencia absoluta entre las tres matrices de similitud directa y la matriz de similitud de mapas de características.

Más allá de la inspección visual, las normas L1 y L2 de las diferencias y la distancia coseno entre matrices aportan datos cuantitativos más precisos:

Métrica similitud	L1 (matriz filtros - matriz mapas características)	L2 (matriz filtros - matriz mapas características)	Distancia coseno (matriz filtros y matriz mapas características)
Producto escalar	31663	124.72	0.0673
Similitud coseno	10223	55.04	0.0267
Complemento a la distancia euclídea	14821	65.33	0.0353

Tabla 3.1. Comparación entre la matriz de distancias entre mapas de características y las distintas matrices de distancia directa entre filtros. Los valores absolutos obtenidos no tienen ningún valor interpretable por sí mismos. Son sólo útiles como medio para realizar las comparaciones entre las diferentes métricas estudiadas.

Nuevamente, los resultados apoyan a la similitud coseno como métrica de similitud entre filtros más adecuada para aproximar la similitud que los mapas de características producidos por dichos filtros tendrán. Utilizando tanto la distancia Manhattan (L1) como la distancia euclidiana (L2) y la distancia coseno, se arriba a la misma conclusión. La diferencia acumulada entre la matriz de mapas de características producidos por los filtros y la matriz de similitud directa entre filtros, es menor, en ambos casos, para la similitud coseno.

3.2. Agrupamiento de filtros

A partir de la métrica definida en la sección anterior como la más adecuada para reflejar la similitud entre filtros, se utilizaron dos técnicas de agrupamiento de filtros diferentes.

El objetivo de los agrupamientos es el de comprender de forma global el conjunto de filtros con el que se ha trabajado y adicionalmente encontrar evidencia respaldatoria para las hipótesis planteadas:

- La presencia de filtros redundantes dentro de un mismo modelo (filtros que obtienen mapas de características muy similares) dando cuenta del carácter subóptimo de las soluciones.
- La presencia de filtros muy similares en diferentes modelos, apoyando la hipótesis de la restricción en el espacio de soluciones posibles.

En cuanto a las técnicas de agrupamiento, por un lado se empleó DBScan con un número mínimo de elementos por grupo de 2, buscando el valor de epsilon que maximiza el score DBCV. El empleo de DBCV como métrica para valorar la calidad de los agrupamientos, es más adecuada para algoritmos como DBScan, donde los agrupamientos generados no tienen necesariamente una forma convexa. Al contrario de Silhouette, que presenta un sesgo favorable a agrupamientos convexos, DBCV permite evaluar agrupamientos de cualquier forma. Se basa en la densidad del espacio, por lo que resulta más adecuada para evaluar la calidad de agrupamientos generados por DBScan, también basados en densidad.

Adicionalmente, se utilizó el algoritmo de agrupamiento aglomerativo con enlace completo, buscando, mediante búsqueda en grilla, el número de grupos que maximiza el score DBCV.

No se utilizó el algoritmo K-Medias por la naturaleza propia de su implementación que implica la utilización de la distancia euclidiana para estimar los grupos [2].

3.2.1. Agrupamiento por medio de DBScan

Para el agrupamiento por medio de DBScan, se estableció la mínima cantidad de elementos por grupo en 2. Por medio de búsqueda en grilla, se encontró el valor óptimo de distancia epsilon que maximiza el score DBCV. Esta métrica para agrupamientos no supervisados resulta más adecuada que la métrica Silhouette para agrupamientos no globulares, ya que se basa en la densidad de elementos en el espacio. Se ilustran también los valores del score de Silhouette a fines comparativos.

Se utilizó la métrica de distancia coseno para el agrupamiento, de acuerdo a los hallazgos realizados en la sección anterior.

Los parámetros utilizados para DBScan fueron los siguientes:

- Epsilon: búsqueda en grilla desde el valor 0.001, hasta el valor 0.2, con un paso de 0.001 entre iteración.
- Cantidad mínima de elementos: 2
- Métrica de distancia: precomputada. (1 - similitud coseno entre pares de filtros)
- Algoritmo: Fuerza bruta

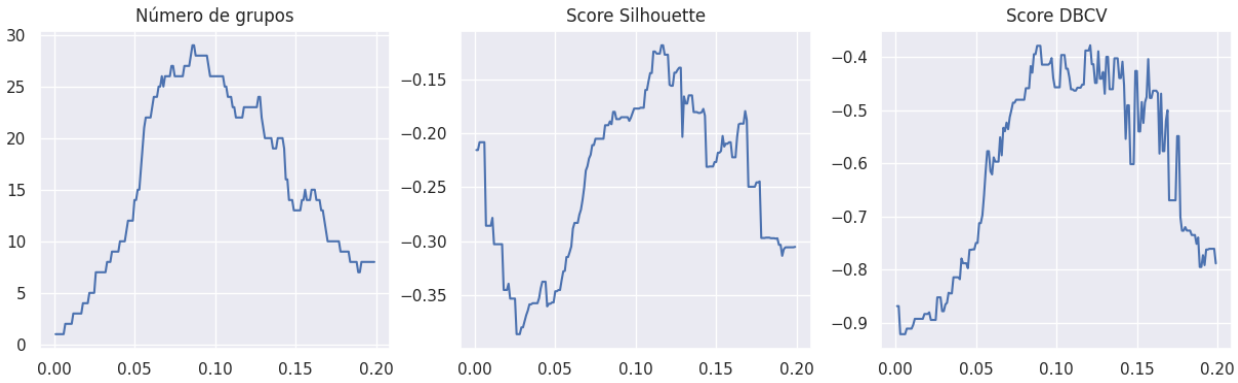


Fig 3.15: Gráfico de la izquierda: cantidad de grupos encontrados en función del valor de epsilon. Gráfico central: Score Silhouette en función del valore de epsilon. Gráfico de la derecha: Score DBCV en función del valor de epsilon.

El valor de epsilon encontrado que maximiza el score DBCV fue de 0.121, permitiendo encontrar un total de 23 grupos. Del total de 272 filtros, 121 de ellos fueron ubicados por DBScan como pertenecientes a un grupo. Los 151 restantes no han sido agrupados y permanecen como elementos aislados (no se visualizan a continuación).


Grupo 1
Nº 1 (efficientnetv2-l)
Nº 8 (efficientnetv2-l)
Nº 73 (MobilenetV3large)
Nº 80 (NASNet)
Nº 81 (NASNet)
Nº 85 (NASNet)
Nº 88 (NASNet)
Nº 95 (NASNet)
Nº 110 (NASNet)
Nº 111 (NASNet)
Nº 112 (NASNet)
Nº 116 (NASNet)
Nº 118 (NASNet)
Nº 131 (NASNet)
Nº 133 (NASNet)
Nº 145 (NASNet)
Nº 155 (NASNet)
Nº 160 (NASNet)
Nº 161 (NASNet)
Nº 170 (NASNet)
Nº 207 (vgg19)
Nº 248 (xception)
Grupo 2
Nº 4 (efficientnetv2-l)
Nº 69 (MobilenetV3large)
Grupo 3
Nº 5 (efficientnetv2-l)
Nº 206 (vgg19)
Nº 211 (vgg19)
Nº 213 (vgg19)
Nº 223 (vgg19)
Nº 229 (vgg19)
Nº 234 (vgg19)
Nº 236 (vgg19)
Grupo 4
Nº 20 (efficientnetv2-l)
Nº 83 (NASNet)
Grupo 5
Nº 23 (efficientnetv2-l)
Nº 41 (inception_v3)
Nº 44 (inception_v3)
Nº 64 (MobilenetV3large)
Nº 65 (MobilenetV3large)
Nº 98 (NASNet)
Nº 108 (NASNet)
Nº 119 (NASNet)
Nº 120 (NASNet)
Nº 129 (NASNet)
Nº 137 (NASNet)
Nº 140 (NASNet)
Nº 158 (NASNet)
Nº 166 (NASNet)
Nº 181 (vgg19)
Nº 184 (vgg19)
Nº 191 (vgg19)
Nº 192 (vgg19)
Nº 194 (vgg19)
Nº 195 (vgg19)
Nº 208 (vgg19)
Nº 215 (vgg19)
Nº 246 (xception)
Nº 259 (xception)
Grupo 6
Nº 25 (efficientnetv2-l)
Nº 139 (NASNet)
Grupo 7
Nº 27 (efficientnetv2-l)
Nº 84 (NASNet)
Nº 150 (NASNet)
Nº 252 (xception)
Grupo 8
Nº 29 (efficientnetv2-l)
Nº 57 (inception_v3)
Nº 91 (NASNet)
Nº 132 (NASNet)
Nº 159 (NASNet)
Nº 177 (vgg19)
Nº 198 (vgg19)
Nº 201 (vgg19)
Nº 212 (vgg19)
Nº 226 (vgg19)
Nº 249 (xception)
Grupo 9
Nº 30 (efficientnetv2-l)
Nº 104 (NASNet)
Nº 128 (NASNet)

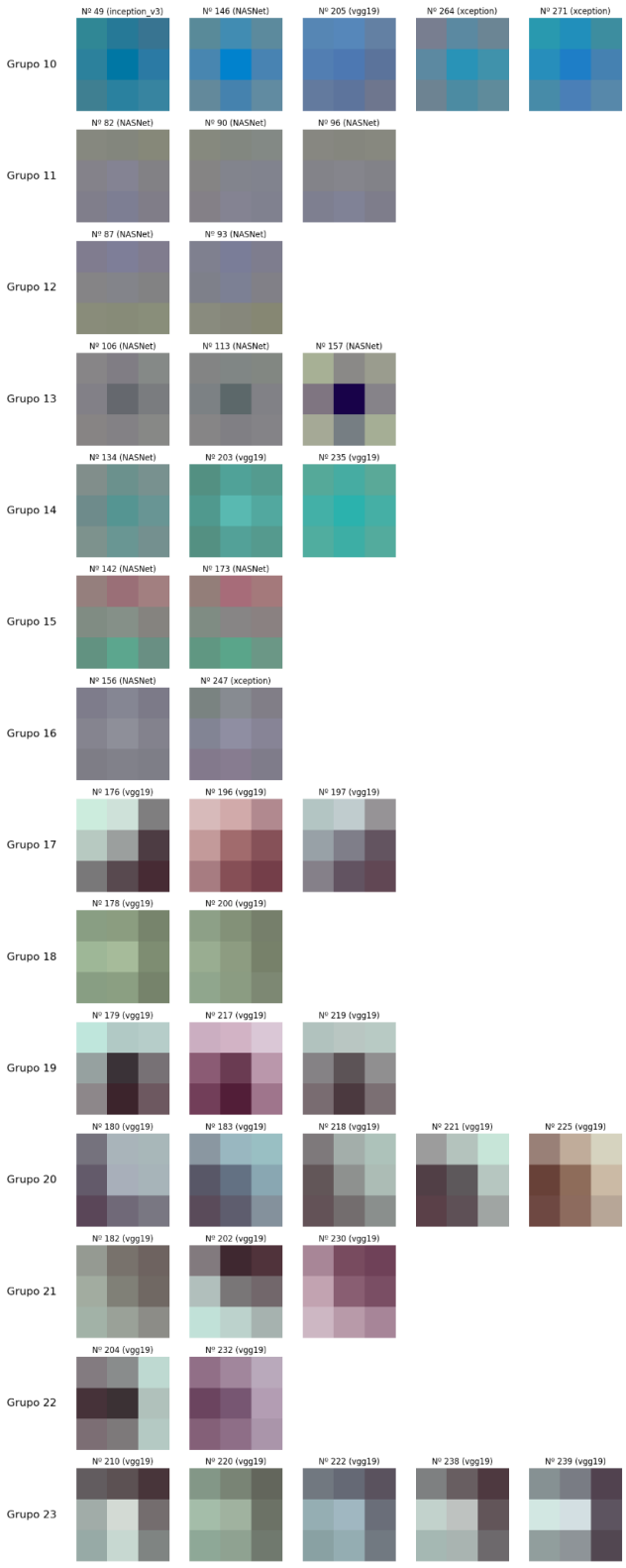


Fig 3.16: Grupos de filtros encontrados por medio del algoritmo DBScan.

3.2.2. Agrupamiento por medio de agrupamiento aglomerativo

Para el agrupamiento por medio de agrupamiento aglomerativo, se estableció el tipo de enlace en “completo” y por medio de búsqueda en grilla, se encontró el número óptimo de grupos que maximiza el score DBCV. Se ilustran también los valores del score de Silhouette para fines comparativos.

Se utilizó la métrica de distancia coseno para el agrupamiento, de acuerdo a los hallazgos realizados en la sección anterior.

Los parámetros utilizados para el algoritmo de agrupamiento aglomerativo fueron los siguientes:

- Cantidad de grupos: búsqueda en grilla desde el valor 2 hasta el valor 271, con un paso de 1 grupo adicional entre iteración.
- Tipo de enlace: completo
- Métrica de distancia: precomputada. (1 - similitud coseno entre pares de filtros)
- Umbral de distancia: No definido (en este algoritmo se debe definir o bien la cantidad de grupos o bien el umbral de distancia, pero no ambos, ya que será uno u otro criterio el utilizado para detener el algoritmo)

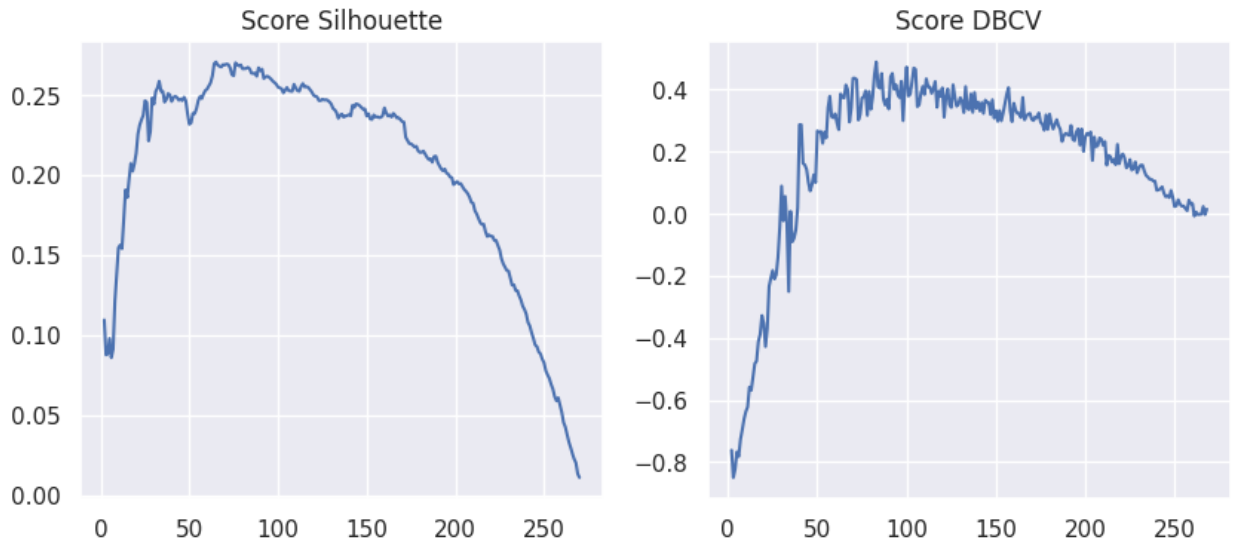
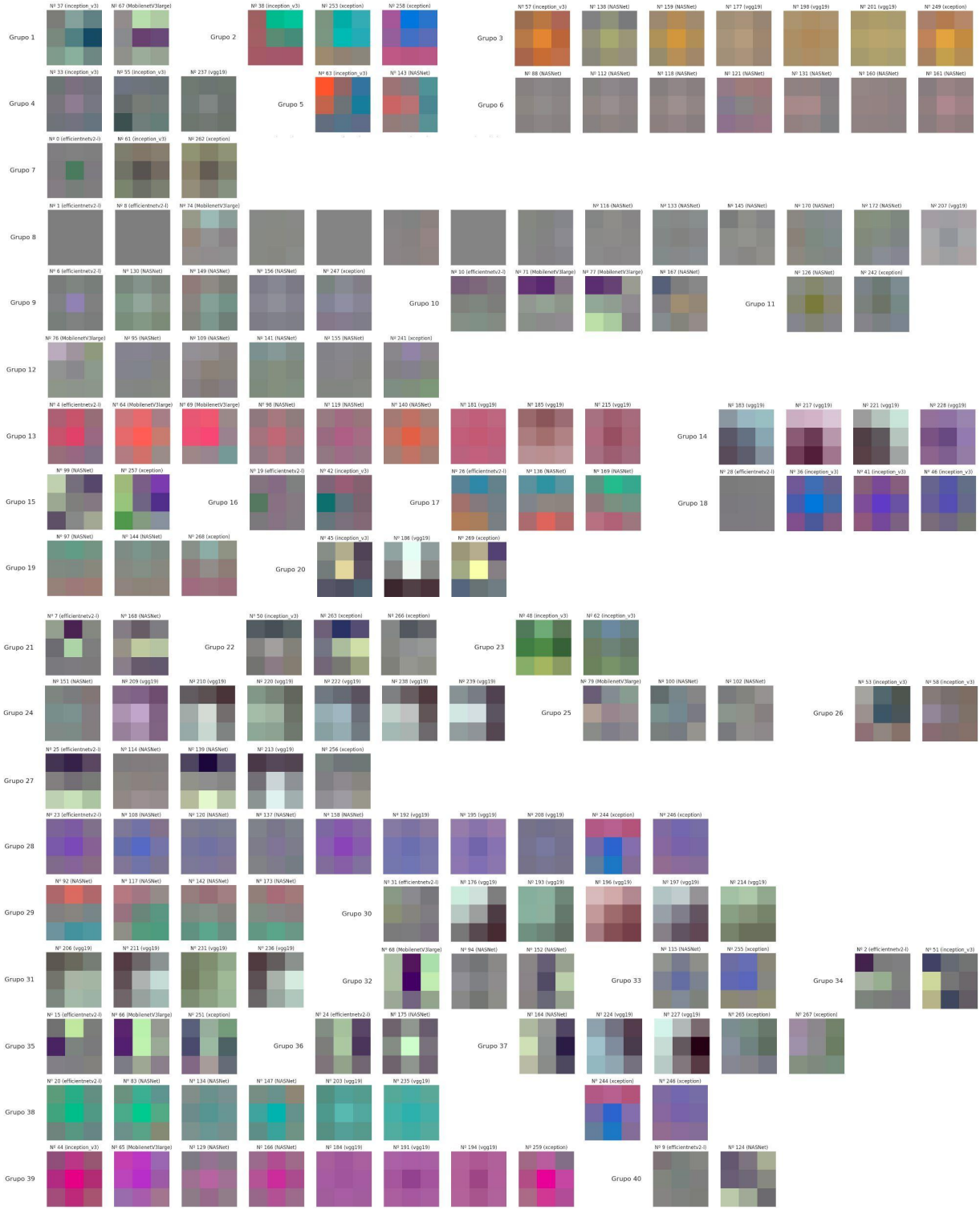


Fig 3.17: Gráfico de la izquierda: Score Silhouette en función del número de grupos encontrados por medio del algoritmo de agrupamiento aglomerativo. Gráfico de la derecha: Score DBCV en función del número de grupos encontrados por medio del algoritmo de agrupamiento aglomerativo.

El número de grupos encontrado que maximiza el score DBCV fue de 83, 66 de ellos con más de un elemento. Se visualizan a continuación solamente los grupos conformados por más de un filtro.



