UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EXTRACCIÓN DE ATRIBUTOS VISUALES EN PRENDAS DE VESTIR A TRAVÉS DE NEURONAS OCULTAS DE MODELOS CONVOLUCIONALES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

ANDRÉS GARABED BALOIAN GACITÚA

PROFESOR GUÍA:
JOSÉ MANUEL SAAVEDRA RONDO

MIEMBROS DE LA COMISIÓN:
BENJAMÍN BUSTOS CÁRDENAS
PATRICIO INOSTROZA FAJARDIN

SANTIAGO DE CHILE
2021

Resumen

La visión computacional es un área de la computación en la que se construyen modelos para analizar e interpretar imágenes. Con la aparición de las redes neuronales profundas optimizadas para GPU, hace alrededor de una década, este campo ha experimentado un desarrollo vertiginoso y ha logrado resolver problemas que antes se pensaban imposibles de abordar. Sin embargo, de la mano con su gran poder predictivo, las redes neuronales profundas tienen la característica de ser poco interpretables. Una vez que se entrenan, las representaciones internas que generan a partir de las decenas de millones de parámetros que contienen se vuelven demasiado abstractas para ser comprendidas por los seres humanos. Por otro lado, entrenar modelos profundos es un proceso muy costoso tanto en datos como en capacidad computacional, por lo que es natural preguntarse si es posible reutilizar el conocimiento almacenado dentro de un modelo para aplicarlo sobre otros dominios. Así han nacido ramas como la Transferencia de aprendizaje, en la cual se utilizan los bloques convolucionales de un modelo entrenado para alimentar algún tipo de clasificador, el cual puede aprender a reconocer nuevas clases a partir de las características extraídas por los bloques anteriores. Sin embargo, las técnicas de este tipo requieren igualmente un proceso de entrenamiento (aunque de menor escala) y pueden fallar en utilizar la expresividad de las capas intermedias del modelo. En este trabajo se propone un método para extraer características de imágenes de prendas de vestir a partir de las neuronas ocultas de un modelo previamente entrenado para resolver un problema de clasificación en otro contexto. Además, se muestra que el método puede ser utilizado para clasificar eficazmente las prendas según su color y textura, teniendo la ventaja de poder incorporar nuevas clases sin requerir ningún proceso de entrenamiento adicional.

A mis amigos.

Tabla de Contenido

1. Introducción **1**
1.1. Antecedentes generales . . . 1
1.2. Motivación . . . 2
1.3. Estructura de la memoria . . . 4

2. Hipótesis y objetivos **5**
2.1. Hipótesis . . . 5

3. Marco teórico **7**
3.1. Aprendizaje profundo . . . 7
3.1.1. Redes neuronales convolucionales . . . 9
3.1.2. ResNet . . . 12
3.2. Comportamiento de un modelo convolucional . . . 14
3.2.1. Aprendizaje convergente . . . 14
3.2.2. Indexación de selectividad de las neuronas . . . 15
3.3. Visualización de modelos convolucionales . . . 17
3.4. Extracción de atributos . . . 19
3.5. Transferencia de aprendizaje . . . 21
3.6. Mapas de activación de clase . . . 21
3.7. Técnicas para visualizar datos con dimensiones altas . . . 22
3.7.1. Aproximación y proyección con colector uniforme para reducción de dimensión (UMAP) . . . 22
3.7.2. GridCut . . . 23
3.8. Métricas para evaluar clusterings . . . 24
3.8.1. Adjusted Rand Index . . . 24
3.8.2. Adjusted Mutual Information Score . . . 25
3.9. Resumen del capítulo . . . 25

4. Diseño experimental **27**
4.1. Arquitectura de las redes . . . 27
4.2. Conjuntos de datos . . . 27
4.3. Entrenamiento . . . 29
4.4. Entrenamiento del modelo de clasificación de ropa . . . 30
4.5. Modelo entrenado con ImageNet . . . 31
4.6. Resumen del capítulo . . . 31

5. Experimentos y resultados **32**
5.1. Visualización de mapas de características . . . 33

5.1.1. Class Activation Mapping . . . 33
5.1.1.1. Descripción del experimento . . . 33
5.1.1.2. Resultados . . . 34
5.1.1.3. Discusión de los resultados . . . 35
5.1.2. Mascaras binarias . . . 36
5.1.2.1. Descripción del experimento . . . 36
5.1.2.2. Resultados . . . 37
5.1.2.3. Discusión de los resultados . . . 38
5.1.3. Maximización de activación . . . 39
5.1.3.1. Descripción del experimento . . . 39
5.1.3.2. Resultados . . . 39
5.1.3.3. Discusión de los resultados . . . 40
5.2. Análisis de activaciones . . . 41
5.2.1. Promedio de activaciones por clase . . . 41
5.2.1.1. Descripción del experimento . . . 41
5.2.1.2. Activaciones promediadas para prendas anotadas por color . 43
5.2.1.3. Activaciones promediadas para prendas anotadas por textura 44
5.2.1.4. Discusión de los resultados . . . 45
5.2.2. Reducción de dimensión y clusterización . . . 46
5.2.2.1. Descripción del experimento . . . 46
5.2.2.2. Resultados . . . 47
5.2.2.3. Discusión de los resultados . . . 49
5.2.3. Visualización de activaciones de los conjuntos de datos usando GridCut 50
5.2.3.1. Descripción del experimento . . . 50
5.2.3.2. Resultados . . . 50
5.2.3.3. Discusión de los resultados . . . 51
5.2.4. Distribución de distancias entre pares de descriptores . . . 53
5.2.4.1. Descripción del experimento . . . 53
5.2.4.2. Resultados . . . 54
5.2.4.3. Discusión de los resultados . . . 55
5.3. Clasificación usando descriptores intermedios . . . 56
5.3.1. Clasificación usando distancia al descriptor promedio de cada clase . 57
5.3.1.1. Descripción del experimento . . . 57
5.3.1.2. Resultados . . . 57
5.3.1.3. Discusión de los resultados . . . 58
5.3.2. Clasificación usando K vecinos más cercanos . . . 59
5.3.2.1. Descripción del experimento . . . 59
5.3.2.2. Resultados . . . 59
5.3.2.3. Discusión de los resultados . . . 60
5.3.3. Clasificación en espacio reducido . . . 61
5.3.3.1. Descripción del experimento . . . 61
5.3.3.2. Resultados . . . 61
5.3.3.3. Discusión de los resultados . . . 62
5.3.4. Recuperación de imágenes similares . . . 63
5.3.4.1. Descripción del experimento . . . 63
5.3.4.2. Resultados generales . . . 64
5.3.4.3. Ejemplos de buen desempeño para colores . . . 64

5.3.4.4. Ejemplos de buen desempeño para texturas 65
5.3.4.5. Ejemplos de mal desempeño para colores 66
5.3.4.6. Ejemplos de mal desempeño para texturas 67
5.3.4.7. Discusión de los resultados 68
5.4. Resumen del capítulo 68

6. Conclusiones **70**

Bibliografía **72**

Índice de Tablas

5.1. Exactitud lograda en cada bloque según métrica de distancia para imágenes anotadas según color. 57
5.2. Exactitud lograda en cada bloque según métrica de distancia para imágenes anotadas según textura. 58
5.3. Exactitud promedio para imágenes anotadas por color (izquierda) y textura (derecha) 60
5.4. Exactitud lograda utilizando la salida del bloque 2 y el bloque 4 para color y textura, respectivamente, reduciendo los descriptores a espacios de diferentes dimensiones usando UMAP. 62