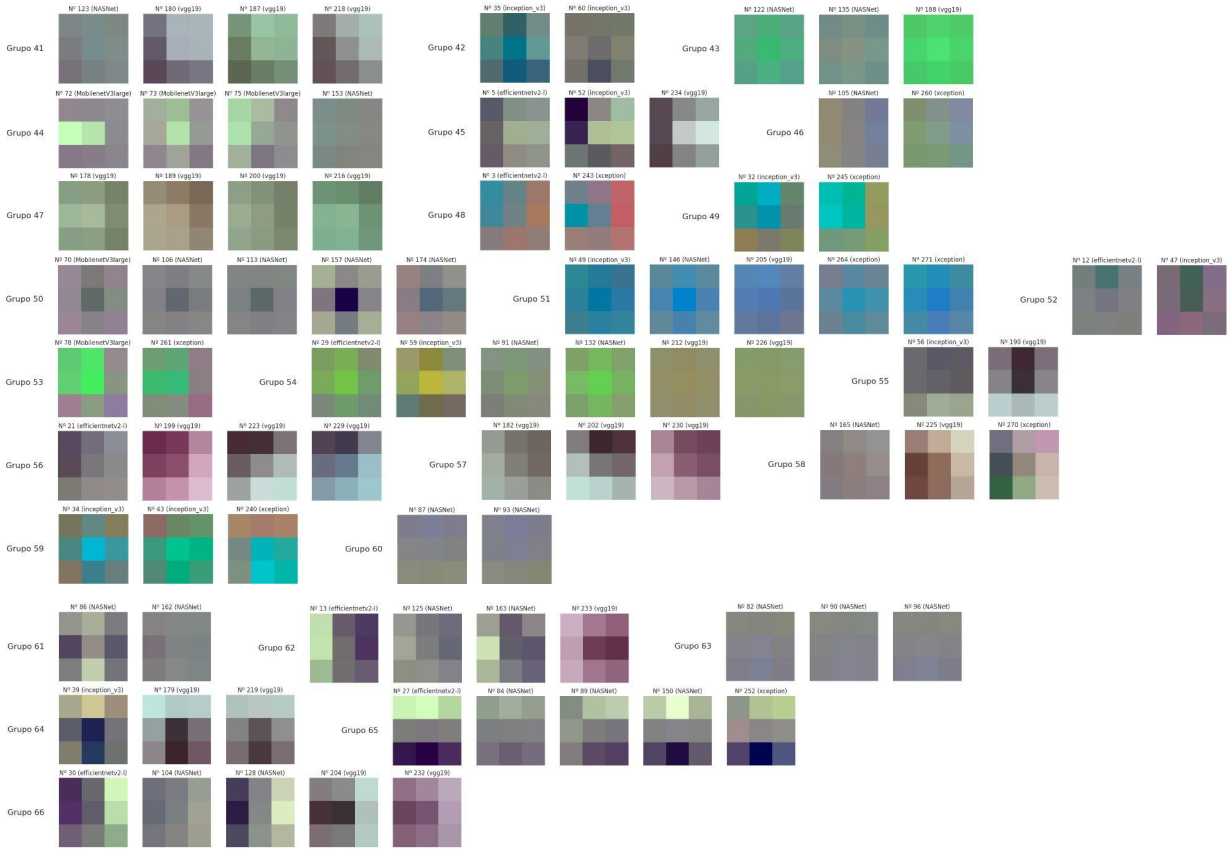


Fig 3.18: Grupos de filtros encontrados por medio del algoritmo de agrupamiento aglomerativo.

3.2.3. Observaciones y conclusiones de la sección

Ambos algoritmos de agrupamiento parecen encontrar grupos coherentes, pero a juzgar por el score DBCV (y también Silhouette), el agrupamiento aglomerativo resulta altamente superior. De cualquier modo, la visualización de los agrupamientos obtenidos por ambos algoritmos resulta complementaria y un buen apoyo para el análisis posterior.

En ambos agrupamientos se presentan uno o más grupos conformados en su mayoría por filtros muertos y/o filtros con parámetros muy cercanos a cero. Es el caso del grupo 1 para DBScan y grupo 8 para el agrupamiento aglomerativo.

Para ambos agrupamientos, cada grupo parece definir tanto una estructura determinada en cuanto a la disposición de los 9 píxeles como un valor tonal preponderante. Existe cierta variabilidad interna al grupo, tanto en cuanto a valor tonal como a estructura, pero en la

mayoría de los casos parece existir buena cohesión. También se percibe la preponderancia de uno de los dos componentes (estructura o valor tonal) en mayor o menor medida en cada grupo.

Más allá de la presencia de filtros muertos como muchos de los presentes en el grupo 1 (DBScan) y grupo 8 (aglomerativo), se puede apreciar la presencia de filtros extremadamente similares en varios grupos pertenecientes a un mismo modelo. Esto avalaría la hipótesis de la generación de filtros redundantes producto del entrenamiento por medio de GDBP. Este hecho es particularmente notorio en los grupos 11, 12 y 15 (DBScan), y 29, 31, 60 y 63 (aglomerativo) donde un par o más de sus filtros resultan extremadamente similares y provienen de un mismo modelo (NasNET y VGG19).

Asimismo, resultan particularmente interesantes los grupos 4, 6 y 16 (DBScan) y 9, 13, 27 y 28 (aglomerativo) donde se puede percibir con claridad un segundo fenómeno, complementario del primero. Esto es, la presencia de filtros extremadamente similares entre sí, pertenecientes, en este caso, a diferentes modelos. Este fenómeno avalaría por su parte, la segunda hipótesis planteada: diferentes modelos, con diferentes arquitecturas y entrenados bajo diferentes hiperparámetros arriban a soluciones con algunos de sus parámetros (filtros) coincidentes. Este hecho no solo es observable en los grupos mencionados, sino que también puede percibirse en menor medida en el resto de los grupos y aportaría evidencias adicionales en favor de la restricción en el espacio de soluciones arribadas.

Si se considera la dimensionalidad (d=27: 9 píxeles y 3 canales por píxel) de los filtros, la probabilidad de obtener 2 elementos extremadamente similares a partir de un conjunto de filtros relativamente pequeño (272 filtros), es realmente muy remota.

Es decir, podría afirmarse que las soluciones a las que arriban diferentes modelos, con diferentes arquitecturas y entrenados bajo diferentes hiperparametros no son independientes.

Para dar una idea de la improbabilidad de encontrar un par de filtros de dimensionalidad 27 dentro de un grupo de 272 filtros con similitud coseno igual o superior a la de los pares mencionados, se realizó el siguiente muestreo.

Se tomaron 100.000 muestras, cada una de ellas de conjuntos de 272 filtros de dimensionalidad 27. El muestreo se realizó con distribución uniforme para el rango [-1, 1] y para cada una de estas muestras se seleccionó el par de filtros con mayor similitud coseno. En el histograma se ilustran los valores de similitud coseno máximo para cada una de las 100.000 muestras.

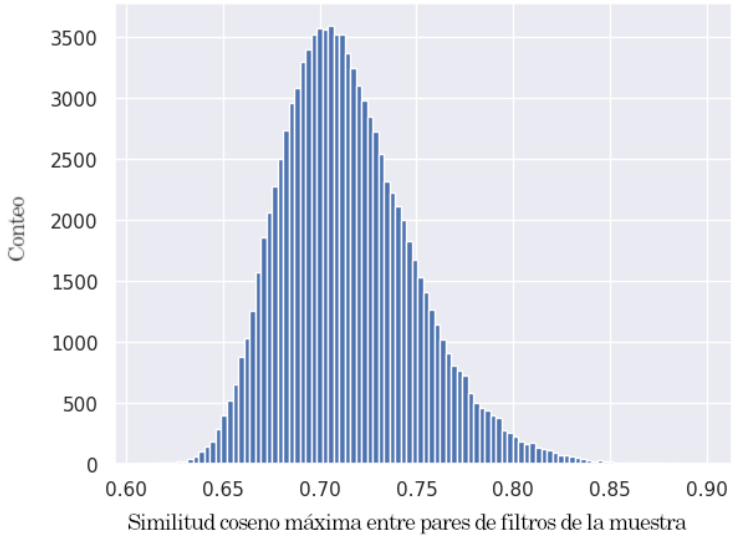


Fig 3.19: Histograma para el valor máximo de similitud coseno entre pares de filtros para cada muestra.

Según se aprecia en el histograma, y bajo la hipótesis de independencia, la probabilidad de obtener pares de filtros de 27 dimensiones, a partir de un conjunto de 272 filtros, con una similitud coseno mayor a 0.85 es prácticamente nula.

Se muestran a continuación, algunos pares de filtros con alta similitud coseno (mucho más allá de 0.85), pertenecientes a diferentes modelos.

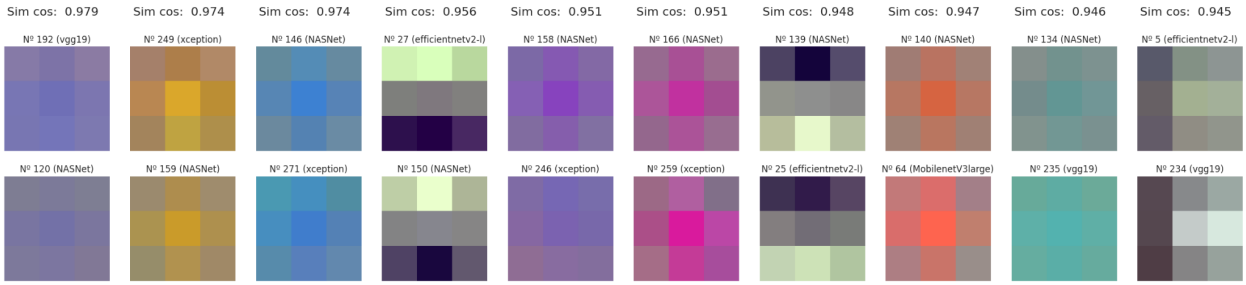


Fig 3.20: Pares de filtros pertenecientes a diferentes modelos con alta similitud coseno.

Se puede afirmar entonces, que diferentes modelos arriban a soluciones que contienen uno o más filtros muy similares. Siendo muy remotas las probabilidades de que esto ocurra bajo la hipótesis de independencia, debería refutarse dicha hipótesis.

Probablemente, el punto a dilucidar será entonces, de qué variable o variables son igualmente dependendientes los distintos modelos.

4. Métodos de obtención de filtros para capas convolucionales

Para el presente trabajo se proponen varios métodos no supervisados para el entrenamiento de filtros para capas convolucionales. Es decir, sin la necesidad de contar con conjuntos de imágenes etiquetadas. El procedimiento seguido para cada uno de estos métodos es diferente en su implementación pero se basa en algunos criterios básicos comunes y algunas consideraciones.

Uno de estos criterios es el de basar el entrenamiento de filtros en la distribución de los datos de entrada. Con datos de entrada aquí se hace referencia no sólo al conjunto de imágenes sobre el que se entrenan los modelos, sino también a los mapas de características producidos por cada una de las capas convolucionales. Es decir, cada capa es entrenada de forma independiente, utilizando como conjunto de entrenamiento la salida producida por la capa inmediata anterior mediante el proceso de inferencia (*forward pass)*. Para el caso de la primera capa, los datos para el entrenamiento son directamente el conjunto de imágenes.

En todos los métodos de entrenamiento el conjunto de datos es preprocesado para obtener un conjunto de parches pertenecientes a las imágenes originales, en el caso de la primera capa, y pertenecientes a los mapas de características, en el caso de capas subsiguientes. Son estos parches con dimensionalidad idéntica a la de los filtros que se desean construir, lo que se utilizan para los diversos métodos de entrenamiento.

El criterio subyacente a todos los métodos ensayados es el de utilizar la distribución de los datos para el entrenamiento de los filtros. Esto se traduce en búsquedas de criterios de separación de los datos que resulten significativas para obtener una buena condensación de la información visual.

Respecto a las consideraciones que motivaron la búsqueda de un algoritmo alternativo de entrenamiento para redes convolucionales, está la intuición de que las soluciones encontradas por medio del entrenamiento tradicional basado en BP y GD son subóptimas.

Esta intuición está fuertemente respaldada por diferentes trabajos, entre los cuales destaca el de Imamura et al [46]. En este trabajo, los autores proponen el empleo de filtros Gabor para las primeras dos capas convolucionales, aprendiendo los parámetros de dichos filtros por medio de GDBP, pero restringiendo las soluciones a este tipo de filtros. Por medio de esta técnica logran reducir la cantidad de filtros utilizados para la primera capa de una arquitectura VGG-16 en un 83% y en un 94% para la segunda. Estas reducciones no provocan un impacto significativo en el desempeño de la red.

Adicionalmente y como fruto de la exploración realizada en el capítulo previo, se puede observar para la primera capa de modelos con alto desempeño, que los filtros encontrados por GDBP incluyen filtros muertos (filtros planos) y filtros redundantes. Este hecho no es aislado u ocasional sino que constituye la regla a lo largo de varios modelos diferentes, tal como se ha podido verificar en el capítulo anterior.

Otra de las consideraciones relevantes para el entrenamiento de filtros convolucionales es el hecho de que un filtro determinado obtendrá su máxima activación frente a un parche idéntico a sí mismo, bajo la condición de que ambos estén previamente normalizados por su norma euclidiana. Es solo en estas circunstancias que la máxima activación se obtiene solo con un parche de imagen idéntico al filtro. En definitiva, una vez normalizados ambos, la convolución resulta ser una mero cálculo de similitud coseno. Esto es debido a que la operación de convolución al nivel de interacción entre parche y filtro no es más que un producto punto en el caso de que ambos estén previamente normalizados.

Este último punto resulta evidente al vectorizar tanto el parche como el filtro. Una vez transformados en vectores, la operación de producto punto, es decir, la suma del producto elemento a elemento de ambos vectores es completamente equivalente a la convolución. Por otra parte, si se considera que el coseno del ańgulo formado por ambos vectores es equivalente al producto punto normalizado, queda claro el punto expuesto anteriormente: un filtro obtendrá su máxima activación frente a un parche idéntico bajo la condición de que ambos estén previamente normalizados.

Para completar la intuición acerca del comportamiento de la operación de convolución y la relación entre similitud entre filtro y parche y el valor de activación obtenido, se muestran a continuación la salidas obtenidas para un mismo filtro y diferentes parches de imagen:

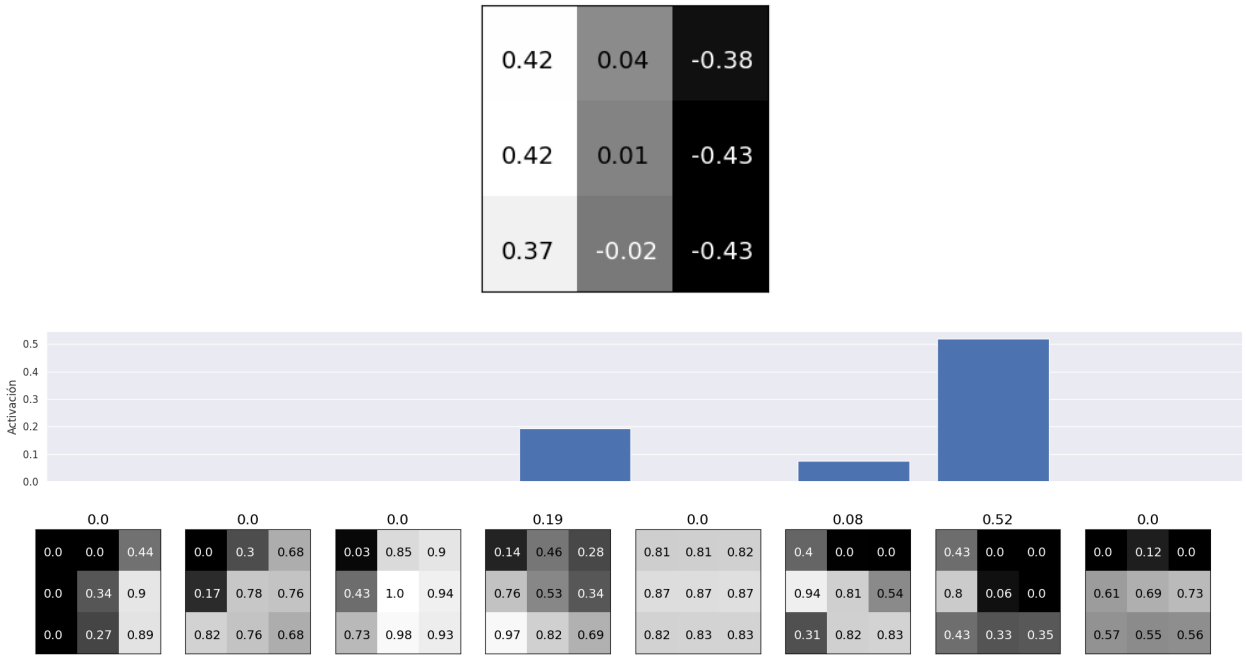


Fig 4.1: Arriba: filtro convolucional de 3x3 normalizado. Fila media: valores de activación para cada uno de los parches de la fila inferior al ser convolucionados con el filtro superior. Fila inferior: parches (no normalizados).

Si bien los filtros generados por medio de los diferentes métodos aquí presentados se normalizan de uno u otro modo, los parches de las imágenes del conjunto de entrada no sufren ninguna normalización a nivel parche. Por consiguiente, no se cumple la condición bajo la cual se obtiene la máxima activación en condiciones de máxima similitud coseno entre filtro y parche para reducir la operación de convolución a la de producto punto (a nivel parche). Bajo esta circunstancia, las máximas activaciones para determinado filtro se obtendrán con parches con escalares máximos para aquellas dimensiones positivas del filtro y con escalares mínimos para dimensiones negativas del filtro. Es decir, desde el punto de vista de la activación de determinado filtro para todo el conjunto de parches, las graduaciones tonales del filtro no son relevantes para maximizar su activación.

Otra es la circunstancia cuando se toma la perspectiva inversa, es decir, frente a determinado parche de imagen, su convolución con los diferentes filtros producirá activaciones diferenciadas de acuerdo a la graduación tonal -el valor escalar- presente en cada dimensión de cada filtro. Es decir, dos filtros con las mismas dimensiones con escalares positivos y las mismas dimensiones con escalares negativos, pero diferentes, no producirán el mismo escalar de activación al realizar la convolución con un mismo parche de imagen. Trasladado al nivel de imagen completa, esto significa que estos dos filtros producirán diferentes mapas de características frente a una misma imagen de entrada, aun con idénticos signos para los escalares de todas sus dimensiones.

Llevado al terreno de la hipótesis planteada anteriormente respecto a la relación entre similitud filtro-parche y su nivel de activación, al generar los filtros a partir de la distribución del conjunto de parches de entrada, se busca generar un conjunto de mapas de características representativo de las imágenes de entrada. Para determinado parche de imagen y considerando filtros normalizados con media de cero, ahora sí, el filtro que mayor activación dará a dicho parche de imagen será aquel con el que mayor similitud tenga.

En la sección 4.1, 4.2 y 4.3 del presente capítulo se desarrollan las metodologías empleadas para el entrenamiento de filtros para capas convolucionales, detallando las etapas de pre-procesamiento, entrenamiento y post-procesamiento para cada uno de los métodos: Componentes Principales (CP), K-Medias (KM) y parches aleatorios.

4.1. Método de obtención de filtros convolucionales a través de Componentes Principales (CP)

En esta sección se abordará el proceso de entrenamiento de filtros convolucionales por medio del método basado en componentes principales (PCFG, por sus siglas en inglés, *Principal Components based Filter Generation*).

El método consiste en utilizar las imágenes de entrada de una capa convolucional para obtener sus parches y determinar las direcciones donde estos parches tienen máxima variabilidad. Un parche es simplemente un recorte de determinada cantidad de píxeles de ancho y alto de una imagen. A partir de determinada imagen de entrada, se recortan todos los posibles parches realizando un barrido por toda la imagen de izquierda a derecha y de arriba a abajo.

Cada parche puede concebirse como un conjunto de valores que bien pueden disponerse en forma vectorial. En los experimentos realizados se han utilizado parches de 3 píxeles de ancho por 3 píxeles de alto. Si consideramos imágenes en color con 3 canales (RGB), cada parche obtenido a partir de la imagen de entrada tendrá dimensiones 3x3x3. Representado en forma vectorial esto es un vector de 27 dimensiones. A partir de todos los parches obtenidos de todas las imágenes de entrada, se obtiene un conjunto de datos con $\boldsymbol{n}$ puntos (tantos como parches se hayan obtenido) de dimensionalidad $\boldsymbol{d}$ (27 en este caso). Utilizando este conjunto de datos, se calculan las direcciones donde su variabilidad es mayor. Se utilizan luego estas direcciones para construir los filtros convolucionales. La intuición aquí es que estos filtros podrán realizar una buena separación de los datos de entrada, al estar construidos específicamente para operar sobre las direcciones de mayor variabilidad de los datos.

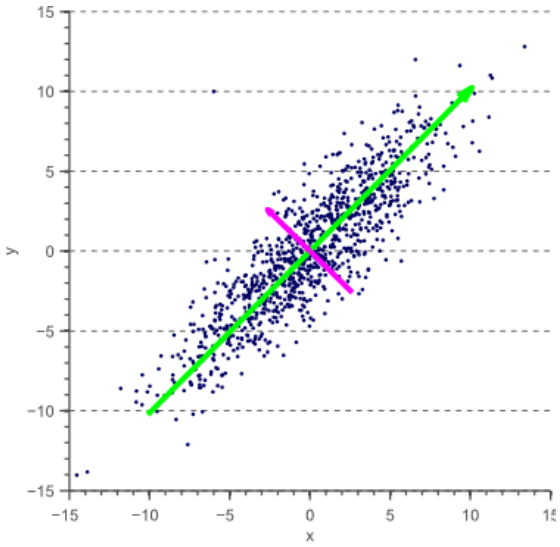


Fig 4.2: Gráfico bidimensional con la distribución de observaciones para 2 dimensiones. En verde y magenta se observan las direcciones de máxima variabilidad de los datos. La longitud de los segmentos es proporcional a la variabilidad de dicha dirección.

La hipótesis que guía esta metodología es la de encontrar las dimensiones de máxima variabilidad en los datos de entrada, de modo tal de facilitar la condensación de la información que se pasará a capas posteriores de la red. Utilizando filtros construidos a partir de los autovectores de la matriz de covarianza se busca generar activaciones, frente a los datos de entrada, que permitan discriminar los parches en las dimensiones del espacio de mayor variabilidad.

Se detallan a continuación, los pasos de preprocesamiento, entrenamiento y post procesamiento de los datos de entrada para la generación de un conjunto de filtros mediante PCFG.

4.1.1. Preprocesamiento

1. **Lectura de las imágenes del conjunto de datos.** Se carga en un tensor de dimensiones N, W, H, C el conjunto de las imágenes de entrada.