Índice de Ilustraciones

2.1. Activación hipotética de las neuronas frente al color del input. 5
3.1. Perceptrón multicapa con una capa oculta. 8
3.2. Conjunto de datos no separable linealmente. A la izquierda, la división del espacio lograda por una regresión lineal. A la derecha, sobre el mismo conjunto de datos, la división lograda por un perceptrón multicapa. 8
3.3. Arquitectura clásica de un modelo convolucional. [17] 9
3.4. A la izquierda, dos formas de ver cómo se aplica un kernel sobre un canal. A la derecha, los distintos kernels que componen un filtro. [17] 10
3.5. Resultado de aplicar un filtro convolucional. Una traslación en la señal de entrada se manifiesta como una traslación equivalente en la señal de salida. [18]. 10
3.6. Max-pooling y average-pooling, ambas con *stride* 2. [19] 11
3.7. Invarianza local introducida por una capa de reducción con *max-pooling*. [20] . 12
3.8. Arquitectura del bloque residual, componente clave de la arquitectura ResNet. 13
3.9. Bloque residual en detalle. [23] 13
3.10. Matrices de correlación para la capa **conv1**, en la que los píxeles más brillantes indican una mayor correlación. **a)** y **b)** muestran la correlación entre los canales del mismo modelo para el caso de dos redes de arquitecturas idénticas y entrenadas sobre el mismo conjunto. **c)** indica la correlación entre los canales de los dos modelos anteriores, mientras que en **d)** se realizó una permutación de los canales de la segunda red para aproximar el orden de los atributos de la primera red. [21] 15
3.11. Para cada canal, se muestra qué atributos de las imágenes del conjunto de validación causaron una mayor activación. A la izquierda se muestran las 8 parejas de canales con mayor correlación, mientras que en la derecha están las de correlación más baja. [21] 15
3.12. Visualización de 5 *Neuron Features* para neuronas de distintas capas de una red VGG-M, con las correspondientes 100 imágenes que causaron una mayor activación. [1] 16
3.13. En el primer gráfico, las secciones rojizas muestran el índice de selectividad de color para neuronas de distintas capas del modelo. En la segunda imagen, el color azul muestra una mayor selectividad de clase. [1] 17
3.14. Visualización de maximizaciones generadas para capas ocultas de *CaffeNet*. Las primeras cuatro filas muestran algunos canales para las 4 capas convolucionales del modelo, mientras que las dos últimas corresponden a neuronas de capas totalmente conectadas. [2] 18
3.15. Inputs obtenidos a través del método *Activation Maximization* para neuronas de la última capa de un modelo CaffeNet, junto a imágenes generadas por una red generadora GAN. 19

3.16. Atributos de diferentes prendas de vestir. 19
3.17. Entrenamiento progresivo del modelo. [3] 20
3.18. Exactitud lograda para atributos de Tops y Jeans. [3] 20
3.19. Las capas de un modelo convolucional se pueden reutilizar en otro modelo para extraer características. [22] 21
3.20. Mapas de activación de clase de un modelo convolucional para distintas clases. 22
3.21. [26] A la izquierda, se calcula el radio de cada punto dado el parámetro *n_neighbors*. A la derecha, las aristas representan la probabilidad de conexión entre los datos. 23
3.22. [27] Visualizaciones creadas con GridCut sobre distintos conjuntos de datos y su comparación con algunas técnicas de reducción de dimensión y otros métodos de visualización basados en grillas. 24
4.1. Imágenes del dataset Fashion Product Images. 28
4.2. Patrones, de izquierda a derecha: Agryle. Paisley, Pata de gallo y Polka. 29
4.3. Entrenamiento de la red ResNet50 sobre el conjunto Fashion Product Images de Kaggle. 30
4.4. Clases con mejores resultados según su índice f1. 31
5.1. Funcionamiento del método CAM. [10] 33
5.2. Resultados del método Class Activation Mapping para inputs de 3 clases diferentes. 34
5.3. Ejemplos del método de máscaras binarias. Las regiones visibles indican que la activación del canal en aquella zona fue superior al umbral. 36
5.4. Resultados del experimento de máscaras binarias, para algunos canales de la última capa convolucional del modelo. 37
5.5. Maximización de activación sobre 4 canales de la última capa convolucional del modelo ResNet50 entrenado sobre ImageNet. 39
5.6. Maximización de activación sobre 4 canales de la última capa convolucional de modelo ResNet50 entrenado sobre Kaggle. 40
5.7. Obtención de un descriptor intermedio mediante la aplicación de Global Average Pooling. 41
5.8. Activación promedio de los canales de salida del bloque 5 de ResNet50 para inputs de 6 clases diferentes del conjunto de imágenes prendas por color. . . . 43
5.9. Activación promedio de los canales de salida del bloque 5 de ResNet50 para inputs de 6 clases diferentes del conjunto de imágenes prendas por textura. . . 44
5.10. Espacio reducido con activaciones de la salida del segundo bloque convolucional del modelo entrenado con ImageNet para inputs del conjunto de imágenes anotadas por color. Además, se muestran las métricas adjusted Rand index y mutual-information score para la distribución de activaciones de cada bloque en el espacio reducido. 47
5.11. Espacio reducido con activaciones de la salida del cuarto bloque convolucional de ResNet50 del modelo entrenado con ImageNet para inputs del conjunto de imágenes anotadas por textura. Además, se muestran las métricas adjusted Rand index y mutual-information score para la distribución de activaciones de cada bloque en el espacio reducido. 48
5.12. Visualización de GridCut para el conjunto de prendas anotadas por color, utilizando las activaciones del bloque 2 del modelo con arquitectura ResNet50 entrenado con ImageNet. 50

5.13. Visualización de GridCut para el conjunto de prendas anotadas por textura, utilizando las activaciones del bloque 4 del modelo con arquitectura ResNet50 entrenado con ImageNet.. 51
5.14. Los histogramas muestran las distancias euclidianas entre 30.000 pares positivos y entre 30.000 pares negativos en distintos espacios reducidos con UMAP. . . . 54
5.15. Ejemplo de clusters bien definidos en los que se pueden formar varios pares positivos con distancia mayor a algunas distancias entre pares negativos 55
5.16. Validación cruzada con 5 splits. 56
5.17. Exactitud lograda por el método en distintos bloques de la red. A la izquierda, los resultados para imágenes anotadas por color. A la derecha, los resultados para imágenes anotadas por textura. 57
5.18. Exactitud desagregada por clases para distintos valores de k 59
5.19. Exactitud lograda usando la técnica del vecino más cercano dentro de espacios reducidos de distinta dimensión generados con UMAP. En ambos gráficos, el valor de exactitud para la dimensión más alta corresponde a la exactitud lograda en las dimensiones originales. 61

Capítulo 1

Introducción

1.1. Antecedentes generales

La visión computacional es un campo científico interdisciplinario que aborda cómo los computadores pueden obtener información de alto nivel a partir de imágenes. Algunas tareas comunes de esta disciplina son la clasificación de imágenes, la detección de objetos dentro de una imagen, la transformación de imagen a texto, el reconocimiento facial y la recuperación de imágenes, entre otras. A través de las técnicas de esta área, muchas tareas que hasta hace una o dos décadas eran imposibles de abordar sin ayuda de los humanos, hoy se pueden replicar y automatizar de manera digital utilizando computadores, en varios casos obteniendo incluso mejores resultados y en menor tiempo. De esta manera, en sectores como el diagnóstico médico, el transporte, la prevención del delito, el monitoreo de incendios forestales, la agricultura y tantos otros, la visión computacional ha logrado entrar de lleno a relevar humanos de sus funciones, trayendo consigo los múltiples beneficios que acarrea la automatización.

Los primeros pasos en esta ciencia se comenzaron a dar en los inicios de la década de 1960, cuando grupos universitarios abocados a la inteligencia artificial intentaban imitar el sistema visual humano para concederles a sus robots algún tipo de comportamiento inteligente. En los años 70 se realizaron estudios que sentaron las bases para muchos de los algoritmos que se utilizaron hasta los años 2000, incluyendo técnicas de extracción de bordes, etiquetado de líneas y representación de objetos, entre otros. En la siguientes décadas, la mayoría de los estudios se centraban en el análisis matemático de las imágenes, incluyendo los conceptos de escala-espacio, modelos de contorno y regularización.

Sin duda, el mayor de los saltos en el área vino de la mano de los avances en aprendizaje profundo (Deep Learning), los cuales permitieron abordar de una nueva manera los problemas de procesamiento e interpretación de imágenes. El aprendizaje profundo consiste en crear y entrenar modelos matemáticos utilizando redes neuronales profundas, las cuales cuentan con varias capas de variables que se encuentran conectadas entre sí por funciones de activación. Para producir resultados interesantes, estas redes son entrenadas a través de algún proceso de optimización, el cual, de forma iterativa, toma el resultado de una predicción, calcula su error y modifica los parámetros del modelo para reducir ese error.

Las redes neuronales son un concepto bastante antiguo en computación. Ya en 1969 se

logró demostrar cómo los perceptrones simples no servían para resolver problemas no lineales, lo que dio paso a los perceptrones multicapa. En 1986 se inventó el conocido método de propagación hacia atrás (Back-propagation), el cual permitía adaptar los pesos de las funciones del modelo dada una función de error, aunque este proceso resultaba lento y requería de una gran capacidad de computación. Sin embargo, los vertiginosos avances en hardware en aquella y las siguientes décadas, junto a la adaptación e implementación de arquitecturas convolucionales para ejecutarse en GPUs en el año 2012, hicieron viable, por primera vez, la utilización de redes profundas para resolver tareas de clasificación de imágenes y detección de objetos.

En el campo de la visión computacional, las redes neuronales convolucionales (CNN, por sus siglas en inglés) han demostrado tener un excelente desempeño y se consolidaron hace más de una década como una de las herramientas más utilizadas para desarrollar tareas y proyectos en los que se requiere clasificar imágenes. Por otro lado, dada la enorme cantidad de parámetros que contienen estas redes (típicamente decenas de millones) y la naturaleza automática de sus procesos de entrenamiento, las representaciones matemáticas que construyen internamente para procesar la información dejan de ser interpretables de manera directa para los seres humanos y se requiere de análisis posteriores para entender, apenas vagamente, qué está ocurriendo en su interior.

1.2. Motivación

Para entrenar redes neuronales convolucionales, normalmente se requiere de grandes cantidades de imágenes previamente etiquetadas. La cantidad definitiva de imágenes que se usan depende, en parte, de qué tan buenos son los resultados que se esperan obtener. Sin embargo, para entrenar, por ejemplo, un modelo de clasificación, lo mínimo es contar con decenas de miles de imágenes anotadas, mientras que algunos modelos llegan a entrenarse con varios millones de ejemplos. Si bien en internet existen muchos conjuntos de datos de acceso público que están disponibles, tales como ImageNet y PASCAL VOC, entre otros, es común que las imágenes de otras fuentes menos revisadas no tengan suficientes etiquetas o que estén derechamente mal etiquetadas. También ocurre con frecuencia que no todas las clases del modelo a entrenar están contenidas en un solo conjunto de datos, por lo que se tiene que recurrir a hacer una combinación de distintos conjuntos para obtener un conjunto adecuado. Finalmente, para problemas de clasificación medianamente específicos, a veces se tienen clases que no se encuentran en ningún conjunto de datos disponible. Esto hace que, en estos casos, los usuarios se vean obligados a construir sus propios datasets, lo que es un proceso muy costoso en tiempo y dinero.

Además de necesitar muchos datos, entrenar una red neuronal convolucional requiere de bastantes recursos computacionales. Las tarjetas gráficas (GPUs) han sido uno de los grandes motores que han impulsado la popularización del Deep Learning. Al ser idóneas para realizar operaciones matriciales en paralelo (que son la principal operación durante los procesos de entrenamiento), las GPUs permiten reducir los tiempos de entrenamiento en varios órdenes de magnitud. Sin embargo, estos componentes de hardware son costosos y no cualquier persona u organización tiene acceso a ellas. De todas formas, incluso utilizando GPUs, entrenar un modelo puede tomar varios días o incluso semanas.

Ya que entrenar un modelo puede ser lento y costoso, tanto por lo datos como por el hardware necesario, es lógico preguntarse si es que existen otras formas de entrenar un modelo que no requieran contar con un conjunto de datos tan grande o de componentes de hardware tan caros. Yendo más lejos, uno podría incluso pensar directamente en formas de evitar el proceso de entrenamiento. Hoy en día existen varias metodologías de transferencia de aprendizaje (Transfer learning), las cuales se centran en reutilizar los pesos de un cierto modelo previamente entrenado para implementar un nuevo modelo que pueda resolver un problema de clasificación diferente. La transferencia de aprendizaje se basa en la premisa de que, durante el proceso de entrenamiento, la red aprende a distinguir ciertos atributos de las imágenes que luego le son útiles para discriminar las distintas clases del problema. De esta forma, un modelo entrenado para clasificar autos, por ejemplo, podría reutilizarse para clasificar buses o camiones, dado que en los tres casos hay aspectos visuales comunes tales como las ruedas, ventanas, etc.

A pesar de que las técnicas de transferencia de aprendizaje permiten reutilizar el conocimiento de un modelo en otro, los pesos heredados del modelo anterior no sirven de nada sin que al menos alguna porción del nuevo modelo pase por un cierto proceso de aprendizaje que le permita reaprovecharlos. En algunos casos, la red completa se entrena sobre un cierto conjunto de datos reducido, mientras que en otros solo se modifican los parámetros de las partes más profundas de la red. Es decir, la transferencia de aprendizaje sirve más bien como un punto de partida que permite obtener buenos resultados con muchos menos datos que los necesarios para entrenar un modelo desde cero, pero sigue siendo necesario entrenar en alguna medida.

En los últimos años se han desarrollado varios métodos para analizar las activaciones de las neuronas. Con algunos de ellos es posible visualizar qué secciones de una imagen producen una mayor excitación en las neuronas de una determinada capa del modelo, mientras que con otras se pueden relacionar las distintas activaciones que producen un conjunto de imágenes diferentes que compartan algún atributo. Lo anterior da pie para entusiasmarse a diseccionar un modelo e investigar qué imágenes o qué atributos de ellas gatillan la activación de las neuronas y determinar si es que se pueden encontrar patrones.

El trabajo de esta memoria se centra en la búsqueda de una metodología para determinar fácilmente cómo se especializan las neuronas en una red convolucional previamente entrenada sobre algún conjunto de datos y utilizar partes de la red para extraer atributos visuales que no necesariamente fueron de interés durante el proceso de entrenamiento. La hipótesis que se quiere comprobar es que un modelo de clasificación (e.g. entrenado sobre ImageNet) aprende a extraer atributos visuales concretos a través una o varias neuronas internas. De esta forma, debe existir un conjunto de neuronas que se activen ante un mismo color, textura, forma u otros atributos que le hayan servido al modelo para discriminar las clases de salida.

De comprobarse la hipótesis mencionada, las neuronas de una red que originalmente fue entrenada para resolver un determinado problema de clasificación podrían ser aisladas y reutilizadas para extraer atributos particulares de las imágenes, evitando así tener que entrenar una CNN para detectar esos atributos.

Específicamente, se analizarán dos modelos. El primer modelo será uno entrenado para

clasificar prendas de vestir. Esta elección se debe principalmente a dos motivos. El primero de ellos es que las imágenes de ropa contienen atributos fácilmente distinguibles, tales como colores, texturas y patrones. Los atributos visuales de este tipo son además de gran importancia para los sitios de e-commerce que contienen catálogos de ropa, ya que, cada vez con mayor frecuencia, estas plataformas de retail incorporan funcionalidades de búsqueda por imagen, las cuales se basan en modelos convolucionales y requieren conjuntos de datos etiquetados. El segundo motivo para trabajar con prendas de vestir es que existen muchos conjuntos de datos disponibles para entrenar modelos de clasificación de ropa, lo que permite ahorrarse el tedioso proceso de crear un dataset y así pasar directamente a la fase de experimentación.

El segundo modelo que se analizará será uno entrenado para resolver un problema de clasificación general entrenado con datos de ImageNet, conjunto que incluye clases relacionadas con elementos de la naturaleza, objetos domésticos y otros. El objetivo de analizar este modelo es determinar si se puede aplicar el conocimiento recopilado por un modelo generalista en un problema específico (el de clasificar prendas de vestir).

1.3. Estructura de la memoria

El resto de este documento se divide en 5 capítulos.