2. **Obtención de una muestra representativa del tensor de datos**. A partir del tensor de datos del conjunto de entrada, se obtiene una muestra reducida de N imágenes. En cuanto a la representatividad de la muestra, resulta conveniente realizar la extracción de forma estratificada para los problemas de clasificación, con el fin de capturar la variabilidad inherente del conjunto de datos. Para el caso de los dos conjuntos de datos

utilizados en los experimentos (CIFAR10 y Mnist-Fashion) donde la distribución de clases es balanceada, basta tomar igual cantidad de elementos de cada clase.

3. **Generación del tensor de parches de imagen**. A partir del tensor con la muestra de imágenes se genera el tensor de parches de imagen. Los parches de imagen deben tener el mismo ancho y alto que los filtros convolucionales que se desean generar. Para el caso de los experimentos realizados en este trabajo se generaron filtros de 3 píxeles de ancho (Kw = 3) por 3 píxeles de alto (Kh = 3). La cantidad de canales de los parches debe ser idéntica a la del conjunto de datos de entrada. Para CIFAR10, esto es C=3 y para Mnist-Fashion C=1. Para realizar la extracción de los parches se itera con dos bucles anidados por todo el área de la imágen, de izquierda a derecha y de arriba a abajo, con incrementos igual a la zancada de filtro que se desee realizar. Para todos los experimentos se generaron parches con zancada S = 1. Es decir, se extraen todos los posibles parches de 3 píxeles de ancho por 3 píxeles de alto de cada imagen.

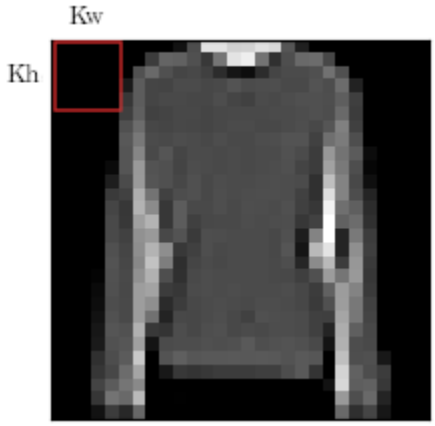


Fig 4.3: A partir de la imagen completa, se extraen todos los parches posibles de Kw ancho y Kh alto. En rojo se ilustra el primer parche a extraer de la imagen. Para extraer cada uno de los parches posibles de una imagen se recorre la imagen de izquierda a derecha y de arriba a abajo realizando zancadas de 1 pixel.

Finalmente, se concatenan todos los parches de modo tal de obtener un único tensor de dimensiones [Kn cantidad de parches, Kw ancho del filtro, Kh alto del filtro, C cantidad de canales]. Donde:

$$Kn = N * (W - Kw + 1) * (H - Kh + 1)$$

Desde un punto de vista computacional, es recomendable realizar la extracción de los parches de forma vectorial para abarcar el total de las imágenes de la muestra en cada iteración.

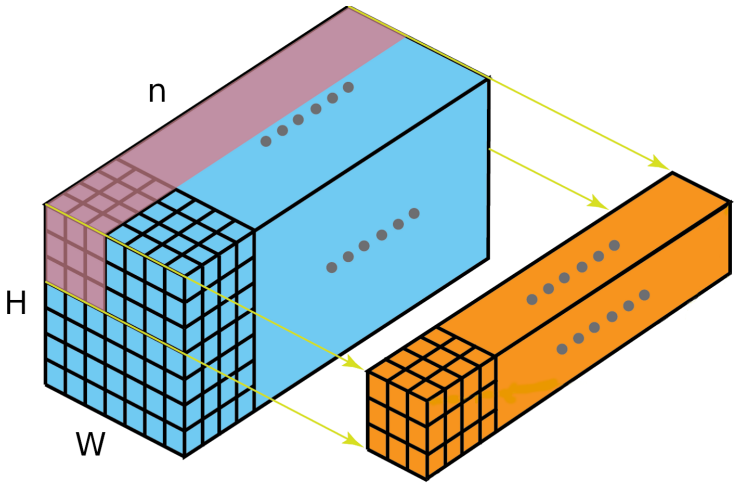


Fig 4.4: Extracción de un parche de imagen para todo el tensor de imágenes en una sola operación.

En la figura 4.4 se ilustra este concepto, sin incluir la dimensión correspondiente a los canales C (que implicaría una ilustración imposible en 4D).

4. **Redimensionado del tensor de parches**. Este paso consta simplemente del redimensionado del tensor de parches, colapsando en una sola las dimensiones Kw de ancho filtro, Kh de alto de filtro y C de canales. Es decir:

$$[Kn, Kw, Kh, C] \text{ —> } [Kn, (Kw * Kh * C)]$$

5. **Estandarización del tensor de parches a nivel parche**. El último paso de la etapa de preprocesamiento consiste en el estandarizado del tensor, a nivel parche. Es decir, para cada parche, de forma independiente, restar su valor medio y dividir por su desvío estándar.

   El objetivo de la estandarización a nivel parche es doble: por un lado, poder generar filtros con capacidades de activación similares. Sin la estandarización, los diferentes filtros que se generarán a partir de los parches no tendrán una misma capacidad de activación, sesgando las salidas de la capa hacia aquellos parches con escalares mayores. Por otra parte, la estandarización de los parches permite centrar en cero el nivel de activación de los filtros que se construirán a partir de ellos. De este modo, los filtros presentarán un rango de salida equilibrado entre valores positivos y negativos.

4.1.2. Entrenamiento de los filtros

1. **Estandarización del tensor de parches a nivel característica.** Se toma como punto de partida el tensor generado en la etapa de preprocesamiento. Dicho tensor consta de 2 dimensiones, la primera de rango igual a la cantidad de parches extraídos ***P*** (equivalente a la cantidad de imágenes ***N*** * la cantidad de parches por imagen ***p***), y la segunda con dimensión igual a la cantidad de características de cada instancia (***Kw*** * ***Kh*** * ***C***). Para cada una de estas características, se calcula su media y desvío estándar y se procede a estandarizarlas, es decir, sustraer la media y dividir por el desvío estándar.

2. **Obtención de la matriz de covarianza**. La matriz de covarianza es una matriz cuadrada con tantas filas y columnas como características tenga el tensor de parches, es decir, el rango de su segunda dimensión (***Kw*** * ***Kh*** * ***C***). Cada entrada de esta matriz será la covarianza entre la característica ***i*** y la característica ***j***, siendo ***i*** el número de columna y ***j*** el número de fila de la matriz.

3. **Obtención de los autovectores de la matriz de covarianza.** Siendo ***A*** la matriz de covarianza calculada en el paso anterior, ***v*** un autovector y λ el autovalor asociado a ***v***, se calculan todos los autovectores con sus autovalores. Para este caso particular: (***Kw*** * ***Kh*** * ***C***).

   Se ordenan de mayor a menor los autovalores con sus correspondientes autovectores y se conservan los primeros ***Kn*** autovectores, tantos como filtros convolucionales se desea construir. Este método de entrenamiento tiene la limitante de poder generar solo hasta (***Kw*** * ***Kh*** * ***C***) filtros. Esto es así debido a las características propias del algoritmo de Componentes Principales. La cantidad de autovectores y autovalores que se obtienen es siempre igual a la cantidad de dimensiones del conjunto de datos de entrada. Si bien no es posible generar una cantidad de filtros mayor a (***Kw*** * ***Kh*** * ***C***), es posible seleccionar un subconjunto de autovectores menor a esta cantidad. Para ello se pueden utilizar los correspondientes autovalores para realizar la selección.

4.1.3. Post Procesamiento

Redimensionado y concatenado del conjunto de autovectores. Cada uno de los autovectores generados en el paso anterior tendrá una dimensión (Kw * Kh * C). El paso final es descomponer estos vectores en tensores de dimensionalidad [Kw, Kh, C]. Una vez hecho esto solo resta concatenar todos los tensores en una nueva dimensión. Se obtiene así, un único tensor de dimensión [Kw, Kh, C, Kn]. Este tensor es directamente aplicable como conjunto de pesos para una capa convolucional de Kn filtros de tamaño Kw x Kh, con C canales de entrada. Los sesgos de la capa son fijados en cero.

Fig 4.5: Fila superior: los 27 filtros de 3x3 obtenidos por medio de CP con una muestra de 100 imágenes del conjunto de datos CIFAR10. Fila inferior: los primeros 32 filtros (con mayores autovalores) de 7x7 obtenidos por medio de CP con una muestra de 100 imágenes del conjunto CIFAR10.

Tal como se menciona en el capítulo 2 sección 2.6.1, existen varios trabajos precedentes como el Chan et al [25] y Ren et al [26] que utilizan el algoritmo de Componentes Principales para el entrenamiento de filtros para capas convolucionales o como método de inicialización de los pesos. La principal diferencia entre estos trabajos mencionados y el presente radica en el objetivo final que persigue este último: la aplicación sistemática del algoritmo para el entrenamiento de filtros para todas las capas de una red profunda de arquitectura actual. Si bien la presente tesis explora en profundidad la aplicación de este y otros algoritmos en el entrenamiento de la primera capa convolucional solamente, plantea las bases para extrapolar el procedimiento a capas subsiguientes en futuros trabajos. Adicionalmente, y en contraposición con los trabajos previamente mencionados, donde el énfasis está puesto en la reducción de recursos computacionales, este trabajo busca obtener resultados comparables o superiores al entrenamiento por GDBP.

4.2. Método de obtención de filtros convolucionales a través de K-Medias (KM)

En esta sección se abordará el proceso de entrenamiento de filtros convolucionales por medio del método basado en K-Medias.

Esta metodología de entrenamiento de filtros está basada en dos premisas hipotéticas. Por un lado, la suposición de que existen regiones del espacio de parches más densamente pobladas que otras, e incluso regiones completamente libres de parches. Es decir, se presume la existencia de conglomerados de parches o modas, en contraposición de una distribución relativamente uniforme en el espacio de los parches.

La segunda de las hipótesis consiste en la suposición de que los centroides de estos agrupamientos o conglomerados de parches, al ser utilizados como filtros convolucionales, permiten capturar de forma eficiente la mayor parte de la información contenida.

La propuesta con K-Medias es entonces encontrar n regiones y sus centroides, siendo n la cantidad de filtros a generar, que mejor cobertura puedan obtener de la distribución de los parches de entrada en el espacio d dimensional. Estos centroides son, idealmente, los que mayor cantidad de parches pueden capturar a partir de sus activaciones, generando una representación lo más completa posible de la distribución de los parches de entrada en los mapas de características de salida.

Adicionalmente, se plantea como alternativa el empleo de K-Medoides, es decir, utilizando la mediana en lugar de la media del cúmulo como filtro. Este algoritmo se presenta como alternativa de mayor robustez frente a la presencia de eventuales parches aislados o *outliers*.

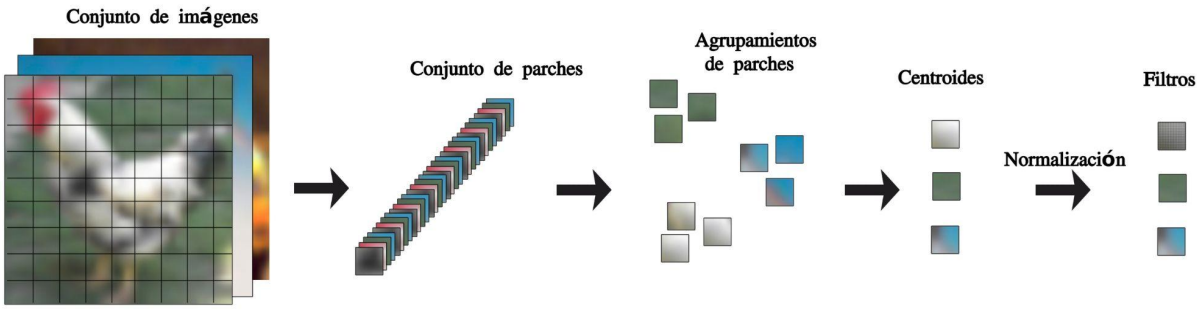


Fig 4.6: Proceso de entrenamiento de filtros por medio de K-Medias (KM). A partir del conjunto de imágenes se extraen los parches y luego se realiza el agrupamiento de los mismos. Los centroides obtenidos, una vez normalizados, son aplicados como filtros en la red convolucional.

4.2.1. Preprocesamiento

Esta etapa es idéntica a la descrita en 4.1.1 para el caso de entrenamiento de filtros mediante componentes principales (CP).

4.2.2. Entrenamiento de los filtros

Búsqueda de los centroides por medio de K-Medias. Se toma como punto de partida el tensor generado en la etapa de preprocesamiento Dicho tensor consta de 2 dimensiones, la primera de rango igual a la cantidad de parches extraídos, y la segunda con rango igual a la cantidad de características de cada instancia (Kw * Kh * C). Se realiza una ejecución del algoritmo K-Medias utilizando el tensor de parches como datos de entrada, seleccionando K =

Kn, es decir, tantos grupos como filtros se deseen generar. Como salida del algoritmo se obtienen K centroides de dichos agrupamientos. Cada uno de ellos será un vector de (Kw * Kh * C) dimensiones.

4.2.3. Post Procesamiento

Redimensionado del conjunto de centroides. Cada uno de los centroides o medoides generados en el paso anterior tendrá una dimensión (Kw * Kh * C). El paso final es redimiensionar estos vectores en tensores de dimensionalidad [Kw, Kh, C]. Los sesgos de la capa son fijados en cero.

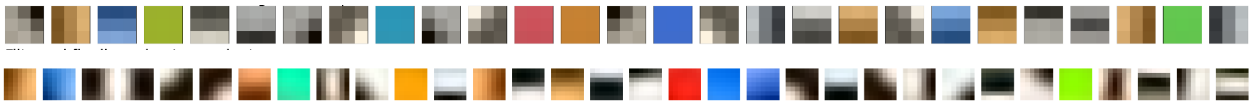


Fig 4.7: Fila superior: 27 filtros de 3x3 obtenidos por medio de k-Medias (KM) con una muestra de 100 imágenes del conjunto de datos CIFAR10. Fila inferior: 32 filtros de 7x7 obtenidos a través de k-Medias (KM) con una muestra de 100 imágenes del conjunto CIFAR10.

4.3. Método de obtención de filtros convolucionales a través de parches aleatorios

Este método de obtención de filtros convolucionales se plantea fundamentalmente como punto de comparación con los otros dos métodos de entrenamiento. Generando filtros a partir de una muestra aleatoria de parches se pretende establecer la validez o refutar las hipótesis que subyacen detrás de los métodos de obtención de filtros por medio de componentes principales (CP) y K-Medias (KM).

El procedimiento para el entrenamiento de filtros por medio de parches aleatorios es el siguiente.

4.3.1. Preprocesamiento

Esta etapa es idéntica a la descrita en 4.1.1 para el caso de entrenamiento de filtros mediante componentes principales (CP).

4.3.2. Entrenamiento de los filtros

Muestreo aleatorio sobre el tensor de parches. Se toma como punto de partida el tensor generado en la etapa de preprocesamiento. Dicho tensor consta de 2 dimensiones, la primera de rango igual a la cantidad de parches extraídos y la segunda con rango igual a la cantidad de características de cada instancia. Se seleccionan *Kn* elementos de forma aleatoria, tantos como filtros se desea obtener, y se construye un nuevo tensor idéntico al anterior pero conteniendo solamente los elementos seleccionados.

4.3.3. Post Procesamiento

Redimensionado del tensor. El paso final consiste en descomponer la segunda dimensión del vector en 3 componentes: Kw, Kh, C, de modo tal que el tensor resultante tendrá dimensionalidad [Kn, Kw, Kh, C]. Los sesgos de la capa son fijados en cero.

Fig 4.8: Fila superior: 27 filtros de 3x3 obtenidos por medio de un muestreo aleatorio de parches a partir de una muestra de 100 imágenes del conjunto de datos CIFAR10. Fila inferior: 32 filtros de 7x7 obtenidos por medio de un muestreo aleatorio de parches a partir de una muestra de 100 imágenes del conjunto de datos CIFAR10.

En este capítulo se presentaron los métodos de entrenamiento de filtros convolucionales utilizados en los experimentos, detallando los procedimientos empleados en su implementación. En todos los casos se trata de métodos basados en la distribución de los datos de entrada sin el empleo de las anotaciones del conjunto de imágenes. Es decir, se trata en todos los casos de procedimientos de entrenamiento no supervisado, donde los parámetros de los filtros convolucionales son calculados exclusivamente a partir de las imágenes.

5. Evaluación de los métodos

En este capítulo se presentan los experimentos realizados con cada uno de los métodos de entrenamiento de filtros para capas convolucionales desarrollados en el capítulo precedente. El objetivo de estos experimentos es el de poder validar las hipótesis planteadas a lo largo de este trabajo y ganar en comprensión respecto al funcionamiento de las redes convolucionales. Específicamente, se pretende estimar la utilidad de los métodos alternativos de entrenamiento propuestos y validar la hipótesis subyacente. Es decir, la relevancia de la distribución de los datos de entrada para la construcción de una primera capa de filtros convolucionales.

En la sección 5.1 se detallan los objetivos de la experimentación y en las secciones 5.2 y 5.3 se desarrollan los criterios empleados en la realización de los experimentos y los criterios con los cuales fueron evaluados los resultados.

En la sección 5.4 se presentan los resultados de los entrenamientos en conjunto con un resumen detallado de los desempeños alcanzados por ambos métodos de entrenamiento, tomando como referencia el entrenamiento por medio de GDBP.

Los filtros generados para cada una de las metodologías propuestas son presentados en la sección 5.5, mostrando su estado inicial y su estado final, luego del entrenamiento por GDBP. Observar y medir los cambios producidos entre ambos estados permite profundizar en la comprensión del proceso de entrenamiento mediante GDBP así como evaluar la eficacia del método de inicialización.

Por su parte, la sección 5.6 desarrolla los resultados obtenidos de la aplicación de una variante del algoritmo de K-Medias, K-Medoides y su impacto en el entrenamiento de filtros convolucionales.

Finalmente, en la sección 5.7 se presentan los resultados y análisis de diversos experimentos confirmatorios. Estos experimentos han tenido la finalidad de profundizar en la comprensión del impacto del conjunto de datos en la calidad de los filtros obtenidos. Este punto resulta de particular interés dado que los métodos de entrenamiento aquí propuestos se basan en la distribución de los datos de entrada.

Los experimentos confirmatorios han sido los siguientes:

1. Impacto del tamaño muestral.
2. Impacto del desbalance de clases
3. Impacto del conjunto de imágenes seleccionado

5.1. Objetivos

Se plantearon varios objetivos para los experimentos llevados a cabo sobre diversos métodos de entrenamiento de filtros basados en la distribución de los datos de entrada.

El **primero** y más importante de ellos fue el de validar si el modelo puede obtener un conjunto de filtros útiles para el resto de la red neuronal, en función de la tarea de clasificación de imágenes, y de forma no supervisada y sin GDBP.

Más allá de la eficiencia que puedan presentar las diversas y variadas arquitecturas de redes convolucionales existentes en la literatura, son los parámetros de los modelos los que permiten obtener altos desempeños en la tarea de clasificación. Generar parámetros para una primera capa convolucional que sean útiles y coadyuven al resto de la red en la separación de clases, es entonces el objetivo principal de los experimentos. No se trata de una tarea trivial, como bien queda demostrado en los resultados de los experimentos, contrastando con el entrenamiento de filtros de modo aleatorio. El hecho de poder generar filtros útiles de modo no supervisado abre nuevas posibilidades para la explotación del método, con potencial aplicabilidad a todas las capas de la red.

En **segundo** lugar, se planteó como objetivo la comparación, en cuanto a desempeño, de una misma arquitectura convolucional con diferentes métodos de entrenamiento de filtros.

Cada método genera un conjunto diferente de filtros para la primera capa de dicho modelo. Realizar un análisis comparativo de los diversos métodos en cuanto al desempeño obtenido, permite ganar en comprensión respecto al funcionamiento de las redes convolucionales. El hecho de que un método de entrenamiento en particular permita obtener conjuntos de filtros

de mayor utilidad a la red, de modo consistente, permite indagar en las razones detrás de estas diferencias en desempeño. Cada método de entrenamiento de filtros introduce, a través de los algoritmos utilizados, determinado sesgo inductivo. Comparar los resultados obtenidos con uno u otro método es, en definitiva, un modo de contrastar la validez de los sesgos inductivos de cada uno de ellos, en cuanto a su utilidad para el resto de la red. Identificar qué sesgo inductivo es más útil, permite realizar nuevas conjeturas sobre el funcionamiento de las redes convolucionales.

Un objetivo **adicional** para esta serie de experimentos es el de poder establecer un punto de comparación entre filtros generados a partir de determinado sesgo inductivo y aquellos generados de modo aleatorio. Esto permite clarificar el impacto de determinadas políticas para el entrenamiento de filtros y poder comparar dichas políticas con los resultados obtenidos por medio de la generación meramente probabilística. Asimismo, permite determinar cuánto influyen estas políticas en el resultado final de desempeño de la red en contraposición a la generación basada en un simple muestreo aleatorio.