- En el primero de ellos (capítulo 2), se presentan los objetivos de la memoria y la hipótesis que se desea comprobar.
- En el siguiente capítulo, correspondiente al marco teórico, se revisan los aspectos fundamentales del aprendizaje profundo y las redes convolucionales y se presentan brevemente algunos trabajos y técnicas relacionadas al tema de la memoria.
- En el capítulo 4, se describe la configuración de las redes neuronales utilizadas para hacer los experimentos, además de presentar los conjuntos de datos que se usaron.
- A continuación, en el capítulo 5, se presentan los experimentos que se realizaron en orden cronológico. Para cada experimento, se explica su motivación, los resultados obtenidos y una discusión sobre estos.
- El último capítulo corresponde a las conclusiones obtenidas luego de analizar los resultados junto a una reflexión sobre los hallazgos que se generaron y su relevancia.

Capítulo 2

Hipótesis y objetivos

2.1. Hipótesis

En un modelo entrenado, se espera que los mapas de características (que corresponden a los canales de cada capa) se activen a partir de ciertas características del input, como se ejemplifica en la figura 2.1. En particular, se espera que los mapas de características de la última capa convolucional detecten características relativamente complejas de la imagen, mientras que, en el caso de las capas más superficiales, se espera ver activaciones más marcadas frente a características más elementales. Por ejemplo, si una arquitectura contiene 5 bloques principales, cada uno de ellos conteniendo a la vez varias capas, se espera que en el bloque 1 y 2 existan capas con algunos canales especializados en detectar colores y formas básicas, mientras que en los bloques 4 y 5 podrían haber capas con canales especializados en detectar patrones y figuras más complejas.

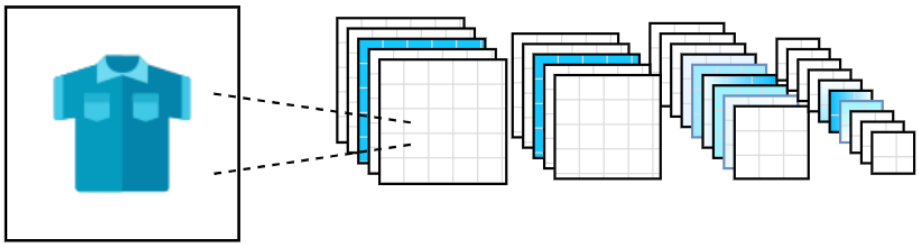

Figura 2.1: Activación hipotética de las neuronas frente al color del input.

A través del análisis de la activación de las neuronas frente a distintos datos de entrada, se cree que es posible atribuir a ciertas zonas de la red la extracción un determinado tipo de atributo, o bien, directamente un atributo concreto. El propósito de estos análisis es lograr reutilizar modelos originalmente entrenados en un problema general de clasificación (como ImageNet, por ejemplo) para extraer atributos específicos de las imágenes, a pesar de que los atributos buscados no hayan sido clases del modelo al momento de entrenar. De esta forma, una red entrenada para clasificar animales se podría utilizar para extraer características como el color, textura y forma de prendas de vestir. Esto permitiría, en algunos casos, ahorrarse la necesidad de crear o conseguir de conjuntos de datos anotados para entrenar un modelo, así como también serviría para evitar el costoso proceso de entrenamiento de los modelos

convolucionales cada vez que se agrega una clase nueva al problema.

Objetivo General

El objetivo de esta memoria es desarrollar un método de extracción de atributos visuales en imágenes de prendas de vestir basándose en los patrones de activación de las neuronas en las capas ocultas de modelos convolucionales. Logrando lo anterior, se podría producir un clasificador que no requeriría ser reentrenado cada vez que se agregue una clase nueva, sino que podría ser actualizado dinámicamente. Esto facilitaría la implementación y operación de buscadores basados en imágenes que usen clasificadores convolucionales, sobre todo en aplicaciones como el e-commerce, en donde es necesario agregar clases nuevas de manera frecuente.

Objetivos Específicos

1. Entrenar modelos convolucionales en forma supervisada para clasificación de prendas de vestir.
2. Entrenar modelos convolucionales en forma supervisada para un problema de clasificación general (e.g. ImageNet).
3. Crear un conjunto de datos para evaluar el comportamiento de los modelos ante diferentes tipos de patrones.
4. Visualizar la activación de las neuronas en distintas capas de los modelos.
5. Visualizar qué áreas de las imágenes producen una mayor activación en los canales de una determinada capa del modelo.
6. Determinar frente a qué características se activan las neuronas de los canales de una capa.
7. Encontrar o diseñar una metodología que permita identificar grupos de neuronas que respondan a ciertos tipos de patrones visuales.
8. Crear un dataset para evaluación que agrupe prendas de vestir en conjuntos definidos por atributos.
9. Utilizar los resultados para realizar tareas de clasificación y recuperación sobre conjuntos de datos de ropa con un modelo entrenado para otro problema.

Capítulo 3

Marco teórico

En este capítulo se hará un breve repaso sobre el aprendizaje profundo y otros trabajos relacionados con esta memoria. Además, se presentarán varias de las técnicas que más adelante serán utilizadas en los experimentos.

3.1. Aprendizaje profundo

El aprendizaje profundo (o Deep Learning) es una rama del aprendizaje automático que trabaja con redes neuronales artificiales para crear modelos que, a diferencia de modelos no profundos o superficiales, pueden aplicar muchas transformaciones en cascada a las señales de entrada. Esta larga sucesión de transformaciones permite que el modelo pueda crear representaciones de los datos con un alto nivel de abstracción, lo que hace posible generar modelos eficaces para aplicaciones complejas como la clasificación y segmentación de imágenes, el procesamiento del lenguaje natural, los sistemas de recomendación, la bioinformática, la predicción metereológica y el diseño de medicamentos, entre muchas otras.

En un modelo de aprendizaje profundo, las capas más superficiales (las más cercanas a la entrada del modelo) extraen características relativamente básicas del input, mientras que, a medida que avanzamos hacia capas más profundas, las neuronas de las capas se activan frente a atributos progresivamente más abstractos y complejos. Por ejemplo, en un modelo que analiza imágenes de automóviles, las primeras capas podrían activarse frente a elementos como bordes y colores específicos, mientras que las neuronas de capas más profundas podrían activarse frente a ruedas, ventanas y luces.

La arquitectura más básica de redes neuronales profundas es la del Perceptrón Multicapa (MLP), la cual consiste en una capa de entrada sucedida por una o varias capas ocultas y una capa de salida (figura 3.1). Cada capa consiste en un vector de neuronas, y cada neurona recibe como input los valores de todas las neuronas de la capa anterior.

Otro aspecto importante de los modelos de aprendizaje profundo es la utilización de unidades de procesamiento no lineal, las cuales son generalmente transformaciones que contienen funciones no lineales tales como la tangente hiperbólica, sigmoide o ReLU (Rectified Linear Unit). En 1969, en el libro Perceptrons [16], los autores M. L. Minsky y S. A. Papert demuestran que los perceptrones simples no sirven para aproximar problemas no lineales. Junto a la utilización de capas profundas, las transformaciones no lineales confieren a las redes neuronales profundas la habilidad de distinguir correctamente datos que no son separables

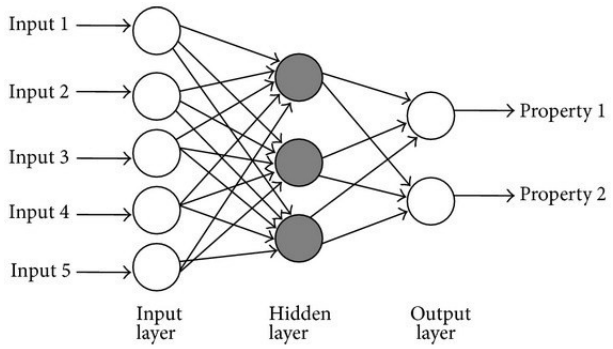

Figura 3.1: Perceptrón multicapa con una capa oculta.

mediante una función lineal [14] (ver figura 3.2). La razón por la que esta característica es tan importante es que, en el mundo físico, la mayoría de los fenómenos no siguen patrones lineales. Para que estos fenómenos se puedan aproximar con un modelo que entregue buenos resultados, es necesario contar con funciones no lineales como las mencionadas.

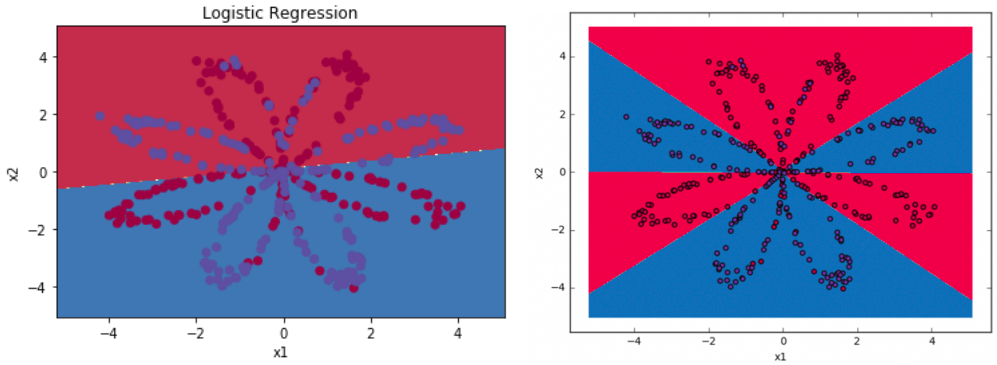

Figura 3.2: Conjunto de datos no separable linealmente. A la izquierda, la división del espacio lograda por una regresión lineal. A la derecha, sobre el mismo conjunto de datos, la división lograda por un perceptrón multicapa.

Las redes neuronales profundas pueden ser entrenadas tanto de manera supervisada como no supervisada. En el caso del entrenamiento supervisado, donde se encuentran la mayoría de las aplicaciones para estos modelos, las redes son entrenadas a partir de un conjunto de ejemplos etiquetados, los cuales se utilizan para medir el error del modelo y para hacer correcciones que disminuyan ese error. El algoritmo para modificar los parámetros de la red a partir del error de los ejemplos se llama *propagación hacia atrás* (o Backpropagation), y es una característica fundamental para el aprendizaje de las redes neuronales artificiales. Este algoritmo recibe su nombre debido a que el ajuste de parámetros, que en este caso son los pesos de las conexiones entre las neuronas, se realiza desde las capas más profundas hacia las capas más superficiales, contrario al sentido del procesamiento de los inputs. Durante la propagación hacia atrás, los pesos son ajustados según las derivadas parciales de la función de error, las cuales, para evaluarse, necesitan que las derivadas parciales de las capas posteriores se evalúen primero.

3.1.1. Redes neuronales convolucionales

Una red neuronal convolucional (CNN) es un tipo especial de red neuronal profunda utilizada principalmente para realizar tareas de visión computacional, aunque también tiene aplicaciones en áreas como el procesamiento de señales de audio y la clasificación de datos volumétricos, entre otras. La arquitectura de las redes neuronales convolucionales está inspirada en los patrones de conexión que existen entre las neuronas del córtex visual del cerebro [15], donde cada neurona responde a estímulos de una sección específica del campo visual y, además, las secciones que estimulan a cada neurona se solapan entre sí. Las capas convolucionales de un modelo contienen un determinado número de canales o mapas de características (figura 3.3), los cuales son matrices de números reales (también pueden verse como imágenes) y cada uno es es resultado de la aplicación de un filtro convolucional distinto sobre los canales generados en la capa anterior.

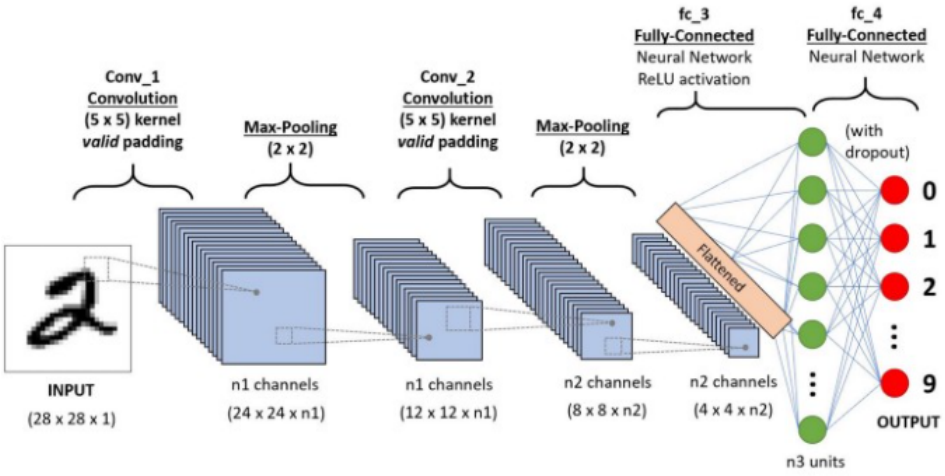

Figura 3.3: Arquitectura clásica de un modelo convolucional. [17]

Estos filtros son aprendidos durante el proceso de entrenamiento de la red y su función es extraer atributos que luego le sean útiles al modelo para hacer una predicción correcta. Los filtros convolucionales son un arreglo de pequeñas matrices de números reales llamadas kernels, frecuentemente de tamaño 3x3 o 5x5, las cuales se deslizan por sobre los canales de la capa de entrada, realizando una multiplicación elemento a elemento y luego sumando todos los resultados en un número que corresponde a un píxel del canal de salida correspondiente, como se ilustra en la figura 3.4. De esta forma, los pesos que se aprenden durante el aprendizaje de un modelo convolucional no son más que los valores contenidos en cada kernel de los filtros. Las neuronas de una red de este tipo corresponden, entonces, a los píxeles de cada canal de salida de una capa convolucional.

El hecho de que se tengan pequeños kernels que se deslizan sobre todo el canal de entrada en vez de conexiones directas entre todas las neuronas de los canales de entrada y de salida implica que la cantidad de pesos no crece con el tamaño del input, ya que todas las neuronas de un mismo canal comparten los mismos pesos. Gracias a esto, en comparación a los perceptrones multicapa, los modelos convolucionales tienen muchos menos parámetros, lo que es de suma importancia, dado que la cantidad de parámetros de un modelo está directamente relacionada con su tiempo de entrenamiento y con lo que demora en procesar los inputs.

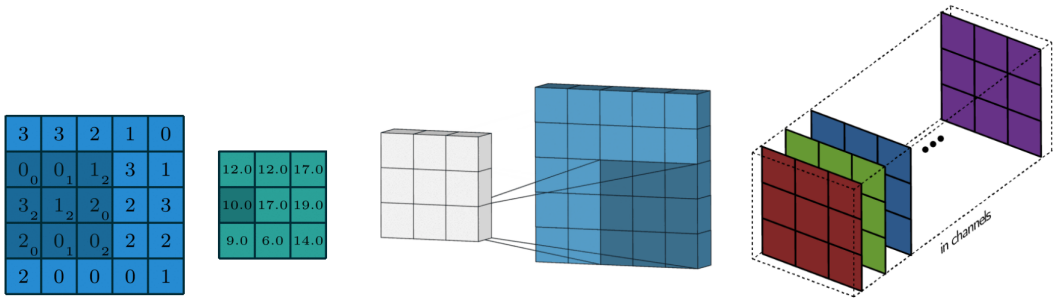

Figura 3.4: A la izquierda, dos formas de ver cómo se aplica un kernel sobre un canal. A la derecha, los distintos kernels que componen un filtro. [17]

Otro beneficio de la utilización de este tipo de convoluciones es que permiten lograr lo que se denomina como equivarianza traslacional. En este contexto, la equivarianza traslacional significa que, en una operación de convolución, la traslación espacial de los atributos del canal de entrada se manifiesta también como una traslación espacial equivalente en el canal de salida (ver figura 3.5). Supongamos que tenemos un modelo entrenado para detectar personas y que algún filtro de sus capas convolucionales es capaz de identificar la silueta de una persona. Si en una imagen de entrada la persona está a la derecha, en el canal de salida generado por el filtro detector de siluetas, las neuronas activadas estarán también a la derecha del mapa de características. Esta característica es fundamental, por ejemplo, para los modelos que realizan detección de objetos, ya que deben predecir no solo la clase sino la posición de los objetos dentro de una imagen.