Finalmente, se plantea también la necesidad de evaluar el impacto del posterior entrenamiento con GDBP de los filtros generados, en el desempeño general de la red.

A partir de los conjuntos de filtros generados por cada método, se corren entrenamientos por medio de GDBP para analizar el estado final del mismo conjunto de filtros. Se comparan estos conjuntos en su estadio inicial y su estadio final, luego del entrenamiento. El objetivo, nuevamente, es el de ganar en comprensión sobre el modo en que operan las redes convolucionales, particularmente en este caso GDBP. Asimismo, aportar un criterio adicional de evaluación de los diversos métodos. Podría esperarse poca variación entre la inicialización del filtro y su estado final, para aquel o aquellos métodos de obtención de filtros que impacten de forma más favorable el desempeño general de la red.

5.2. Criterios para la realización de los experimentos

Para poder hacer comparables los resultados de las diferentes estrategia de entrenamiento de filtros entre sí y en relación a estrategias tradicionales de entrenamiento (GDBP) se plantean una serie de criterios para la realización de los experimentos.

1. Para todos los experimentos se utilizarán dos arquitecturas de red predefinidas, tanto en cuanto a sus capas convolucionales, como en sus capas densas, de normalización por lotes, de aumentación de datos, etc. Se utilizarán el modelo EfficientNet V2 Small y un

segundo modelo alternativo construido a propósito de esta serie de experimentos denominado SimpleConv. Ambas arquitecturas pueden encontrarse en el apéndice de este trabajo.

2. En todos los experimentos realizados en este trabajo se han utilizado filtros de 3x3 con relleno de 1 píxel para mantener las dimensiones originales de entrada de la imagen y zancada de 1, es decir, desplazando el filtro de a un píxel por paso.

3. Todos los experimentos serán realizados con ambas arquitecturas y sobre 2 conjuntos de datos de amplia utilización en literatura: CIFAR10 y Mnist-Fashion.

4. Cada experimento se realizará con validación cruzada, promediando los resultados. Este procedimiento permite minimizar los riesgos de resultados no concluyentes producto de la estocasticidad propia del procedimiento de entrenamiento con inicialización aleatoria.

5. Cada experimento de una estrategia de entrenamiento de filtros alternativa estará constituido por el entrenamiento de los modelos con 4 variantes:

    a. Entrenamiento tradicional por GDBP con inicialización Glorot Uniforme para todas las capas
    b. Inicialización Glorot uniforme para la capa inicial y frisado de los pesos de esta capa. Entrenamiento por GDBP para todo el resto de las capas.
    c. Inicialización con los parámetros de los filtros generados a partir de la estrategia correspondiente para la primera capa y frisado de los pesos de dicha capa. Entrenamiento por GDBP para todo el resto del modelo.
    d. Inicialización con los parámetros de los filtros generados a partir de la estrategia correspondiente para la primera capa SIN frisado de los pesos para dicha capa. Entrenamiento por GDBP para todo el resto del modelo.

    Para cada combinación de modelo y conjunto de datos se establecerán hiperparámetros fijos: tasa de aprendizaje inicial y su secuencia de decaimiento, en caso de aplicar y su tamaño de lote. Todos los experimentos realizados sobre esa combinación de modelo y conjunto de datos serán realizados utilizando los mismos hiperparámetros.

5.3. Criterios de evaluación

Para la evaluación de los diferentes métodos de entrenamiento de conjuntos de filtros se plantean los siguientes criterios:

1. Valor final de *accuracy* (total de clasificaciones correctas sobre el total de clasificaciones realizadas) para los conjuntos de datos de entrenamiento y validación

2. Tiempo (medido en épocas de entrenamiento) de convergencia del modelo sobre ambos conjuntos de datos

3. Cambios producidos en el conjunto de filtros producto del entrenamiento por medio de GDBP, medidos por medio de la distancia coseno. Para cada filtro del conjunto, se calcula la distancia coseno entre su parametrización inicial y final. Luego, se promedian los valores para el conjunto de filtros generados.

4. Impacto del tamaño muestral en el entrenamiento de filtros. La evaluación se realiza calculando la distancia coseno entre cada filtro y el filtro más cercano obtenido con el conjunto entero de datos. Luego se promedian las distancias para el conjunto de filtros.

5. Impacto del balance de clases en el entrenamiento de filtros. La evaluación se realiza calculando la distancia coseno entre cada filtro y el filtro más cercano obtenido con una muestra estratificada. Luego se promedian las distancias para el conjunto de filtros.

6. Impacto de la selección del conjunto de datos en el entrenamiento de filtros. La evaluación se realiza calculando la distancia coseno entre cada filtro y el filtro más cercano obtenido con un conjunto de datos alternativo. Luego se promedian las distancias para el conjunto de filtros.

5.4. Resultados obtenidos para los métodos de entrenamiento de filtros propuestos

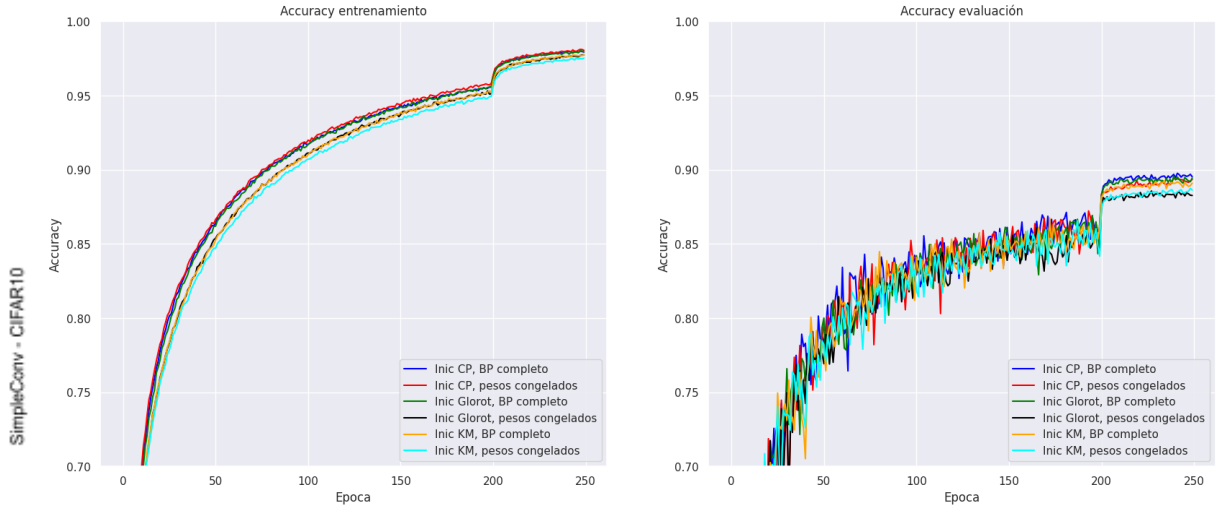
A continuación se visualizan los entrenamientos realizados para todos los métodos propuestos y su comparación con el método tradicional de GDBP. Para cada método, se midió el desempeño de la red en términos de *Accuracy* y velocidad de convergencia. En ningún caso se utilizaron conjuntos de datos adicionales para pre entrenar la red. En todos los casos se partió de una red inicializada con pesos aleatorios.

Para cada método de entrenamiento de filtros, se visualizan los resultados de los entrenamientos realizados con las siguientes combinaciones de modelos y conjuntos de datos:

	CIFAR10	Mnist-Fashion
Arquitectura SimpleConv	✓	✓
EfficientNet	✓	✓

Para cada uno de estos entrenamientos, los resultados permiten comparar el desempeño en términos de *Accuracy* y la velocidad de convergencia para el conjunto de datos de entrenamiento y evaluación en 4 estrategias diferentes:

	Pesos congelados	Pesos entrenables
Inicialización por método propuesto	✓	✓
Inicialización Glorot Uniforme	✓	✓



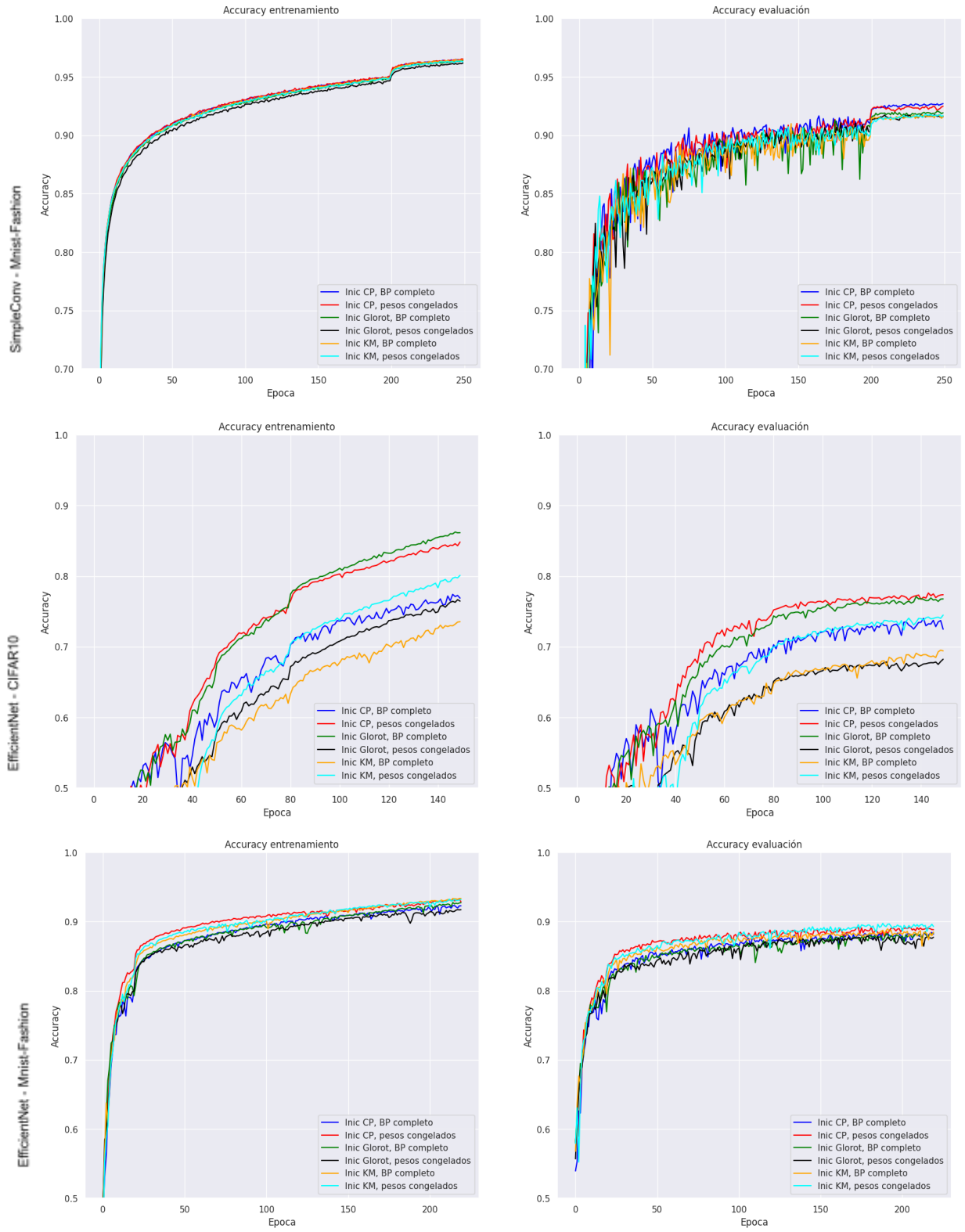


Fig 5.1: Cada par de gráficos ilustra los resultados de accuracy y velocidad de convergencia para el conjunto de datos de entrenamiento (izquierda) y para el conjunto de evaluación (derecha). Los colores indican las 6

modalidades de cada entrenamiento: con y sin pesos congelados para Glorot, Componentes Principales (CP) y K-Medias (KM).

Inicialización	Congelado	SimpleConv	SimpleConv	EfficientNet	EfficientNet
		CIFAR10	Mnist-Fashion	CIFAR10	Mnist-Fashion
Glorot	No	89.43 %	91.91 %	76.78 %	88.33 %
Glorot	Si	88.26 %	91.57 %	68.23 %	87.72 %
KM	No	89.09 %	91.61 %	69.42 %	88.48 %
KM	Si	88.62 %	91.68 %	74.47 %	**89.30 %**
CP	No	**89.52 %**	**92.69 %**	72.48 %	88.17 %
CP	Si	89.34 %	92.47 %	**77.37 %**	88.83 %

Tabla 5.1: Resultados finales de los entrenamientos sobre el conjunto de evaluación para ambas arquitecturas, ambos conjuntos de datos y los 6 procedimientos de entrenamiento juntos, con y sin pesos congelados, para Glorot, Componentes Principales (CP) y K-Medias (KM). Valores en términos de accuracy (porcentaje de clasificaciones correctas sobre el total de instancias de evaluación).

A modo de resumen de los resultados obtenidos en los diferentes entrenamientos y con el fin de facilitar el posterior análisis, se presenta una gráfica con los valores finales de *accuracy* obtenidos. En la figura 5.2 se comparan los desempeños de ambos métodos de obtención de filtros desarrollados en este trabajo (Componentes Principales CP y K-Medias KM) y el entrenamiento supervisado por medio de GDBP.

Para los dos métodos desarrollados en este trabajo y adicionalmente GDBP, se presentan dos alternativas para la configuración de los pesos de la primera capa convolucional: inicialización y posterior entrenamiento por medio de GDBP, e inicialización y congelamiento de los pesos (es decir, sin entrenamiento posterior de la capa).

Los resultados se presentan de forma independiente para cada combinación de arquitectura de modelo y conjunto de datos. Los resultados se presentan en cuatro bloques diferenciados.

Es importante destacar y recordar que todos los resultados presentados en este trabajo corresponden a variaciones de entrenamiento realizadas siempre y únicamente sobre la primera capa convolucional de ambas arquitecturas (SimpleConv y EfficientNet). El resto de las capas fueron entrenadas de modo tradicional, es decir, por medio de GDBP. Es por ello que la escala del impacto en el desempeño, entre uno u otro método de entrenamiento, es reducido.

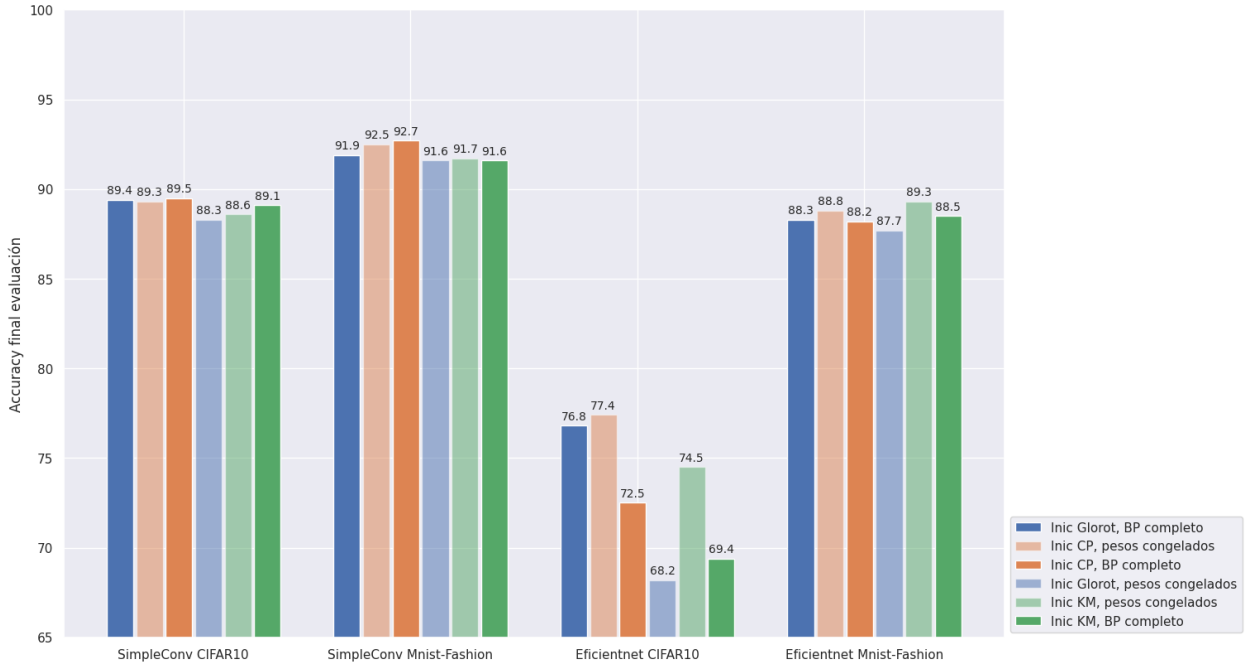


Fig 5.2: Desempeño de cada una de las 6 estrategias Componentes Principales (CP), K-Medias (KM) y Glorot con y sin entrenamiento posterior, en términos de accuracy (porcentaje de instancias correctamente clasificadas por el modelo) para cada combinación de arquitectura y conjunto de datos utilizados.

A modo de validación adicional, se presentan los resultados del entrenamiento con conjuntos de filtros generados por medio de muestreo aleatorio de parches. Es decir, parches al azar (distribución uniforme) tomados de las imágenes que conforman el conjunto de datos. Se visualizan a continuación los resultados de los entrenamientos realizados con:

	27 filtros capa 1	128 filtros capa 1
SimpleConv - CIFAR10	✓	✓

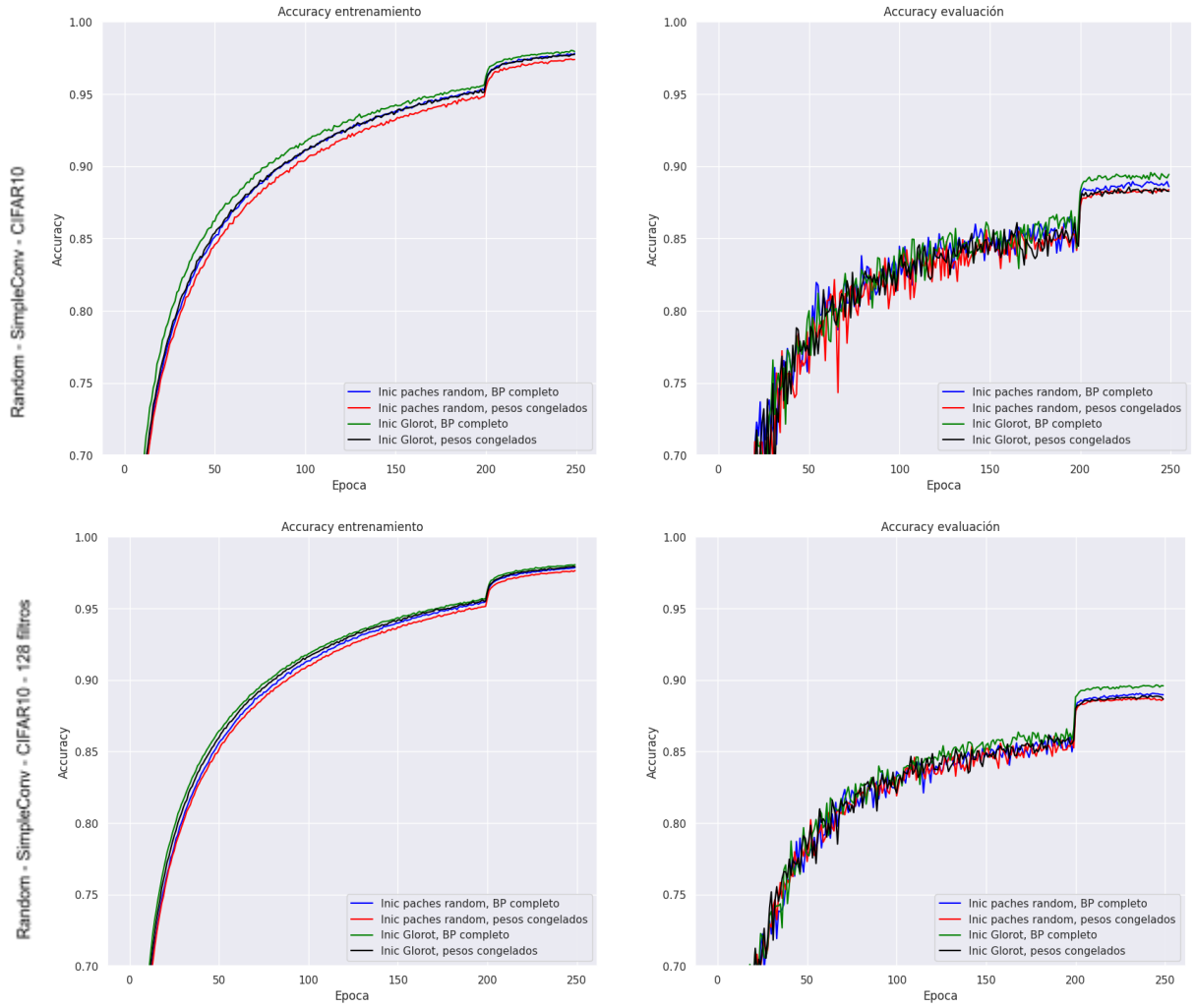


Fig 5.3: Cada par de gráficos ilustra los resultados de accuracy y velocidad de convergencia para el conjunto de datos de entrenamiento (izquierda) y para el conjunto de evaluación (derecha). Los colores indican las 4 modalidades de entrenamiento.