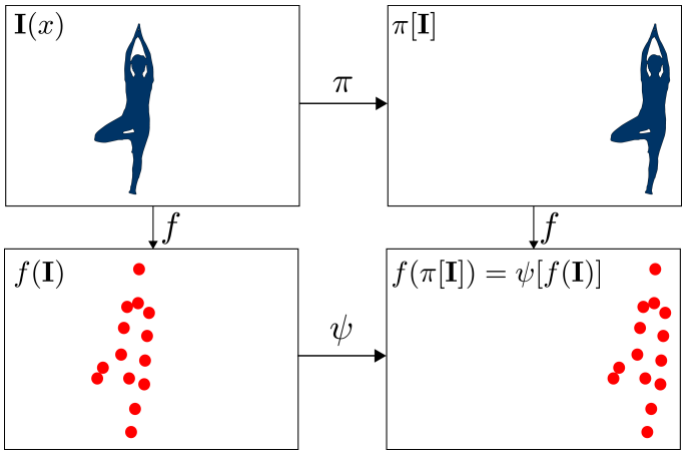

Figura 3.5: Resultado de aplicar un filtro convolucional. Una traslación en la señal de entrada se manifiesta como una traslación equivalente en la señal de salida. [18].

Además de las capas convolucionales, una arquitectura convolucional suele contener capas de reducción. En estas capas, las dimensiones de los canales de entrada son reducidas mediante alguna función como el promedio o el máximo. Al igual que en las capas de convolución, sobre cada canal de entrada se deslizan kernels, con la diferencia de que, en este caso, cada kernel solo agrega los píxeles sobre los que está operando, resumiendo su información en un solo píxel. Por ejemplo, si se tiene una capa de reducción con kernels de tamaño 2x2 y la función de agregación es el promedio, entonces, cada píxel de los canales de salida corresponderá al promedio de los 4 píxeles del canal de entrada operados por el kernel, como se muestra en la

figura 3.6.

Este proceso en el que se resume la información de los canales en canales más pequeños se denomina también como *pooling*. Las capas de pooling no contienen parámetros ni tampoco contribuyen a la extracción de características, sino que tienen otros dos propósitos principales. El primero de ellos es disminuir la complejidad del modelo, ya que, al reducir el tamaño de los canales, también se reduce el número de parámetros, lo que acelera significativamente los procesos de inferencia y entrenamiento. Por ejemplo, luego de una capa de reducción con kernels de tamaño 2x2, las dimensiones de los canales de entrada se reducen a la mitad o menos, dependiendo de cuánto se traslada el kernel entre cada paso. Esto último es un hiperparámetro que depende de cada arquitectura y se conoce como *stride* o paso de la capa de pooling.

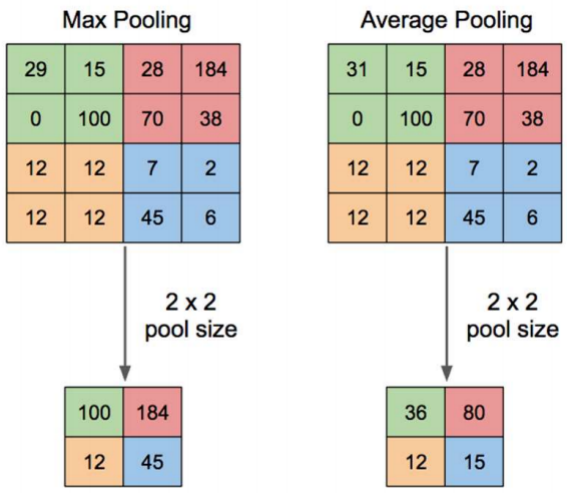

Figura 3.6: Max-pooling y average-pooling, ambas con *stride* 2. [19]

El otro objetivo de las capas de reducción es lograr la invarianza traslacional, lo que significa hacer al modelo robusto frente a cambios en la posición de los atributos de las imágenes. Al colapsar la información de los canales de entrada en canales de salida más pequeños, parte de la información espacial de los atributos se descarta, mientras que la intensidad de las activaciones se preserva. Esto produce un cierto grado de desensibilización frente a la posición de los atributos al atravesar una capa de reducción, comportamiento que se intensifica a medida que el modelo se hace más profundo y la información atraviesa más capas de reducción. En las capas finales del modelo, la información sobre la posición de los atributos con respecto a la imagen original es mínima si no inexistente, influyendo menos en la predicción (ver figura 3.7). De esta forma, si entrenamos un modelo con imágenes de aves en las que la mayoría o todas las aves se encontraban en la parte superior de la imagen, el modelo será de todas formas capaz de reconocer aves en las partes inferiores del input cuando se utilice para hacer predicciones.

A la salida de la capa convolucional más profunda de la arquitectura, los píxeles de los mapas de características resultantes suelen apilarse en un solo vector, el cual es seguido de una o más capas completamente conectadas, como en un perceptrón multicapa tradicional. En esta última etapa, la red se encarga de decidir una clase para la imagen de entrada. Así,

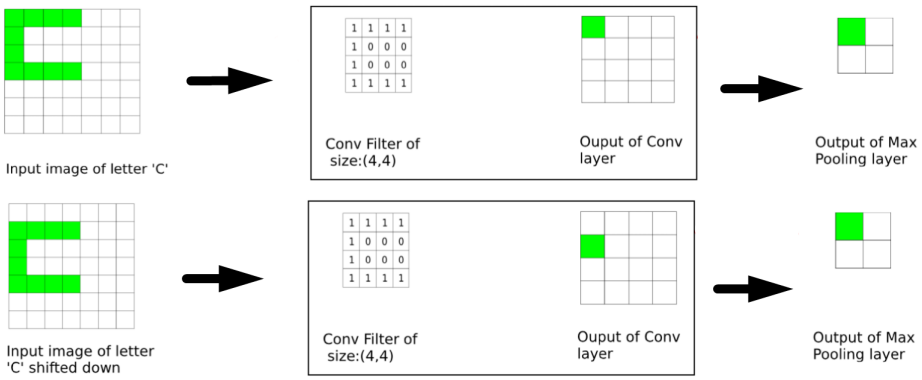

Figura 3.7: Invarianza local introducida por una capa de reducción con *max-pooling*. [20]

las arquitecturas de redes neuronales convolucionales suelen ser una sucesión alternada de capas convolucionales, capas de no-linearidad (como ReLU u otra) y capas de reducción, a veces con un pequeño perceptrón superficial de unas pocas capas totalmente conectadas en el extremo profundo del modelo convolucional.

Desde que debutó la arquitectura AlexNet en 2012 [5], la cual demostró la efectividad de los modelos convolucionales acelerados por GPU en el campo de la visión computacional, las CNN's se han mantenido como el estándar inequívoco para abordar problemas de reconocimiento visual. En el desafío ImageNet Large Scale Visual Recognition de 2012, Alexnet logró reducir el error del anterior campeón en más de diez puntos porcentuales. En el texto de su publicación se argumentaba el hecho de que la profundidad del modelo era fundamental para su desempeño, lo que comenzó una era de viajes hacia las profundidades del océano convolucional en búsqueda de mejores resultados. Desde entonces, año a año se publican nuevas arquitecturas que identifican y abordan diferentes problemas de las arquitecturas anteriores y, en algunos casos, obtienen mejores resultados y redefinen el estado del arte. Esto ciertamente ocurrió en 2015, cuando Microsoft Research dio a conocer la arquitectura ResNet [6], la cual permitía construir redes excepcionalmente profundas de hasta 150 o más capas, sin sacrificar exactitud. Luego, en 2016, ResNext [8] introdujo una nueva topología que añadía a ResNet ramificaciones paralelas, con lo que se obtuvieron mejores resultados. En mayo 2019, las redes Squeeze-and-excitation [7] se convirtieron en las nuevas campeonas en el ámbito de clasificación, hasta que en noviembre de ese mismo año fue destronada por EfficientNet [9], la cual se mantiene hasta hoy como la arquitectura de clasificación con mejor desempeño.

3.1.2. ResNet

ResNet es una arquitectura convolucional presentada en el año 2015 [6], en la que se aborda el problema del desvanecimiento de gradiente. Intuitivamente, se podría pensar que, tan solo aumentando la profundidad de las arquitecturas y entrenando los modelos con suficientes datos, se pueden obtener mejores resultados de predicción, dado que es sabido que el nivel de abstracción de las representaciones de los datos también aumenta con la profundidad. Sin embargo, esta afirmación no resultó ser cierta para las primeras arquitecturas convolucionales. En estas redes, se observaba que, a medida que añadían más capas convolucionales a la arquitectura, la magnitud del gradiente de las primeras capas convolucionales era muy pequeño

al realizar propagación hacia atrás. Por este motivo, las redes demasiado profundas tendían a no converger durante el entrenamiento y, como consecuencia, tenían malos resultados.

Los creadores de ResNet sostenían que esta caída en el gradiente de las capas superficiales se debía a que las redes planas, en las que las capas de la red están conectadas únicamente de manera secuencial, no eran buenas generando mapeos de identidad. Esto quiere decir que, en esas arquitecturas, las capas no logran aprender a replicar el resultado de la capa anterior, aún cuando la pérdida resultante pudiera ser menor. Para solucionar aquel problema, se introdujeron los bloques residuales (figuras 3.8 y 3.9), los que contienen conexiones «skip». Estas conexiones toman el input del bloque convolucional y lo suman al output del bloque. Con esto se facilita que la red pueda implementar el mapeo de identidad cuando es conveniente, sin aumentar la complejidad de la red.

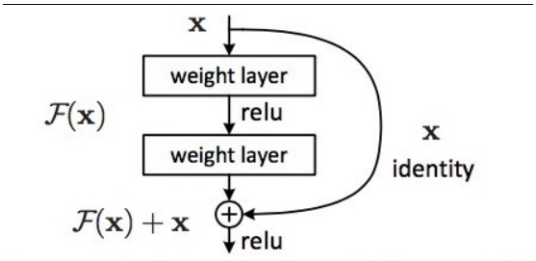

Figura 3.8: Arquitectura del bloque residual, componente clave de la arquitectura ResNet.

Existen varias versiones de esta arquitectura, tales como ResNet18, ResNet34, ResNet50 y ResNet 101 y otras. Entre estas versiones, lo que varía es, principalmente, la cantidad de capas convolucionales (y, por lo tanto, la profundidad de la arquitectura) y el tamaño de los bloques residuales. Por ejemplo, ResNet50 cuenta con 49 capas convolucionales, de las cuales 48 están agrupadas en 16 bloques residuales de 3 capas convolucionales cada uno. La capa faltante para alcanzar las 50 es una capa de max-pooling, la cual se encuentra justo después de la primera capa convolucional del modelo. Es importante mencionar que, después de cada capa convolucional, se normalizan las activaciones a través del método *batch-normalization* y luego se les aplica la función de activación ReLU.

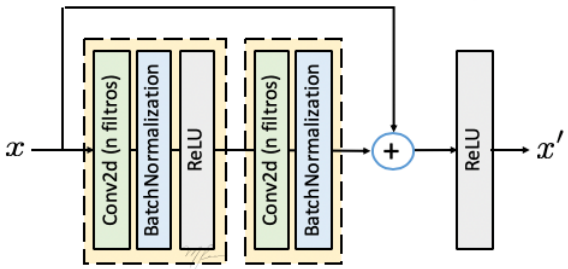

Figura 3.9: Bloque residual en detalle. [23]

3.2. Comportamiento de un modelo convolucional

Como se comentaba en las secciones anteriores, se ha visto que los modelos basados en redes convolucionales logran excelentes resultados clasificando imágenes a medida que aprenden características sobre los datos de entrada. Sin embargo, dada la cantidad de parámetros que contienen (usualmente millones) y la complejidad de su estructura interna, acaba siendo difícil entender precisamente cómo estos modelos generan los resultados que entregan una vez que han sido entrenados. Por ejemplo, si se quiere averiguar por qué un modelo clasificó una imagen de cierta manera, no existe una manera directa de determinar qué aspectos de la imagen incidieron en la decisión y cuáles no. Esto convierte a los modelos convolucionales (y, en general, a las redes neuronales profundas) esencialmente en cajas negras, en las que rara vez se presta atención a lo que ocurre entre el input y el output.

3.2.1. Aprendizaje convergente

En los últimos años se han publicado varios trabajos cuyos objetivos están relacionados con entender el comportamiento de las capas ocultas de CNN's y la forma en que los modelos convolucionales generan representaciones internas de la información. En esta área, la investigación se puede abordar tanto de forma teórica como también de manera empírica. En este último camino se encuentra el trabajo de Li et al[21], en el cual se analiza una pregunta bastante fundamental acerca de las redes neuronales. ¿Aprenden diferentes redes neuronales las mismas representaciones? Es decir, ¿convergen los modelos a la extracción de atributos similares cuando se entrenan sobre un mismo conjunto de datos? La duda se levanta principalmente dado el hecho de que, incluso inicializándose con valores distintos para sus parámetros, los modelos entrenados sobre un mismo conjunto de datos suelen lograr desempeños muy parecidos.

Para abordar esta incógnita, el trabajo propone utilizar varios métodos para evaluar la similitud de las representaciones creadas por modelos entrenados independientemente pero con la misma arquitectura y sobre el mismo conjunto de datos. Uno de ellos es alimentar el modelo con un conjunto de datos y analizar distintas métricas, tales como el promedio, la desviación estándar y la correlación entre los canales de las capas de dos modelos. El conjunto de datos utilizado en este caso fue *ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012*, el cual contiene cerca de un millón de imágenes y mil clases diferentes.

En el cuadro **d)** la figura 3.10, vemos que es posible encontrar, para varios de los canales de la capa «conv1» primer modelo, un canal en el segundo modelo con el cual hay una alta correlación entre sus activaciones. Este proceso de alinear los canales de ambos modelos permutando los canales del segundo consiste en, para cada canal del primer modelo, seleccionar aquel canal del segundo modelo con el que se tenga mayor correlación e igualar su índice con el del canal del primer modelo. Una vez realizado lo anterior, podemos evaluar los emparejamientos visualizando qué atributos de las imágenes del conjunto de validación causaron una mayor activación en cada canal de esa capa.

Los resultados de este método, visibles en la figura 3.11, muestran que definitivamente existen atributos que son comúnmente aprendidos por distintos modelos, incluso cuando se inicializan de manera diferente. En el caso de los canales con mayor correlación, la coincidencia de atributos es casi total. Sin embargo, también hay otros atributos que no son

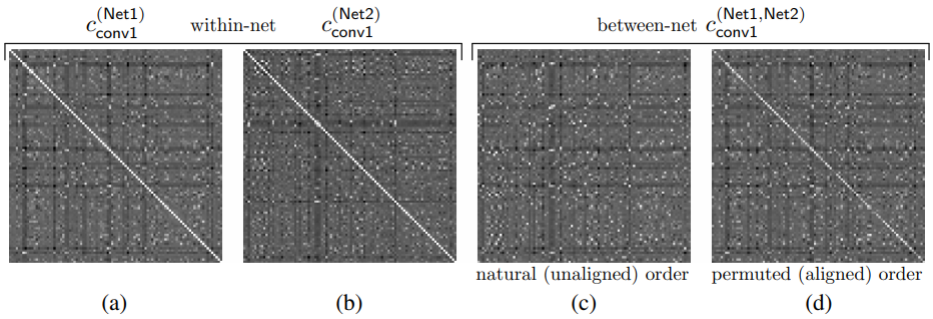

Figura 3.10: Matrices de correlación para la capa **conv1**, en la que los píxeles más brillantes indican una mayor correlación. **a)** y **b)** muestran la correlación entre los canales del mismo modelo para el caso de dos redes de arquitecturas idénticas y entrenadas sobre el mismo conjunto. **c)** indica la correlación entre los canales de los dos modelos anteriores, mientras que en **d)** se realizó una permutación de los canales de la segunda red para aproximar el orden de los atributos de la primera red. [21]