Como se puede observar en la figura 5.3, la selección aleatoria de parches del conjunto de imágenes para conformar una capa convolucional no permite obtener buenos resultados de desempeño. Tanto si se permite el posterior entrenamiento de los filtros por GDBP como si se los conserva congelados, la red experimenta una merma en su desempeño.

Considerando este último análisis así como todos los resultados anteriormente expuestos, se puede arribar a las siguientes conclusiones:

a. Dependencia de la arquitectura y del conjunto de datos

A partir de los resultados obtenidos, se hace evidente que el desempeño de la red en la tarea de clasificación para los diferentes métodos de entrenamiento no es independiente

de la arquitectura de la red ni del conjunto de datos. Para cada combinación de arquitectura y conjunto de datos, el mejor desempeño se obtiene con un método de entrenamiento diferente.
Cabe mencionar que, de por sí, existen diferencias estructurales entre ambos conjuntos de datos. En un caso se trata de imágenes en escala de grises, con un único canal (Mnist-Fashion), mientras que en el otro caso, el conjunto de datos consta de imágenes de tres canales, R, G y B (CIFAR10).

Debe considerarse que ambos métodos de entrenamiento de filtros se basan en la distribución de los parches extraídos de estos conjuntos de imágenes, y que dichos parches son vectorizados previamente a poder aplicar ambos algoritmos. Esto significa que la variabilidad espacial y la variabilidad entre canales son tratadas del mismo modo, perdiéndose la diferenciación entre ambas.

Este hecho no tiene ningún impacto en el caso de Mnist-Fashion por tratarse de un conjunto de imágenes de un solo canal, pero sí lo tiene para el conjunto de datos de CIFAR10, con imágenes en RGB.

b. Inicialización Glorot Uniforme con pesos congelados

Una de las observaciones más claras a primera vista de los resultados, es el desempeño del método de inicialización Glorot Uniforme con pesos congelados. A través de todas las combinatorias de arquitectura y conjuntos de datos, es éste el método con peor desempeño de forma consistente.

Si bien la cantidad de experimentos realizados no permiten realizar aseveraciones tajantes al respecto, bien puede concluirse de esta observación que la obtención de pesos para filtros convolucionales de este modo, dista de ser óptima. Cualquiera de los dos métodos propuestos en este trabajo, superan el desempeño de Glorot Uniforme consistentemente, a través de todas las configuraciones utilizadas de arquitectura y conjunto de datos.

Este resultado no resulta sorpresivo, si se considera que ambos métodos de entrenamiento de filtros, Componentes Principales (CP) y K-Medias (KM), se basan en la distribución del conjunto de datos para obtener los pesos, mientras que Glorot Uniforme es independiente de ellos.

c. Inicialización Glorot Uniforme con GDBP (método estándar de entrenamiento)

Otra de las observaciones destacables acerca de los resultados es el desempeño obtenido por el método estándar o tradicional de entrenar las redes convolucionales, es decir, GDBP con inicialización Glorot. Para todas las configuraciones de arquitectura y

conjunto de datos, su desempeño, en ningún caso, ha sido el mejor del conjunto. Este hecho refuerza la hipótesis esbozada en capítulos previos acerca del carácter sub óptimo de este método de entrenamiento para la primera capa de la red. En referencia a este punto, particularmente el capítulo 3 presenta evidencia adicional acerca de esta hipótesis, ilustrando la existencia tanto de filtros muertos (*dead kernels*) como filtros redundantes (con respuestas de activación muy similares).

d. Ajuste de parámetros por GDBP para métodos alternativos de entrenamiento de filtros convolucionales

En base a los resultados obtenidos, cabe en este punto preguntarse por la utilidad o conveniencia de efectuar el entrenamiento por GDBP luego de la inicialización para el caso de los métodos alternativos de entrenamiento de filtros. ¿Mejora el desempeño de la red entrenando por GDBP los pesos obtenidos por alguno de los métodos de entrenamiento alternativos?

Para el caso de la inicialización Glorot Uniforme, no hay dudas al respecto y se obtienen mejoras de modo en el desempeño de la red de modo consistente para todas las combinatorias de arquitecturas y conjuntos de datos. Este resultado es, por supuesto, completamente esperable y constitutivo del entrenamiento de redes neuronales por GDBP.

Para el caso de entrenamiento de filtros por medio de Componentes Principales (CP), los resultados no son concluyentes. Se puede apreciar que para los entrenamientos sobre la arquitectura SimpleConv se obtienen resultados ligeramente superiores entrenando la capa a posteriori de su inicialización por medio de Componentes Principales (CP). Sin embargo, para los entrenamientos realizados sobre EfficientNet, los resultados indican una pérdida significativa de desempeño de la red al modificar los pesos iniciales, por medio del entrenamiento por GDBP. Esto es particularmente notorio para CIFAR10, donde el desempeño cae de 77.37 % a 72.48 %, mientras que para Mnist-Fashion la caída es de menos de un punto porcentual (de 88.83 % a 88.17 %).

En el caso del entrenamiento de filtros por medio de K-Medias (KM), los resultados son mayormente similares. El entrenamiento posterior por medio de GDBP produce una mejora significativa del desempeño de la red para la arquitectura SimpleConv con el conjunto de datos CIFAR10. Para las otras tres configuraciones, los resultados son opuestos, indicando un perjuicio en el desempeño al modificar los pesos iniciales por medio de GDBP. Particularmente para la combinación de EfficientNet con CIFAR10, el entrenamiento posterior por GDBP produce una merma en desempeño de unos 5 puntos porcentuales.

Los resultados no parecen ser concluyentes, en cuanto a la utilidad o no de efectuar el entrenamiento por GDBP, posterior a la inicialización por los métodos de entrenamiento aquí propuestos. En todo caso, queda claro el impacto sobre este punto, de la configuración de conjunto de datos y arquitectura utilizada, especialmente esta última.

e. K-Medias (KM) y Componentes Principales (CP)

Resta analizar y comparar los desempeños de ambos métodos de entrenamiento de filtros convolucionales. En este punto, si bien los resultados no son definitivos, Componentes Principales (CP) obtiene mejores desempeños en la mayoría de las configuraciones; en algunos casos, como con EfficientNet + CIFAR10, por amplio margen. K-Medias (KM) se desempeña mejor solamente en la configuración EfficientNet y Mnist-Fashion. Considerando estos resultados podría suponerse que el método de entrenamiento basado en Componentes Principales (CP) es superior en términos generales, siendo los resultados obtenidos para EfficientNet - Mnist-Fashion un caso excepcional. Puede deberse esto a las particularidades del conjunto de datos (1 solo canal, escala de grises), pero resultará imprescindible en futuros trabajos realizar un mayor número de experimentos que puedan corroborar esta hipótesis.

A través de los resultados obtenidos puede comprobarse de modo consistente el desempeño superior en cuanto a inicialización, de ambos métodos de entrenamiento de filtros por sobre la inicialización Glorot Uniforme. En las cuatro configuraciones (SimpleConv-CIFAR10, SimpleConv-Mnist-Fashion, EfficientNet-CIFAR10 y EfficientNet-Mnist-Fashion), ambos métodos propuestos, sin entrenamiento posterior por GDBP, han resultado superiores a la inicialización Glorot Uniforme.

El hecho de obtener mejores resultados de desempeño con técnicas alternativas y de forma consistente a través de las distintas configuraciones de arquitecturas y conjuntos de datos, sienta las bases para futuras investigaciones con aplicabilidad a todas las capas de los modelos convolucionales. Cabe recordar que los experimentos y resultados aquí presentados corresponden a la aplicación de métodos alternativos de entrenamiento de filtros solo para la primera capa convolucional.

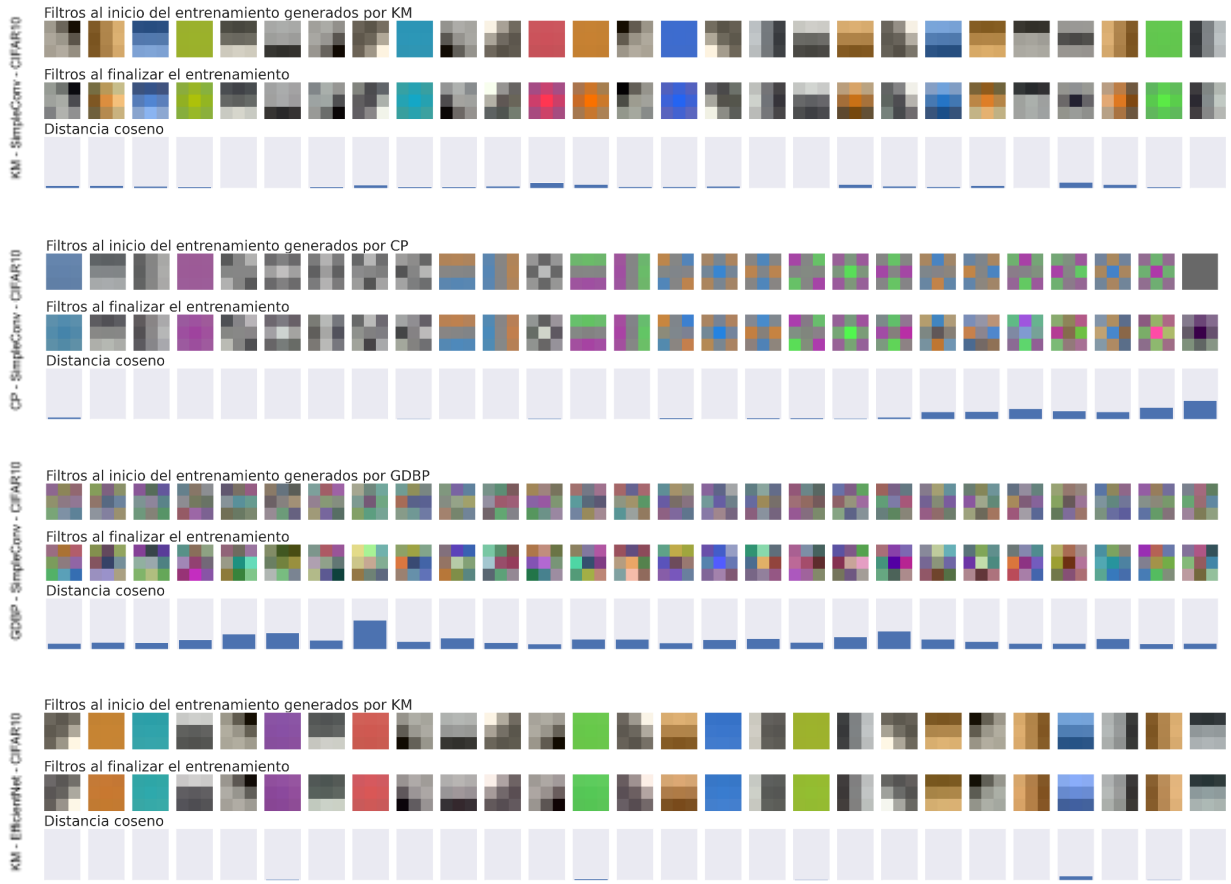
5.5. Filtros obtenidos por medio de los diferentes métodos propuestos

Complementariamente a la sección anterior, se visualizan los filtros generados para cada método de entrenamiento, sobre cada conjunto de datos y cada arquitectura, así como su estado

final, luego de aplicar el entrenamiento por medio de GDBP y la distancia coseno promedio entre su estado inicial y final.

Para todos los experimentos sobre CIFAR10 (métodos y arquitecturas) se generaron 27 filtros. La razón detrás de esta elección radica en las limitaciones propias del método de entrenamiento de filtros por medio de Componentes Principales (CP). Ocurre que el método puede generar como máximo, tantos filtros como dimensiones posean los parches de imágenes que se utilizan como entrada. Para el caso de imágenes a color de 3 canales (RGB) con parches de 3x3 píxeles, esto resulta en una dimensionalidad de 27 (3 canales x 3 píxeles de ancho x 3 píxeles de alto). Con la finalidad de hacer comparables los resultados obtenidos con los diferentes métodos y arquitecturas, se optó por fijar en 27 (el máximo posible para Componentes Principales) el número de filtros a emplear.

De modo análogo, en todos los experimentos realizados sobre Mnist-Fashion se emplearon 9 filtros de modo de hacer comparables los resultados entre diferentes arquitecturas y métodos. Ocurre que en el caso de Mnist-Fashion, con un solo canal y parches de 3x3, son 9 la máxima cantidad de filtros que es posible generar por medio de Componentes Principales (CP).



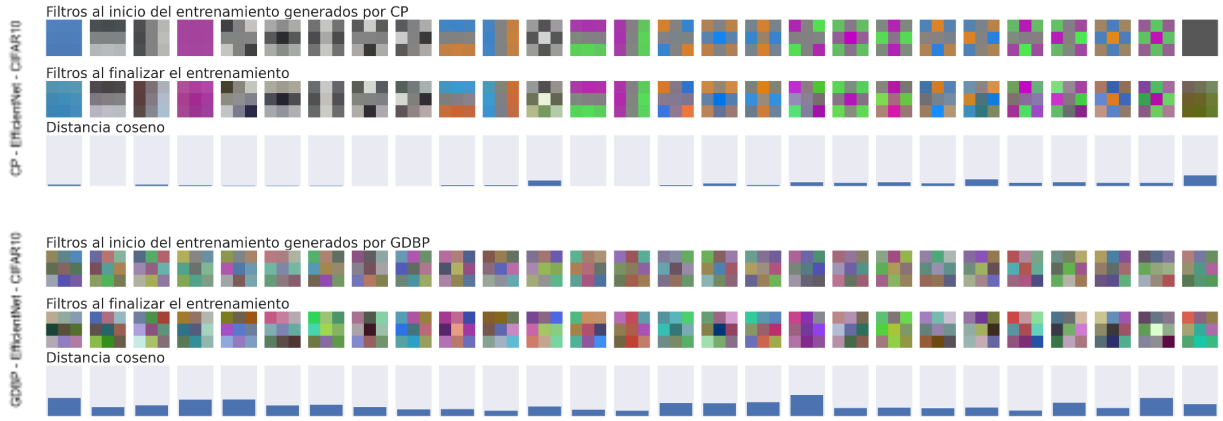


Fig 5.4: Filtros resultantes del entrenamiento por medio de K-Medias (KM), Componentes Principales (CP) y GDBP sobre el conjunto de datos CIFAR10. En la fila superior se observa el resultado de la inicialización, en la fila central se visualizan los mismos filtros luego del entrenamiento por medio de GDBP y en la fila inferior la distancia coseno entre la configuración inicial y final de cada filtro.

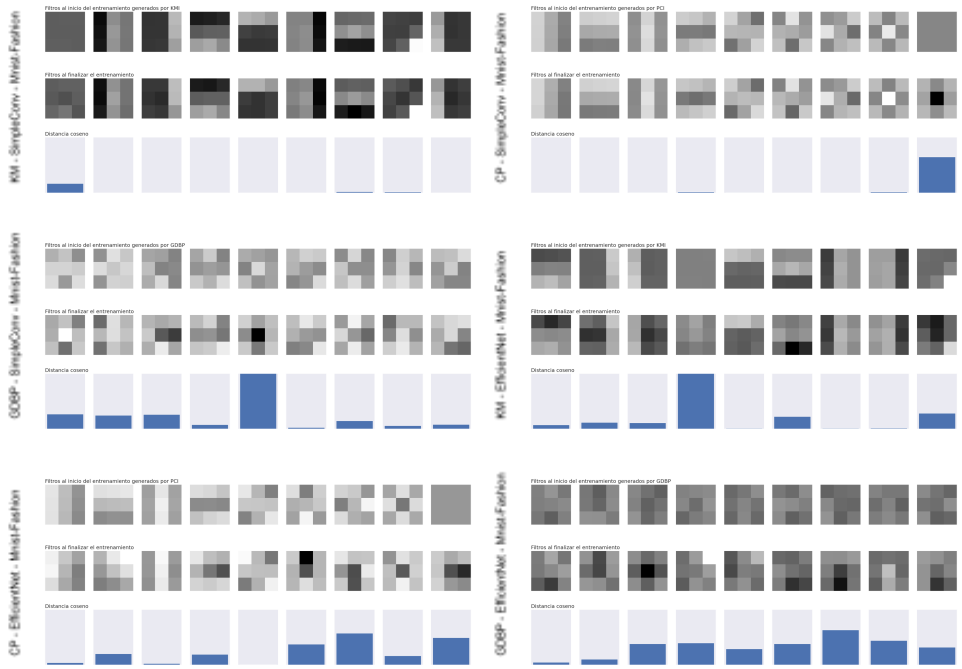


Fig 5.5: Filtros resultantes del entrenamiento por medio de K-Medias (KM), Componentes Principales (CP) y GDBP sobre el conjunto de datos Mnist-Fashion. En la fila superior se observa el resultado de la inicialización, en la fila central se visualizan los mismos filtros luego del entrenamiento por medio de GDBP y en la fila inferior la distancia coseno entre la configuración inicial y final de cada filtro.

Para las 4 configuraciones (arquitectura y conjunto de datos) los filtros inicializados con Glorot Uniforme son los que mayores modificaciones experimentan durante el entrenamiento por

GDBP. Los filtros inicializados por medio de Componentes Principales (CP) y K-Medias (KM), son modificados en menor medida durante el entrenamiento. Esto parecería sugerir que la inicialización por medio de cualquiera de estos dos métodos ya provee cierto acercamiento a un mínimo local, para la primera capa convolucional.

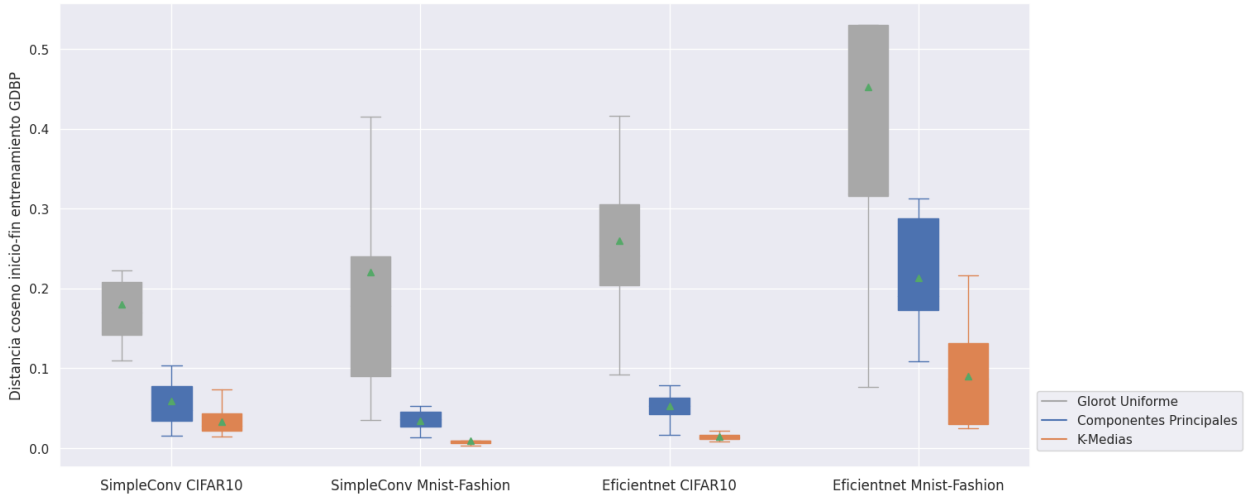


Fig 5.6: Distancias coseno para el conjunto de filtros entre su configuración inicial obtenida por cada uno de los métodos de entrenamiento y su configuración final, luego del entrenamiento por medio de GDBP.

Adicionalmente se puede observar que los filtros que sufren mayor modificación en el caso de inicialización por Componentes Principales (CP) son los últimos del conjunto. Parece incluso existir cierto crecimiento en la distancia coseno entre filtro inicial y final a medida que se avanza progresivamente sobre el conjunto de filtros. Este hecho podría tener su fundamento teórico en el modo en que estos filtros son generados. Su orden no es aleatorio ya que fueron generados a partir de los autovectores producto de la descomposición por medio de Componentes Principales. El primero de ellos es el que mayor variabilidad de los datos de la muestra explica, mentiras que el último filtro es el que menor varianza explicada aporta.

Los resultados son consistentes para ambos conjuntos de datos y ambas arquitecturas.

Para el caso del método de entrenamiento por medio de K-Medias (KM), no existe un ordenamiento similar del conjunto de filtros generados, tal como se da en Componentes Principales (CP). Sin embargo, la distancia coseno media entre la configuración inicial y la configuración final de los filtros, luego del entrenamiento por GDBP, resulta consistentemente menor a la observada en Glorot Uniforme y en casi todas las configuraciones menor a Componentes Principales (CP).

Como puede apreciarse en la figura 5.6, la red realiza modificaciones sustancialmente inferiores en términos cuantitativos, durante el entrenamiento por GDBP, para ambos métodos de

inicialización propuestos. Estos resultados son consistentes a través de las 4 configuraciones (arquitectura y conjunto de datos). Es decir, ambos métodos de entrenamiento de filtros proveen una configuración inicial de parámetros que resultan mucho más eficientes en términos de las necesidades posteriores de optimización por medio del entrenamiento por GDBP.

Las implicancias de estos resultados son de suma trascendencia en el marco del entrenamiento de redes convolucionales.

Por un lado, se presentan dos métodos de obtención de filtros convolucionales que pueden utilizarse como métodos de inicialización, generando parámetros que requieren menos ajustes durante el entrenamiento que los métodos de inicialización habituales. Esto redunda, por supuesto, en la optimización de recursos computacionales durante el entrenamiento, ya que potencialmente implica la necesidad de un menor número de iteraciones para obtener la convergencia de la red.

Por otra parte, permite especular con la posibilidad de obtener un método de entrenamiento de redes convolucionales libre del costoso proceso de GDBP. Si los filtros obtenidos por los métodos de entrenamiento propuestos no sufren mayores modificaciones durante el proceso de entrenamiento, esto bien podría suponer que los filtros ya se encuentran muy cerca de un mínimo local. Podría evaluarse en futuros trabajos, la factibilidad de prescindir directamente del ajuste posterior de parámetros por medio de GDBP.

5.6. Resultados comparativos para el entrenamiento de filtros por medio de K-Medias y K-Medoides

A partir de los experimentos realizados para el entrenamiento de filtros de primera capa por medio del algoritmo de K-Medias, se plantea la siguiente afirmación. Al tomarse los centroides producto del agrupamiento no supervisado de los parches de imagen y tratándose de un espacio de alta dimensionalidad (d=27 en el caso de CIFAR10), podría ocurrir que los centroides resulten de promediar parches muy alejados entre sí. En dicho caso, los filtros generados podrían no ser representativos del agrupamiento sino meros centroides de cúmulos esparsos y poco densos.

Para verificar la validez de esta hipótesis, se procedió a comparar los centroides y los medoides de una misma agrupación por medio de K-Medias. Es decir, para una única ejecución del algoritmo de K-Medias, se compararon los centroides con los medoides de cada agrupación. En

caso de que la hipótesis fuera válida, los filtros generados a partir de una y otra estrategia deberían divergir de forma significativa, ya que K-Medoides no toma el valor promedio del conjunto de datos sino su mediana. No se plantea ninguna métrica específica para cuantificar dicha divergencia dejando la evaluación a la subjetividad de una mera inspección visual.

Para CIFAR10 y Mnist-Fashion, estos son los filtros generados a partir de una muestra aleatoria de 100 imágenes (90.000 parches y 67.600 respectivamente) para K-Medias y K-Medoides:

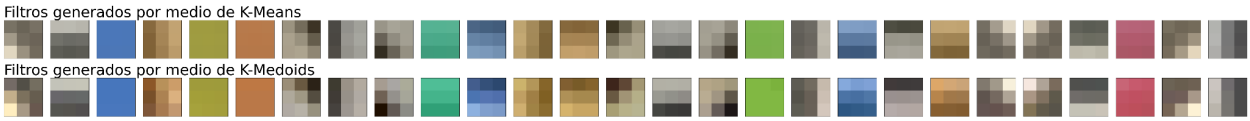


Fig 5.7: Filtros generados por medio de K-Medias y K-Medoides sobre CIFAR10 con una muestra aleatoria de 90.000 parches.

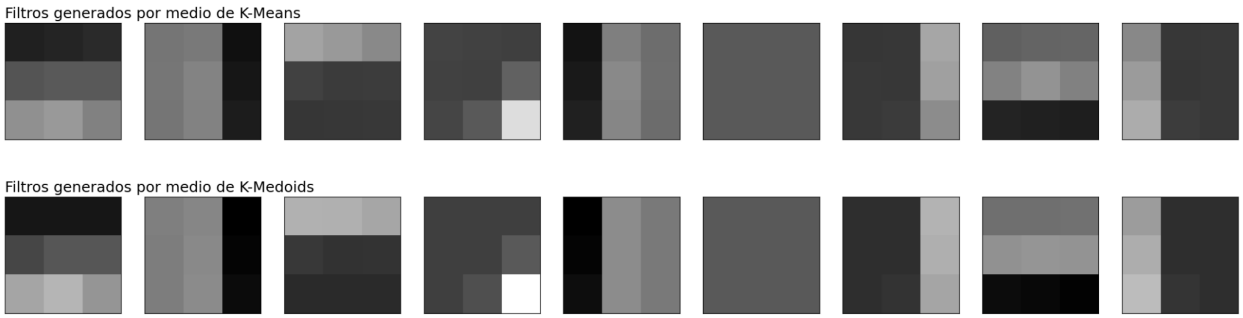


Fig 5.8: Fila superior: filtros generados por medio de K-Medias. Fila inferior: filtros generados por medio de K-Medoides. En ambos casos sobre el conjunto de datos Mnist-Fashion.

Si se comparan los filtros generados por uno y otro método, puede comprobarse la extrema similitud entre ambos conjuntos de filtros.

Si bien no se han realizado entrenamientos con filtros generados a partir de K-Medoides, la inspección visual confirma la presencia de agrupamientos sin sesgos notorios en la distribución de los parches de imágenes en el espacio de 27 y 9 dimensiones (CIFAR10 y Mnist-Fashion, respectivamente), ubicando los puntos medios y medianos de la distribución multivariada extremadamente cerca, unos de otros.

Puede concluirse entonces que resulta indistinto la utilización de uno u otro algoritmo (K-Medias y K-Medoides) en términos de los filtros que se generan. Este hecho refleja la extrema cercanía entre media y mediana de la distribución de parches para cada uno de los agrupamientos. En definitiva, este hallazgo permite concluir que el centroide encontrado por medio de K-Medias es en sí, uno de los parches. Es decir, el centroide representativo del grupo

que se utilizará posteriormente como filtro convolucional no es el resultado artificial de un promedio, sino que constituye en sí mismo un elemento de la distribución original.

Este resultado permite especular con la presencia de parches que serían “ejes naturales” de la distribución de la que provienen. Parches que, de modo natural, permiten dividir el espacio a través de uno de sus hiperplanos de mayor variabilidad.

5.7. Impacto del conjunto de datos en el entrenamiento de filtros

En esta sección se exploran varios tipos de dependencias en el entrenamiento de filtros. En particular, cómo varía el conjunto de filtros obtenidos en función del tamaño de muestra adoptado. Adicionalmente, se exploran las diferencias entre filtros generados condicionales a la clase y al conjunto de datos seleccionado.

5.7.1. Impacto del tamaño de muestra en el entrenamiento de filtros

Se plantea la incógnita respecto al impacto del tamaño de muestra en el entrenamiento de los filtros. Se desea indagar en particular, cuánto repercute o cuánto modifica la calidad de los filtros generados por ambos algoritmos, frente a diversos tamaños muestrales.

Este análisis resulta de vital importancia en cuanto a dos puntos centrales. Por un lado, aporta al conocimiento de determinado conjunto de datos, indicando cuántas muestras son necesarias para representar la diversidad de las instancias que lo componen. Por otro lado, permite adecuar de forma más eficiente el empleo de recursos computacionales. En caso de que puedan obtenerse filtros muy similares por medio de un muestreo pequeño del conjunto de datos, esto permite reducir drásticamente el presupuesto de tiempo / infraestructura computacional.

Los siguientes experimentos se realizaron con esta motivación en mente. Se entrenaron ambos algoritmos, Componentes Principales (CP) y K-Medias (KM), sobre ambos conjuntos de datos (CIFAR10, Mnist-Fashion) para diferentes tamaños de muestra:

Imágenes	Parches en CIFAR10	Parches en Mnist-Fashion
1 imagen	900	676
2 imágenes	1800	1352

Imágenes	Parches en CIFAR10	Parches en Mnist-Fashion
4 imágenes	3600	2704
8 imágenes	7200	5408
16 imágenes	14400	10816
32 imágenes	28800	21632
128 imágenes	115200	86528
1024 imágenes	921600	692224
50.000 imágenes	45 millones	33,8 millones

Para todos los experimentos se realizó un muestreo estratificado de acuerdo a la clase de pertenencia de las imágenes. Cabe mencionar que ambos conjuntos de datos están perfectamente balanceados, con igual cantidad de instancias para cada clase.

Para determinar en qué medida se obtienen los mismos o similares filtros con los diferentes tamaños de muestra se utilizó la métrica de similitud coseno. Para cada tamaño de muestra, se compara cada filtro de la serie, con su filtro más cercano de la serie final, la de mayor tamaño muestral.

Finalmente, se toma el valor medio de los diferentes valores de similitud coseno de cada filtro para obtener un valor único de similitud por serie. Por supuesto, la última serie, con la que se realizó la comparación, tendrá por fuerza una similitud coseno promedio de 1. Es esta serie la que se utiliza como valor de referencia para el resto ya que es la que se ha generado con la muestra de mayor tamaño (mayor cantidad de parches/imágenes).

Un detalle relevante en la comparación es que se toma el valor absoluto de la similitud coseno. Es decir, dos filtros completamente opuestos en sentido de la dirección de los vectores que los representan en el espacio d dimensional (d=27 para el caso de filtros de 3x3x3) serán tan similares entre sí como lo son dos filtros idénticos. La motivación para tomar el valor absoluto de la similitud coseno está basada en las características de la operación de convolución. Dos filtros completamente opuestos definen un mismo hiperplano de separación y producen una misma distribución de valores de activación pero de sentido opuesto.

Para ejemplificar este hecho, se visualizan a continuación dos filtros generados por Componentes Principales (CP) para diferentes tamaños muestrales, con una similitud coseno

muy cercana a -1 (-0.999). Es decir, son filtros completamente opuestos. Al convolucionarse con la misma imagen, producen mapas de características exactamente inversos.

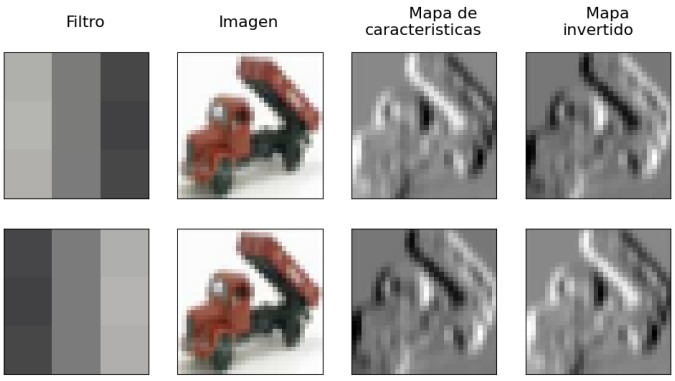


FIg 5.9: En la primera columna, los dos filtros, completamente opuestos (similitud coseno -0.999). En la segunda columna la imagen de entrada. En la tercera, se visualizan los mapas de características producidos por uno y otro filtro. Finalmente, en la última columna se muestran los mismos mapas pero invertidos (multiplicados por -1) para facilitar la comparación.

Se visualizan a continuación los conjuntos de filtros generados por el método de Componentes Principales (CP) sobre CIFAR10 para diferentes tamaños muestrales, así como la similitud coseno media en función del tamaño muestral.

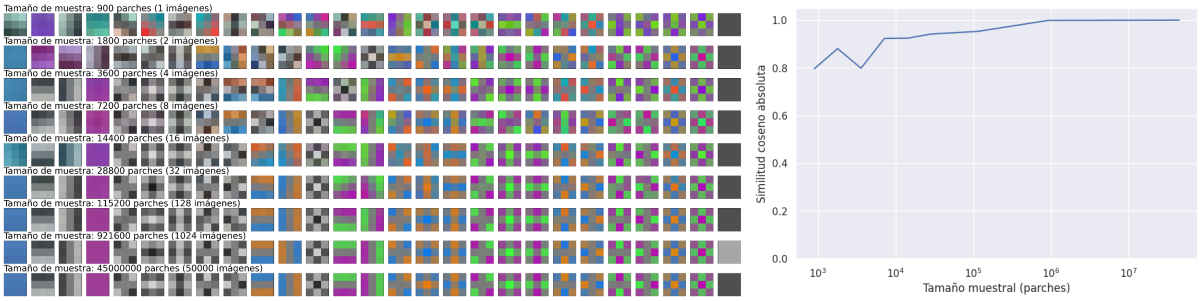


Fig 5.10: Izquierda: Filtros generados para diferentes tamaños muestrales sobre CIFAR10 utilizando el algoritmo de Componentes Principales (CP). Derecha: similitud coseno media en función del tamaño muestral.

A continuación, se visualizan los resultados para el mismo conjunto de datos, pero empleando K-Medias (KM) con inicialización “random” y 3 iteraciones. A diferencia de Componentes Principales (CP), cada ejecución del algoritmo devuelve un conjunto de centroides sin un orden predeterminado. En la visualización se ordenaron los filtros de acuerdo a su cercanía coseno con la última serie (la de muestra de mayor tamaño).

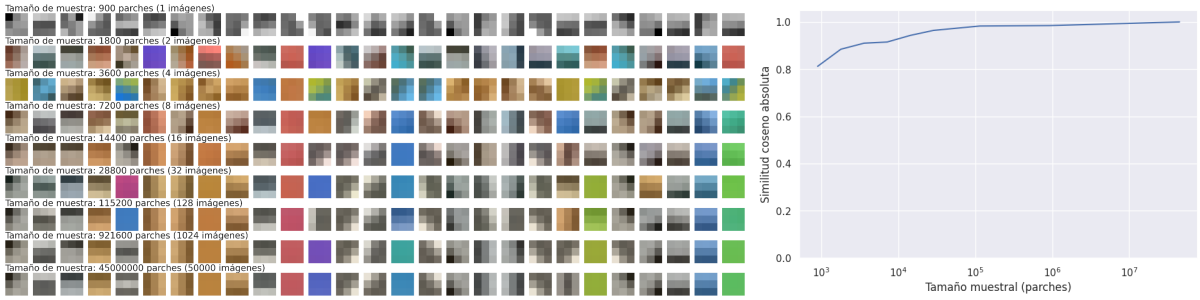


Fig 5.11: Izquierda: Filtros generados para diferentes tamaños muestrales sobre CIFAR10 utilizando el algoritmo de K-Medias (KM). Derecha: similitud coseno media en función del tamaño muestral.

Finalmente, se muestran los resultados del mismo experimento sobre el conjunto de datos Mnist-Fashion, utilizando ambos métodos de entrenamiento, Componentes Principales (CP) y K-Medias (KM).

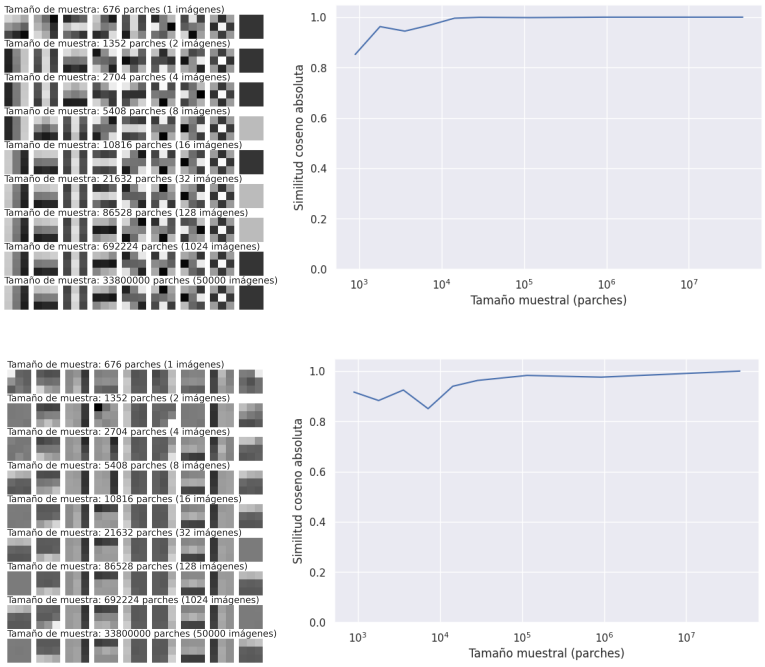


Fig 5.12: Fila superior: Filtros generados para diferentes tamaños muestrales sobre Mnist-Fashion utilizando el algoritmo de Componentes Principales (CP) y a su derecha la similitud coseno en función del tamaño muestral. Fila inferior: Filtros generados para diferentes tamaños muestrales sobre el mismo conjunto de datos (Mnist-Fashion) pero utilizando el método de K-Medias (KM). A su derecha, la similitud coseno media en función del tamaño muestral.

Tal como se puede apreciar en los experimentos, para ambos algoritmos de entrenamiento de filtros y ambos conjuntos de datos, el tamaño de muestra puede ser extremadamente reducido y aún así lograr el entrenamiento de filtros muy similares a los que se obtendrían con el conjunto completo de datos.

Un tamaño de muestra de unos 100.000 parches (unas 100 imágenes) o incluso menor, parece ser suficiente para capturar con precisión, las direcciones de máxima variabilidad, en el caso de Componentes Principales (CP), y los centroides de los agrupamientos, en el caso de K-Medias (KM). Particularmente, Componentes Principales (CP) parece converger más rápidamente en ambos conjuntos de datos, lo cual es en cierto modo esperable ya que se trata de un algoritmo determinista. Contrariamente, K-Medias (KM) introduce un componente de aleatoriedad con cada inicialización, obteniendo centroides de mayor variabilidad entre ejecuciones.

Un punto adicional de consideración para este experimento son los tiempos de ejecución necesarios para realizar los entrenamientos. En los siguientes gráficos se pueden visualizar estos tiempos en función del tamaño de muestra, para ambos conjuntos de datos y ambos algoritmos de entrenamiento. Como referencia, los experimentos se han realizado con un procesador Intel i7 de 8 núcleos, es decir, procesados con un CPU, y no con GPU, como suele ser el caso para el entrenamiento de redes neuronales por medio de GDBP.

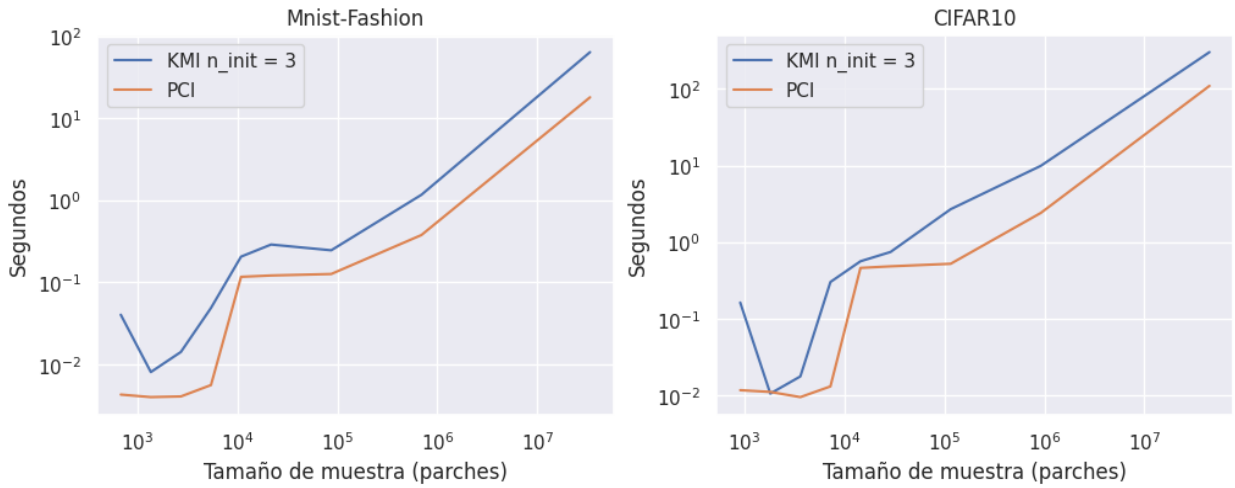


Fig 5.13: Tiempos de ejecución para ambos algoritmos sobre ambos conjuntos de datos.

Para el caso de Componentes Principales (CP), su complejidad temporal es O(p2n+p3), siendo *p* la dimensionalidad de las instancias (27 para CIFAR10 y 9 para Mnist-Fashion) y *n* la cantidad de instancias. El tiempo requerido para su ejecución crece entonces linealmente con la cantidad de instancias (parches) que se utilicen. Lo mismo ocurre con el algoritmo de K-Medias, ya que su complejidad temporal es O (NTK), donde *N* es el número de conjuntos de datos, *K* es la cantidad de grupos y *T* el número de iteraciones.

Estos experimentos pueden considerarse totalmente concluyentes por la consistencia de los resultados obtenidos sobre diferentes conjuntos de datos y diferentes métodos de entrenamiento de filtros. Los experimentos han demostrado que el entrenamiento de filtros

para la primera capa convolucional puede bien realizarse con una muestra muy reducida del conjunto de datos, obteniendo conjuntos de filtros prácticamente idénticos a los que se obtienen con el conjunto completo.

Este punto resulta de suma importancia en relación a la evaluación del costo computacional del entrenamiento por los métodos aquí presentados. El hecho de obtener prácticamente los mismos conjuntos de filtros utilizando una muestra muy reducida del conjunto de datos, permite reducir, así mismo, el costo computacional inherente al entrenamiento de los filtros.

Tomando como ejemplo el caso del conjunto de datos CIFAR10 y utilizando el método de entrenamiento de Componente Principales, queda claro que con una muestra de tan solo 100 imágenes (alrededor de 100.000 parches), se obtienen prácticamente los mismos filtros con utilizando el conjunto de datos completo de 50.000 imágenes (alrededor de 50 millones de parches).

5.7.2. Impacto del balance de clases en el entrenamiento de filtros

Se explora en esta sección el impacto del desbalance de clases en el entrenamiento de filtros. Se intenta dilucidar en qué grado, la pertenencia a determinada clase de una imagen sobre la que se obtienen los parches, influye en la distribución de estos. Es decir, en cuánto modifica a la distribución del muestreo y por ende, los filtros obtenidos por los diferentes métodos, el hecho de sesgar la muestra para la obtención de los parches hacia determinada clase en particular.

La hipótesis aquí planteada es que, por la naturaleza del tipo de parches que se están extrayendo (primera capa convolucional, de bajo nivel), la pertenencia de las imágenes a una u otra clase no tiene impacto en la distribución de los parches muestreados. Por ende, los filtros obtenidos a partir de esta muestra de parches son muy similares a los obtenidos a partir de una muestra balanceada.

Utilizando un tamaño de muestra fijo de 1024 imágenes (921.000 parches para CIFAR10 y 692.224 para Mnist-Fashion), se generaron 10 conjuntos de filtros, uno por clase. Cada conjunto de filtros es generado con imágenes pertenecientes a una clase exclusiva en particular.

Se visualizan a continuación los resultados de los experimentos realizados para los dos conjuntos de datos (CIFAR10 y Mnist-Fashion) y los 2 métodos de entrenamiento de filtros (Componentes Principales y K-Medias con 3 iteraciones e inicialización “random”) ilustrando los filtros generados para cada clase y la distancia coseno media con los filtros obtenidos por medio de muestreo estratificado.

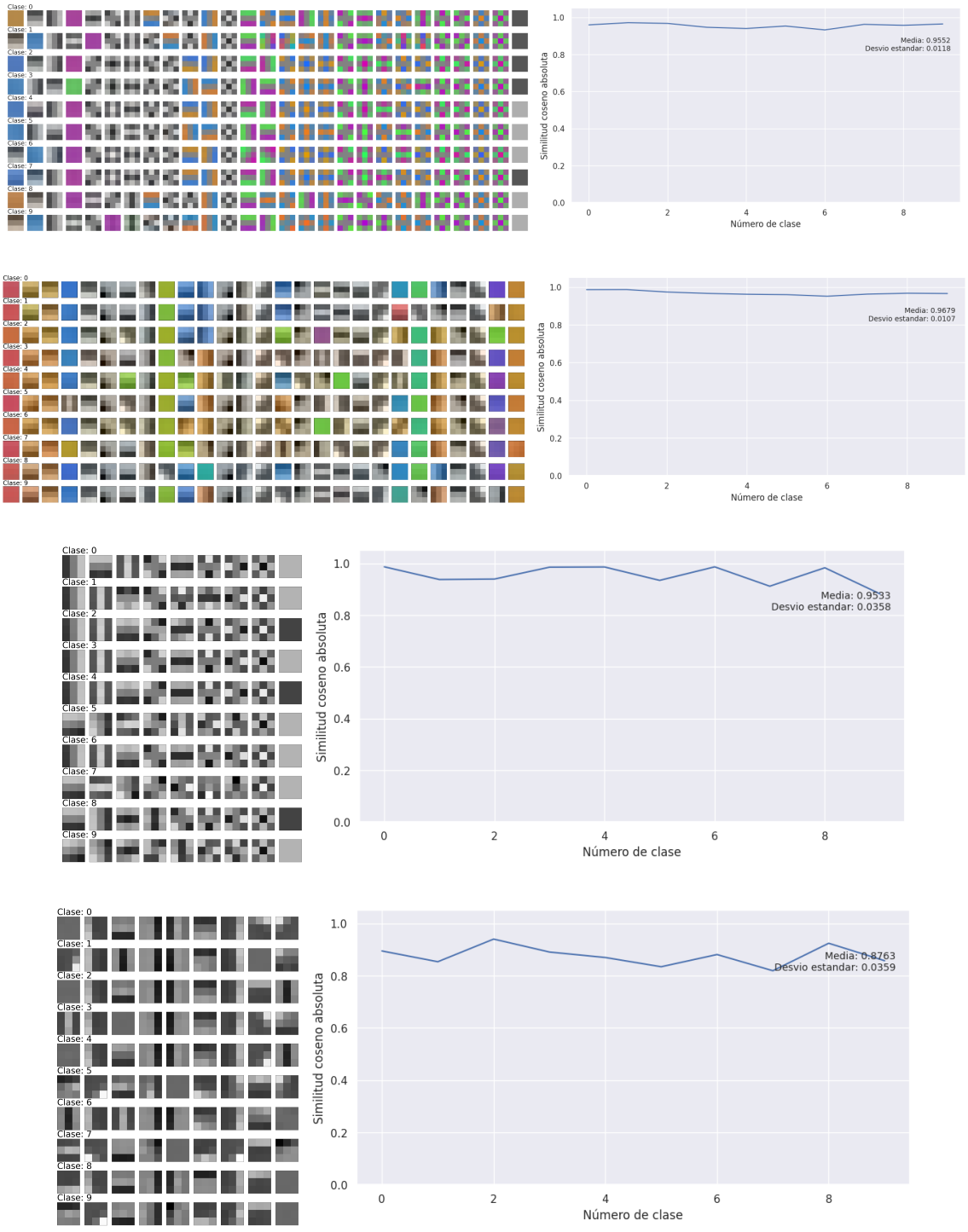


Fig 5.14: Fila 1: CIFAR10 y Componentes Principales (CP). Fila 2: CIFAR10 y K-Medias (KM). Fila 3: Mnist-Fashion y Componentes Principales (CP). Fila 4: Mnist-Fashion y K-Medias (KM). Para cada fila se ilustra a la Izquierda los filtros generados, donde cada fila del mosaico representa un conjunto de filtros obtenidos con un tamaño muestral de 1024 imágenes tomadas de una única clase. A la derecha se ilustra la Similitud coseno absoluta media para cada conjunto de filtros generados condicionados a una clase específica y el conjunto de filtros generado por muestreo estratificado.

En relación al impacto del desbalance de clases para el muestreo de parches y el posterior entrenamiento de los filtros, los resultados pueden considerarse totalmente consistentes a través de los diferentes conjuntos de datos y métodos de entrenamiento de filtros. Se confirma entonces la hipótesis planteada y se puede afirmar de forma concluyente que no existe impacto visible del desbalance de clases en el entrenamiento de filtros por medio de los métodos presentados en este trabajo. Al generar filtros por ambos métodos a partir de conjuntos de datos completamente desbalanceados, se obtienen conjuntos de filtros muy similares entre sí. Por completamente desbalanceados, se hace aquí referencia al empleo de subconjuntos de datos con la presencia de una única clase.

Tomando los subconjuntos de filtros conformados por una única clase para CIFAR10 para cualquiera de los dos métodos de entrenamiento, la similitud coseno media con respecto al conjunto de datos completamente balanceado es de cerca de 0.95. Esto indica una gran similitud entre los conjuntos de filtros generados con conjuntos completamente desbalanceados y el conjunto de datos completamente balanceado. Situación, de por sí, completamente extrema y sesgada, pero que permite contrastar con mayor contundencia ambas situaciones: conjunto de datos completamente balanceado y conjunto de datos completamente desbalanceado.

5.7.3. Impacto del conjunto de datos en el entrenamiento de filtros

En este experimento se comparan los filtros generados por ambos algoritmos para dos conjuntos de datos diferentes, ambos en color. Se parte del supuesto de que, para filtros de bajo nivel, como los que conforman una primera capa convolucional, con un campo receptivo muy acotado (3x3 píxeles de la imagen, en este caso), prácticamente cualquier conjunto de imágenes naturales tendrá una distribución muy similar, generando filtros muy similares.

A continuación se comparan los filtros generados por Componentes Principales (CP) y por K-Medias (KM) con inicialización “random” y 3 iteraciones, sobre una muestra de 921.600 parches (1024 imágenes) de CIFAR10 y una muestra del mismo tamaño del conjunto de datos ImageNet:

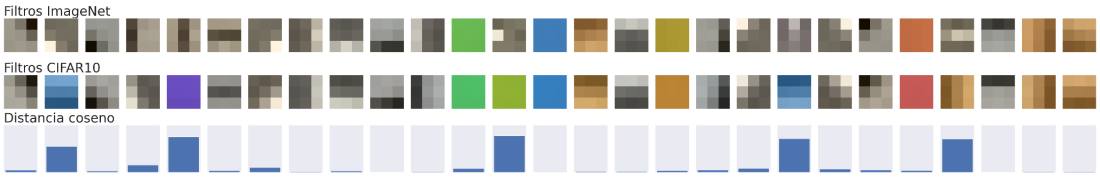


Fig 5.15: En el conjunto de filas superior, la comparación de filtros generados sobre ImageNet y CIFAR10 utilizando Componentes Principales(CP). En el conjunto de filas inferior, la misma comparación utilizando K-Medias (KM) como método de entrenamiento. Se visualizan los dos conjuntos de filtros generados sobre cada conjunto de datos para ambos métodos, así como la distancia coseno entre pares de filtros.

Los resultados de estos experimentos indican una distancia coseno media entre ambos conjuntos de datos para Componentes Principales (CP) de 0.0283 con un desvío estándar de 0.0558. Para el caso de K-Medias (KM), la distancia coseno media entre los filtros generados por CIFAR10 e ImageNet fue de 0.1339 con un desvío estándar de 0.2303.

Para el caso de Componentes Principales (CP), la distancia coseno es realmente muy baja, confirmando la hipótesis planteada. Por su parte, por medio de K-Medias (KM) se obtienen valores mayores que, muy probablemente respondan en cierta medida, al componente aleatorio, no determinista del algoritmo.

De cualquier modo, los resultados obtenidos pueden considerarse consistentes y concluyentes. Resulta evidente, a juzgar por los resultados, que la selección de uno u otro conjunto de datos no tiene impacto sobre el conjunto de filtros generados. Esto es así, al menos para el caso de ImageNet y CIFAR10, que contienen imágenes de dominios similares, y donde se ha podido comprobar la similitud entre los filtros generados a partir de uno u otro conjunto de datos.

5.8. Conclusiones

A través de los resultados obtenidos para los diferentes experimentos realizados, puede observarse de manera consistente la superioridad de los métodos de entrenamiento de filtros propuestos en cuanto a métodos de *inicialización* para la primera capa. Tanto los resultados obtenidos en términos de *accuracy* final de la red como la distancia coseno entre la configuración inicial y final de los filtros apoyan estas conclusiones. Estos resultados son consistentes para las cuatro configuraciones de arquitectura y conjunto de datos ensayadas.

Estos resultados, asimismo, permiten especular con la posibilidad de prescindir, en el marco de futuros trabajos, de GDBP para el entrenamiento de la primera capa de las redes convolucionales. A partir de los métodos de entrenamiento propuestos se parte de inicializaciones muy cercanas a mínimos locales. Este hecho resulta sumamente sugerente en cuanto a la posibilidad de obtener métodos de entrenamiento mucho más eficientes que GDBP para redes convolucionales.

Si bien no parece existir ningún obstáculo para implementar estos mismo métodos de entrenamiento en capas posteriores de la red, es necesario considerar que los experimentos se han realizado únicamente sobre la primera capa convolucional. Las conclusiones a las que se ha arribado deben considerarse válidas, a priori, solo para esta primera capa. Este punto es de central importancia y es fundamental dejar en claro que las extrapolaciones que puedan realizarse de estas conclusiones para el entrenamiento de filtros para capas subsiguientes no serán necesariamente válidas.

6. Conclusiones y futuros trabajos

Las redes neuronales artificiales han sido catalogadas en numerosas ocasiones como “cajas negras” en referencia a la dificultad que presentan para analizar las razones detrás de las inferencias que producen. Las redes convolucionales, como un caso particular de las redes neuronales, no escapan a esta misma categorización.

En las redes convolucionales, los parámetros aprendidos poseen una interpretabilidad visual, característica única, no presente en otros tipos de redes neuronales. Estos parámetros conforman conjuntos de filtros convolucionales que son aprendidos por la red durante su entrenamiento. La dificultad, en este caso, radica en poder dar interpretabilidad a estos conjuntos de filtros y sus productos parciales, a la salida de cada una de las capas sucesivas de la red. En definitiva, se trata de poder comprender los mecanismos internos detrás de las predicciones que realiza la red.

Adicionalmente a la problemática de la falta de interpretabilidad de las redes neuronales, el método por medio del cual son entrenadas, presenta también varias características que merecen especial atención.

Por una parte, el alto costo computacional de GDBP lo vuelve dependiente del uso de infraestructura dedicada como GPUs y TPUs, convirtiendo el entrenamiento de redes neuronales en una actividad que muy pocas organizaciones pueden realizar a gran escala. Naturalmente, esto tiene un impacto negativo en la democratización del acceso a estas tecnologías y un impacto ecológico no menor.

Adicionalmente, entrenar una red por medio de GDBP implica generalmente un arduo trabajo humano y el uso intensivo de recursos computacionales en la tarea de búsqueda de hiper parámetros óptimos.

A lo largo de este trabajo se ha desarrollado un análisis exploratorio sobre los filtros convolucionales de diversos modelos entrenados sobre ImageNet. La propuesta ha sido la de analizar estos filtros y compararlos entre sí, buscando repeticiones, patrones y regularidades que aportaran al entendimiento de las redes convolucionales en general, asimismo, como al tipo de soluciones a las que GDBP puede arribar.

Por otra parte, en este trabajo, se han presentado dos métodos alternativos a GDBP para el entrenamiento de redes convolucionales. Son propuestas que se basan en la distribución del conjunto de datos de entrada (un conjunto de imágenes) y buscan evitar el empleo de GDBP y sus implicancias negativas ya mencionadas.

6.1. Análisis exploratorio de filtros convolucionales en modelos pre entrenados

De acuerdo a los resultados arribados, se ha podido demostrar que la métrica de similitud entre filtros que mejor refleja la similitud entre mapas de características producto de la convolución de dichos filtros es la similitud coseno. Es esta métrica la que ha ofrecido emparejamientos entre filtros más cercana al emparejamiento de mapas de características. El contar con una métrica de similitud directa entre filtros, evita la necesidad de tener que operar sobre el producto de las convoluciones. A la vez, permite profundizar en la comprensión de las redes convolucionales, aportando a la definición de qué se entiende por filtros similares o disímiles.

Utilizando la métrica de similitud coseno, se han realizado agrupamientos de todo el conjunto de filtros pertenecientes a diferentes modelos pre entrenados. Se ha podido visualizar como filtros muy similares se presentan en diferentes modelos. Estos filtros son producto del entrenamiento que se ha ejercido sobre diferentes modelos de redes convolucionales, con diferentes arquitecturas, regímenes de entrenamiento y número de ciclos de entrenamiento. Sin embargo, se ha podido observar que muchos de estos filtros son aprendidos una y otra vez por los diferentes modelos.

Si bien los filtros de mayor similitud entre sí no son necesariamente idénticos, son lo suficientemente similares y se presentan en número suficiente como para poder descartar coincidencias fortuitas. Adicionalmente, debe considerarse que estos filtros no son de ninguna manera óptimos globales, sino simplemente una de infinitas soluciones a las que cada modelo pudiera haber arribado.

Esto se vuelve aún más claro cuando se considera la cantidad de filtros muertos (*dead kernels*) que son generados como parte de la solución. Estos filtros, al realizar la operación de convolución con cualquier imagen de entrada, obtienen mapas de características nulos o vacíos (sin ninguna información). Más aún, muchas veces son generados varios filtros redundantes en un mismo modelo y en una misma capa.

Existen muchísimos pares de filtros con muy alta similitud y estos pares están compuestos, en ocasiones, por filtros pertenecientes a un mismo modelo. Los mapas de características que puedan producir serán muy similares, de acuerdo a la aproximación para estimar cercanía que demostró la similitud coseno entre filtros. Esto implica la generación de características redundantes para alimentar la capa posterior.

Estos resultados, así como la presencia en cantidad de filtros muertos, -*equivalente a una columna con todos valores nulos para clasificadores de datos estructurados*-, permiten concluir que las soluciones a las que arriban los modelos no son de ningún modo óptimas en el sentido global. Este hecho permite especular con la posibilidad de encontrar nuevos métodos de entrenamiento, o técnicas complementarias a las actuales, que permitan arribar a soluciones más óptimas y eficientes, sobre un espacio de posibles soluciones más restringido.

Por otra parte, queda claro también, que en numerosas ocasiones, las mismas soluciones son obtenidas por diferentes modelos. Si se considera a la vez lo mencionado en el punto anterior, se puede afirmar que aún arribando a soluciones sumamente imperfectas, los modelos tienen una clara tendencia a adoptar una serie de filtros sobre un espacio de soluciones acotado.

Podría ser de sumo interés en futuros trabajos, para aportar a la comprensión de las redes convolucionales, intentar un análisis de las capas subsiguientes a la primera, retomando el mismo enfoque aquí presentado. Es de esperar que a mayor nivel de profundidad en la red, los modelos pre entrenados presenten filtros cada vez más disímiles entre sí. La intuición detrás de esta afirmación es el hecho de que al avanzar a través de las capas, cada una de ellas procesa mapas de características de entrada cada vez más disímiles entre diferentes modelos.

Si se considera la primera capa para diferentes modelos entrenados sobre un mismo conjunto de datos (ImageNet), se puede observar que todos los filtros operan sobre exactamente el mismo conjunto de datos de entrada. En la segunda capa, cada modelo deberá procesar datos de entrada relativamente diferentes, ya que los filtros de la primera capa son diferentes para cada modelo. Los datos que procesa la segunda capa no son más que las activaciones producidas por la primera. Siendo diferentes los filtros para los diferentes modelos, la segunda capa de cada uno de ellos estará procesando datos de entrada diferentes. Esto contrasta con la primera capa, donde todos los modelos procesan el mismo conjunto de datos de entrada. Esta divergencia iría en aumento a medida que se toman los filtros de capas cada vez más profundas. Esto es

presumiblemente así debido a la creciente divergencia de los mapas de activación entre diferentes modelos, a medida que se progresa hacia la salida de la red.

Una posibilidad para salvar esta dificultad podría ser la de reemplazar los filtros de la primera capa de todos los modelos por un conjunto genérico o tomado de un único modelo y luego hacer un breve reentrenamiento de las redes con ajuste fino (*fine tuning*). De este modo, la segunda capa de cada modelo estaría procesando como entrada un mismo conjunto de datos para todos los modelos. Este procedimiento podría ser replicado, en forma progresiva, a través de todas las capas de las redes.

Adicionalmente, puede ser de mucho interés, poder encontrar estructuras de filtros que se presenten de forma repetida para continuar con la búsqueda de un espacio de soluciones reducido. Si existen estructuras prototípicas sobre las que numerosos filtros terminan convergiendo luego del proceso de entrenamiento, estas estructuras podrían ser candidatas ideales para restringir de forma efectiva el espacio de soluciones, reduciendo los recursos necesarios para el entrenamiento así como la tendencia al sobre ajuste, sin impactar en el desempeño de la red.

6.2. Métodos de entrenamiento de filtros

En este trabajo se han propuesto dos métodos alternativos de entrenamiento para redes convolucionales. Utilizando el conjunto de datos de entrada de forma no supervisada se han construido filtros a partir de los algoritmos de K-Medias (KM) y Componentes Principales (CP). Se han desarrollado una serie de experimentos entrenando dos arquitecturas neuronales diferentes sobre dos conjuntos de datos con estos nuevos métodos. En todos los experimentos, se han contrastado los resultados obtenidos con el tradicional GDBP, para el empleo de las metodologías propuestas tanto como modos de entrenamiento, así como modos de inicialización de los filtros convolucionales.

En muchos casos, los resultados no han podido ser totalmente concluyentes en el sentido de proporcionar consistentemente un desempeño superior con un mismo método de entrenamiento o inicialización de los filtros. De cualquier modo, ambos métodos alternativos, con o sin entrenamiento posterior por medio de GDBP logran obtener de forma consistente mejores desempeños que el tradicional GDBP con una inicialización estándar. Sin embargo, no resulta claro cuál de ellos debería priorizarse en futuros trabajos. Resulta evidente la necesidad

de llevar a cabo un mayor número de entrenamientos, sobre mayor número de configuraciones para poder aclarar este punto.

El hecho de obtener resultados divergentes, en cuanto a la conveniencia o no de entrenar por GDBP posteriormente al entrenamiento de los filtros, lleva a pensar en el impacto del conjunto de datos y particularmente de la arquitectura de los modelos utilizados. Sería de sumo interés poder indagar en las razones detrás de la divergencia de resultados obtenidos por diferentes arquitecturas. Especialmente, poder determinar si existe alguna característica particular en una arquitectura convolucional que hace más o menos conveniente realizar el entrenamiento por GDBP una vez generados los filtros.

Por restricciones de tiempo y recursos computacionales, en este trabajo se han utilizado conjuntos de imágenes de menor tamaño, como CIFAR10 y Mnist Fashion, acelerando los tiempos de convergencia durante los entrenamientos. Sería ideal, sin embargo, en futuros trabajos y en función de poder llevar a cabo experimentos más concluyentes, repetir estos experimentos sobre arquitecturas consolidadas en la literatura (EfficientNet, entre ellas) sobre el conjunto de datos para el que han sido diseñadas (ImageNet).

Por otra parte, si bien los resultados obtenidos en este trabajo no pueden considerarse totalmente concluyentes por las razones previamente mencionadas, queda claro el carácter subóptimo del método de entrenamiento por GDBP y con métodos de inicialización estándar para los modelos y conjuntos de datos analizados. En todos los experimentos realizados, este enfoque nunca obtuvo los mejores desempeños. Por el contrario, los mismos se alcanzaron fruto de las técnicas de entrenamiento/inicialización aquí propuestas.

Por otra parte, se han realizado varios experimentos con la finalidad de determinar el impacto del tamaño muestral, el balance (o desbalance) de clases y la selección del conjunto de datos en el entrenamiento de filtros por los métodos propuestos. Para los tres tipos de experimentos, los resultados pueden considerarse concluyentes.

Resulta evidente, a juzgar por los resultados, que la selección de uno u otro conjunto de datos no tiene impacto sobre el conjunto de filtros generados. Esto es así, al menos para el caso de ImageNet y CIFAR10, donde se ha podido comprobar la similitud entre los filtros generados a partir de uno u otro conjunto de datos. Estos resultados son consistentes con los obtenidos en los experimentos de desbalance de clases. Al generar filtros por ambos métodos a partir de conjuntos de datos completamente desbalanceados, se obtienen conjuntos de filtros muy similares entre sí.

Respecto al tamaño muestral, los experimentos han demostrado que el entrenamiento de filtros para la primera capa convolucional puede bien realizarse con una muestra muy reducida del

conjunto de datos, obteniendo conjuntos de filtros prácticamente idénticos a los que se obtienen con el conjunto completo.

Todas estas conclusiones resultan válidas en el proceso de entrenamiento de filtros convolucionales para la primera capa convolucional. Extrapolaciones que puedan realizarse de estas conclusiones para el entrenamiento de filtros para capas subsiguientes no serán necesariamente válidas.

En función de los resultados aquí enunciados, resulta evidente que la dirección natural a tomar para futuros trabajos consiste en la aplicación de estos métodos de entrenamiento de filtros al resto de las capas de una arquitectura convolucional. Será a partir de la aplicación sistemática de una misma metodología a todas las capas de la red que se podrá realmente obtener resultados concluyentes, así como presentar una modalidad de entrenamiento alternativa y completa.

La técnica para la obtención de filtros para capas siguientes a la primera podrá consistir en la utilización de los mapas de características (*feature maps*) producidos por la capa inmediata anterior, como conjunto de datos de entrada. Es decir, se utilizarían los mapas de activación para la obtención de los parches de imágenes, a lo que luego se aplicaría uno u otro método de entrenamiento (Componentes Principales y/o K-Medias) para la obtención de los filtros convolucionales. Este proceso se repetiría a lo largo de toda la red, para todas las capas convolucionales. La última o últimas capas totalmente conectadas podrían entrenarse de modo habitual por medio de GDBP. Por supuesto, la cantidad de parámetros a entrenar será menor en varios órdenes de magnitud, permitiendo el entrenamiento sobre conjuntos de datos muy reducidos y evitando el sobreajuste.

Epílogo

Las redes neuronales se han convertido en la herramienta más importante para producir modelos de IA. Entre ellas, las redes convolucionales son el estado del arte en muchas tareas del campo de la visión artificial. En esta tesis, presenté un análisis y métodos de entrenamiento de filtros para la primera capa convolucional de una red que clasifica imágenes. Espero que los resultados obtenidos contribuyan a la comunidad, planteando caminos alternativos para el entrenamiento de redes convolucionales. Creo que los resultados obtenidos en los experimentos realizados, si bien no son completamente concluyentes, dejan asentado un precedente promisorio para nuevos trabajos de investigación en esta misma dirección.

La realización de esta tesis me sirvió mucho personalmente para cementar y expandir los conocimientos adquiridos durante el transcurso de la carrera. La búsqueda de métodos de entrenamiento de filtros me llevó a profundizar mis conocimientos en la arquitectura y funcionamiento de las redes convolucionales así como a consolidar mi entendimiento de algoritmos clásicos del aprendizaje automático. Asimismo, el análisis de filtros de modelos pre-entrenados me permitió expandir mis conocimientos en técnicas estadísticas y ampliar mi experiencia en el empleo de técnicas de agrupamiento no supervisado.

Bibliografía

[1] Christian Janiesch, Patrick Zschech, Kai Heinrich. “Machine learning and deep learning”. 2021. arXiv:2104.05314

[2] Giuseppe Bonaccorso. “Machine learning algorithms”. 2017. Packt Publishing. ISBN:978-1-78588-962-2

[3] Davoud Moulavi, Pablo A. Jaskowiak, Ricardo J. G. B. Campello, Arthur Zimek, Jörg Sander. “Density-Based Clustering Validation”. 2014. Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA, 2014

[4] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain”. 1958.

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel. “Backpropagation Applied to Handwritten Zip Code Recognition”. 1989. Neural Computation, vol. 1, no. 4, pp. 541-551, Dec. 1989. doi: https://doi.org/10.1162/neco.1989.1.4.541

[6] Robert Geirhos, David Janssen, Heiko Schutt, Jonas Rauber, Matthias Bethge, Felix Wichmann. “Comparing deep neural networks against humans: object recognition when the signal gets weaker”. 2018. arXiv:1706.06969v2

[7] Ian Goodfellow, Yoshua Bengio, Aaron Courvillecita. “Deep Learning”. 2016. MIT Press. http://www.deeplearningbook.org

[8] George Philipp, Dawn Song, Jaime G. Carbonell. “The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions”. 2017. arXiv:1712.05577.

[9] Xavier Glorot, Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. Journal of Machine Learning Research - Proceedings Track. 9. 249-256.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. 2015. arXiv:1502.01852

[11] Andrew M. Saxe, James L. McClelland, Surya Ganguli. “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks”. 2014. arXiv:1312.6120v3

[12] Hendrik J. Weideman. “Orthogonal Initialization in Convolutional Layers”. 2015. https://hjweide.github.io/orthogonal-initialization-in-convolutional-layers

[13] Wan-Duo Kurt Ma, J.P. Lewis, W. Bastiaan Kleijn. “The HSIC Bottleneck: Deep Learning without Back-Propagation”. 2019. arXiv:1908.01580v1

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, Ross Girshick. “Masked Autoencoders Are Scalable Vision Learners”. 2021. arXiv:2111.06377

[15] Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton. “A Simple Framework for Contrastive Learning of Visual Representations”. 2020. arXiv:2002.05709

[16] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, Armand Joulin. “Unsupervised Learning of Visual Features by Contrasting Cluster Assignments”. 2020. arXiv:2006.09882

[17] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, Thomas Brox. “Discriminative Unsupervised Feature Learning with Convolutional Neural Networks”. 2014. arXiv:1406.6909

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, Li Fei-Fei. “Imagenet: A large-scale hierarchical image database”. 2009. In 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255.

[19] Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton. “ImageNet Classification with Deep Convolutional Neural Network”. 2012. Communications of the ACM, 60(6), pp.84-90.

[20] Euijoon Ahn, Ashnil Kumar, Dagan Feng, Michael Fulham, Jinman Kim. “Unsupervised Feature Learning with K-means and An Ensemble of Deep Convolutional Neural Networks for Medical Image Classification”. 2019. arXiv:1906.03359

[21] Andrey Alekseev, Anatoly Bobe. “GaborNet: Gabor filters with learnable parameters in deep convolutional neural networks”. 2019. arXiv:1904/1904.13204

[22] Jun Bai, Yi Zeng, Yuxuan Zhao, Feifei Zhao. “Training a V1 Like Layer Using Gabor Filters in Convolutional Neural Networks”. 2019. International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1-8, doi: 10.1109/IJCNN.2019.8852439.

[23] D. H. Hubel y T. N. Wiesel. “Receptive fields, binocular interaction and functional architecture in the cat's visual cortex” 1962. The Journal of physiology, 160(1), 106–154. DOI 10.1113/jphysiol.1962.sp006837.

[24] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, Shan Carter. “Zoom In: An Introduction to Circuits”. 2020. 10.23915/distill.00024.001

[25] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, Yi Ma. “PCANet: A Simple Deep Learning Baseline for Image Classification?”. 2015. *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017-5032, Dec. 2015, doi: 10.1109/TIP.2015.2475625.

[26] Xu-Die Ren, Hao-Nan Guo, Guan-Chen He, Xu Xu, Chong Di, Sheng-Hong Li. “Convolutional Neural Network Based on Principal Component Analysis Initialization for Image Classification”. 2016. IEEE First International Conference on Data Science in Cyberspace (DSC), 2016, pp. 329-334, doi: 10.1109/DSC.2016.18.

[27] Foo Chong Soon, Hui Ying Khaw, Joon Huang Chuah, Jeevan Kanesan. “Semisupervised PCA Convolutional Network for Vehicle Type Classification”. 2020. IEEE Transactions on Vehicular Technology, vol. 69, no. 8, pp. 8267-8277, Aug. 2020, doi: 10.1109/TVT.2020.3000306.

[28] Caicai Zhang, Mei Mei, Mei Zhuolin, Junkang Zhang, Anyuan Deng et al. ”PLDANet: Reasonable Combination of PCA and LDA Convolutional Networks”. 2022. International Journal of Computers, Communications and Control, Oradea Vol. 17, Iss. 2, (Apr 2022). Doi: 10.15837/ijccc.2022.2.4541

[29] Longqing Sun, Yan Liu, Shuaihua Chen, Bing Luo, Yiyang Li, Chunhong Liu. “Pig Detection Algorithm Based on Sliding Windows and PCA Convolution”. 2019. IEEE Access, vol. 7, pp. 44229-44238, 2019, doi: 10.1109/ACCESS.2019.2907748.

[30] Lei Tian, Chunxiao Fan, Yue Ming, Yi Jin. “Stacked PCA Network (SPCANet): An effective deep learning for face recognition”. 2015. 1039-1043. 10.1109/ICDSP.2015.7252036.

[31] Hui Ying Khaw, Foo Chong Soon, Joon Huang Chuah, Chee Onn Chow. “Image noise types recognition using convolutional neural network with principal components analysis”. 2017. IET Image Processing, 11: 1238-1245. doi.org/10.1049/iet-ipr.2017.0374

[32] Adam Coates, Andrew Y. Ng. “Learning Feature Representations with K-means”. 2012. Publicado originalmente en G. Montavon, G. B. Orr, K.-R. M¨uller (Eds.), Neural Networks: Tricks of the Trade, 2nd edn, Springer LNCS 7700, 2012.

[33] Aruni RoyChowdhury, Prakhar Sharma, Erik Learned-Miller. “Reducing Duplicate Filters in Deep Neural Networks”. 2018.

[34] Sanghyun Son, Seungjun Nah, Kyoung Mu Lee. “Clustering Convolutional Kernels to Compress Deep Neural Networks”. 2018. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science(), vol 11212. Springer, Cham. DOI:10.1007/978-3-030-01237-3_14

[35] Wu Zhihong, Fuxiang Li, Yuan Zhu, Ke Lu, Mingzhi Wu, Changze Zhang. “A Filter Pruning Method of CNN Models Based on Feature Maps Clustering”. 2022. Applied Sciences 12, no. 9. 2022. https://doi.org/10.3390/app12094541

[36] Sanjukta Ghosh, Shashi K K Srinivasa, Peter Amon, Andreas Hutter, André Kaup. “Deep Network Pruning for Object Detection”. 2019. IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 2019, pp. 3915-3919, doi: 10.1109/ICIP.2019.8803505.

[37] Xiaohan Ding, Tianxiang Hao, Jungong Han, Yuchen Guo, Guiguang Ding. “Manipulating Identical Filter Redundancy for Efficient Pruning on Deep and Complicated CNN”. 2021. arXiv:2107.14444v1

[38] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. 2009. Technical Report TR-2009, University of Toronto, Toronto

[39] Han Xiao, Kashif Rasul, Roland Vollgraf. “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms”. 2017. arXiv:1708.07747v2.

[40] Christiane Fellbaum. “WordNet: An Electronic Lexical Database”. 1998. Cambridge, MA: MIT Press.

[41] Mingxing Tan, Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. 2020. arXiv:1905.11946v5

[42] Barret Zoph, Quoc V. Le. “Neural Architecture Search with Reinforcement Learning”. 2017. arXiv:1611.01578v2

[43] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba. “Learning Deep Features for Discriminative Localization”. 2015. arXiv:1512.04150v1.

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. 2018. arXiv:1801.04381.

[45] Sergey Zagoruyko, Nikos Komodakis. “Learning to Compare Image Patches via Convolutional Neural Networks”. 2015. arXiv:1504.03641

[46] Akihiro Imamura, Nana Arizumi. “Gabor filter incorporated CNN for compression”. 2021. arXiv:2110.15644v2

[47] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, Shan Carter. "Zoom In: An Introduction to Circuits". 2020. Distill.

[48] Emma Strubell, Ananya Ganesh, Andrew McCallum. “Energy and Policy Considerations for Modern Deep Learning Research”. AAAI, vol. 34, no. 09, pp. 13693-13696, Apr. 2020.

[49] Li, H., Xu, Z. Taylor, G. Studer, C. and Goldstein, T. “Visualizing the loss landscape of neural nets. Advances in neural information processing systems”. 2018

Apéndice

En este apéndice se exponen las arquitecturas de los modelos convolucionales utilizados en los experimentos de este trabajo. Tal como se mencionó en los capítulos 4 y 5, se utilizaron el modelo EfficientNet V2 Small y un modelo construido "a medida" denominado SimpleConv. Para ambos modelos, sus parámetros fueron inicializados por medio de Glorot Uniforme para todas sus capas, a excepción de la primera, para aquellos experimentos que implicaron la inicialización por otros medios explicitados.

Las arquitecturas de ambos modelos son las siguientes.

Arquitectura SimpleConv

La arquitectura SimpleConv está basada en el uso repetido de un bloque convolucional. Dicho bloque está compuesto por una capa convolucional, luego una de normalización por lote y finalmente una capa de activación ReLU. Para la capa convolucional, en todos los casos los filtros son de 3x3 píxeles, sin relleno y con zancada de 1.

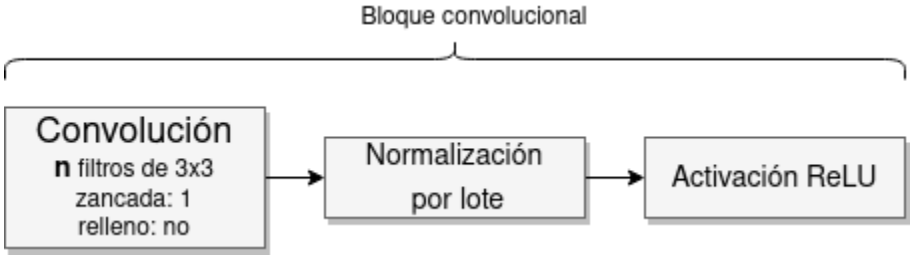


Fig A.1: Bloque convolucional de la arquitectura SimpleConv.

La cantidad de filtros utilizados varía en función de la etapa de la red de que se trate.

Existen 2 variantes de esta arquitectura, una utilizada para el conjunto de datos CIFAR10 y la otra para el conjunto de datos Mnist-Fashion. La única diferencia entre ambas es el empleo de 9 filtros para la primera capa para el caso de Mnist-Fashion y 27 unidades para el caso de CIFAR10. Esta diferencia responde a las limitantes ya mencionadas para el algoritmo de Componentes Principales de acuerdo a lo descrito en el capítulo 5.

La arquitectura completa del modelo SimpleConv, utilizado para el conjunto de datos Mnist-Fashion es el siguiente:

Etapa	Operación	Repeticiones	n filtros
1	Bloque Conv.	1	9
2	Bloque Conv.	4	128
3	Bloque Conv.	2	256
4	Bloque Conv.	2	512
Salida	Dropout 0.8	1	
Salida	Pooling promedio global	1	
Salida	Capa Densa	1	1024
Salida	Activación ReLU	1	
Salida	Dropout 0.8	1	
Salida	Capa Densa	1	10
Salida	Activación Softmax	1	

En el caso del conjunto de datos CIFAR10, la arquitectura es prácticamente idéntica, tal como se mencionó, con la única salvedad de que la primera etapa está constituída por un bloque convolucional de 27 filtros:

Etapa	Operación	Repeticiones	n filtros
1	Bloque Conv.	1	27
2	Bloque Conv.	4	128
3	Bloque Conv.	2	256
4	Bloque Conv.	2	512
Salida	Dropout 0.8	1	
Salida	Pooling promedio global	1	
Salida	Capa Densa	1	1024

Etapa	Operación	Repeticiones	n filtros
Salida	Activación ReLU	1	
Salida	Dropout 0.8	1	
Salida	Capa Densa	1	10
Salida	Activación Softmax	1	

Arquitectura EfficientNet

En el caso del segundo modelo utilizado a lo largo de los experimentos, EfficientNet, la arquitectura es idéntica a la definida por Tan et al [41] con la salvedad de que se ha incorporado una primera capa convolucional adicional. El propósito de esta incorporación ha sido el de adaptar las posibilidades de generación del algoritmo de Componentes Principales a la arquitectura. Tal como se mencionó en el capítulo 5, este algoritmo permite generar un máximo de filtros equivalente a la dimensionalidad de los datos de entrada. Para el caso de filtros de 3x3 píxeles, esto es 27 filtros para CIFAR10 (3 canales) y 9 para Mnist-Fashion (1 canal). Por este motivo, se han construido 2 arquitecturas prácticamente idénticas, a partir de EfficientNet, la primera de ellas adicionando una capa convolucional de 27 unidades (para CIFAR10) y la segunda adicionando una capa de 9 unidades (Mnist-Fashion).

Para el caso de EfficientNet, los autores utilizan 2 tipos de bloques residuales en varias etapas y con diferentes configuraciones a lo largo de la red. Estos bloques son denominados "MBConv" o bloque residual invertido y "Fused-MBConv" o bloque residual invertido fusionado. En este último se reemplazan las operaciones de convolución de expansión y convolución en profundidad presentes en "MBConv" por una convolución normal.

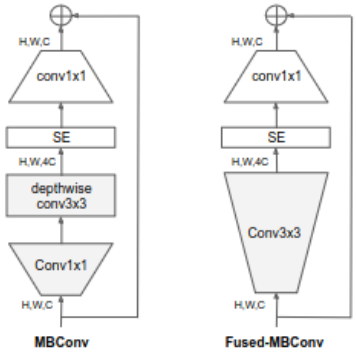


Fig A.2: Bloques MBConv y Fused-MBConv utilizados en EfficientNet.

La arquitectura de EfficientNet:

Etapa	Operación	zancada	canales	# capas
0	Convolución 3x3	2	24	1
1	Fused-MBConv 1, 3x3	1	24	2
2	Fused-MBConv 4, 3x3	2	48	4
3	Fused-MBConv 4, 3x3	2	64	4
4	MBConv 4, 3x3, SE 0.25	2	125	6
5	MBConv 6, 3x3, SE 0.25	1	160	9
6	MBConv 6, 3x3, SE 0.25	2	256	15
7	Conv 1x1 + Pooling + Densa	-	1280	1

Tal como se mencionó, en los experimentos realizados en este trabajo, se agregó a la arquitectura de EfficientNet un bloque convolucional al inicio de 9 filtros para el caso de Mnist-Fashion y de 27 filtros para el caso de CIFAR10. Finalmente se modificó la última capa densa del modelo (etapa 7), fijando en 10 la cantidad de unidades, en concordancia con la cantidad de clases de ambos conjuntos de datos.

Hiperparámetros utilizados en los entrenamientos

En la siguiente tabla se visualizan los hiperparámetros utilizados en los diferentes entrenamientos realizados en este trabajo

Arquitectura	Datos	Tasa de aprendizaje	Optimizador	Tamaño de lote
SimpleConv	CIFAR10	200 épocas a 1e-3 + 50 épocas a 1e-4	Adam (beta1=0.9, beta2=0.999)	512
SimpleConv	Mnist-Fashion	200 épocas a 1e-3 + 50 épocas a 1e-4	Adam (beta1=0.9, beta2=0.999)	512

Arquitectura	Datos	Tasa de aprendizaje	Optimizador	Tamaño de lote
EfficientNet	CIFAR10	50 épocas a 3e-4 + 30 épocas a 2e-4 + 70 épocas a 1e-4	Adam (beta1=0.9, beta2=0.999)	32
EfficientNet	Mnist-Fashion	20 épocas a 1e-3 + 200 épocas a 3e-4	Adam (beta1=0.9, beta2=0.999)	32