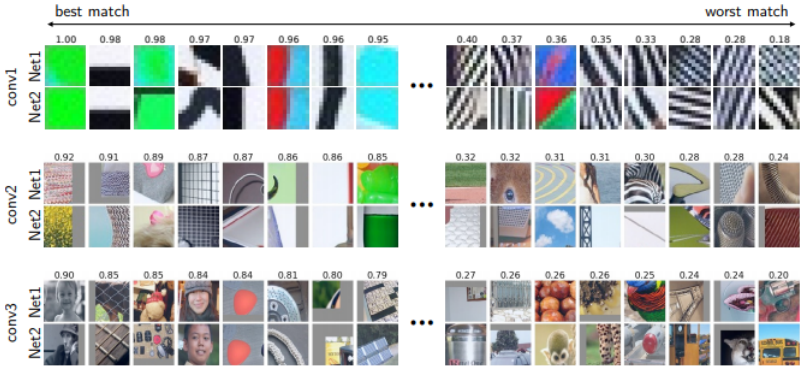

Figura 3.11: Para cada canal, se muestra qué atributos de las imágenes del conjunto de validación causaron una mayor activación. A la izquierda se muestran las 8 parejas de canales con mayor correlación, mientras que en la derecha están las de correlación más baja. [21]

consistentemente compartidos, lo que queda en evidencia al observar los canales con menor correlación.

3.2.2. Indexación de selectividad de las neuronas

En 2019, Ivet Rafegas et al. [1] publicaron un marco de trabajo que describe cómo es posible entender las representaciones internas de un modelo mediante cuantificar y caracterizar la activación de las neuronas en los canales de las capas ocultas. Para realizar los experimentos de la investigación, se decidió utilizar una arquitectura VGG-M. En primer lugar, para cada neurona analizada, se selecciona un conjunto de N imágenes que maximicen su activación. Luego, con este conjunto, se genera una sola imagen tomando el promedio ponderado de las N imágenes, teniendo mayor ponderación aquellas que produjeron una mayor activación.

A este promedio lo denominan *Neuron Feature* (atributo de neurona), el cual permite visualizar el el tipo de atributo que detecta una determinada neurona (figura 3.12). A partir

Figura 3.12: Visualización de 5 *Neuron Features* para neuronas de distintas capas de una red VGG-M, con las correspondientes 100 imágenes que causaron una mayor activación. [1]

de esta imagen, se calculan dos índices: índice de selectividad de color e índice de selectividad de clase. El primero de ellos se inspira en estudios sobre el cerebro humano, en los cuales se ha logrado cuantificar el nivel de activación de neuronas específicas cuando el observador es expuesto a un estímulo visual con un fuerte sesgo hacia un determinado color. Para el caso de las redes convolucionales, los investigadores proponen derivar este índice directamente de la cromaticidad de los atributos de neurona calculados anteriormente, de modo que mientras más inclinado estuviera hacia un color particular, mayor será su índice de selectividad de color. Por otro lado, el índice de selectividad de clase es obtenido mediante un proceso en el que se calcula la frecuencia relativa de cada clase en el conjunto de N imágenes que maximizan la activación de la neurona. Con estas frecuencias, se determina la diferencia entre cuántas clases son necesarias para cubrir una cierta cantidad de activación de la neurona y el total de imágenes del conjunto. De esta forma, si el resultado anterior es un número alto, significa que la neurona se activa fuertemente con ciertas clases específicas, mientras que si es un número pequeño, implica que la neurona se activa frente a muchas clases distintas y no está especializada en ninguna en particular.

Para los experimentos, se utilizó un subconjunto del dataset *ImageNet ILSVRC 2009*, consistente en 1.200.000 imágenes y 1000 clases. Al inspeccionar los valores de ambos índices en neuronas de distintas capas, como se muestra en la figura 3.13, se logró determinar que las neuronas de las capas más superficiales del modelo tienden a tener una mayor selectividad de color, encontrándose que hasta un 30 % de ellas estaban fuertemente especializas en discriminar este atributo. Por otro lado, a medida que se analizan neuronas más profundas, el índice de selectividad de clase aumenta considerablemente. En otras palabras, los resultados muestran que el reconocimiento de colores ocurre primordialmente en primeras capas del modelo, mientras que la detección de atributos más complejos, como siluetas, texturas y patrones característicos de los objetos, tiene lugar en las capas finales de la arquitectura.

Si bien este trabajo da luces sobre qué lugares de la red se pueden utilizar para la extracción de un tipo de atributo u otro, para lograr el objetivo de evitar procesos de entrenamiento, aún queda sin resolver el diseño de una metodología para aislar regiones de la red especializadas en atributos específicos y determinar cómo reutilizarlas para abordar otro problema de clasificación.

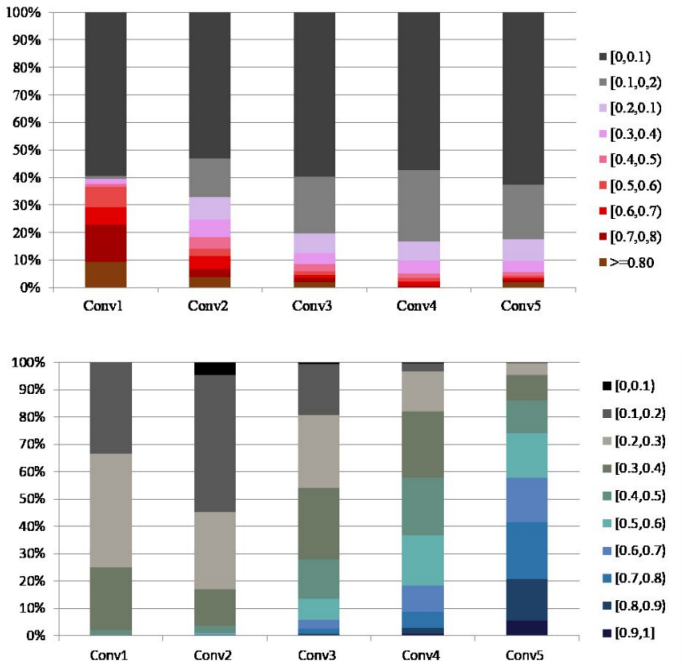

Figura 3.13: En el primer gráfico, las secciones rojizas muestran el índice de selectividad de color para neuronas de distintas capas del modelo. En la segunda imagen, el color azul muestra una mayor selectividad de clase. [1]

3.3. Visualización de modelos convolucionales

Otra área de interés para la realización de este trabajo es la visualización de modelos convolucionales. La visualización de una CNN consiste en traducir las representaciones internas de las redes en patrones visibles e interpretables por una persona. También se han logrado avances considerables en esta materia durante los últimos años, como se resume en la publicación de Zhuwei Qin en 2018 [2]. En este trabajo, se recopilan y analizan algunos de los métodos de visualización más populares en la actualidad.

El primer método que se describe es *Activation Maximization*, en el cual consiste en sintetizar un input cuyos píxeles maximicen la activación de las neuronas de los canales para una determinada capa (ver figura 3.14). La utilidad de hacer esto es que, tan solo inspeccionando visualmente este input artificial, se puede inferir qué características aprendieron a reconocer las neuronas en cuestión. Para producir este input, se comienza generando una imagen cuyos píxeles se inicializan con valores aleatorios y se evalúan las activaciones producidas en la capa bajo análisis. Luego, se calcula el gradiente para la activación de cada neurona con respecto a los píxeles del input, lo que se utiliza para modificar el valor de los píxeles a favor del gradiente. Así, si x^* es la imagen sintetizada, θ son los parámetros del modelo (pesos y sesgos) y a_{il} es el promedio de las activaciones para un canal i en la capa l, el algoritmo se pude ver de la siguiente forma:

$$x^* = \max_x a_{il}(\theta, x)$$

Lo anterior se convierte en un proceso iterativo, repitiéndose hasta que el input logre maximizar la activación de las neuronas.

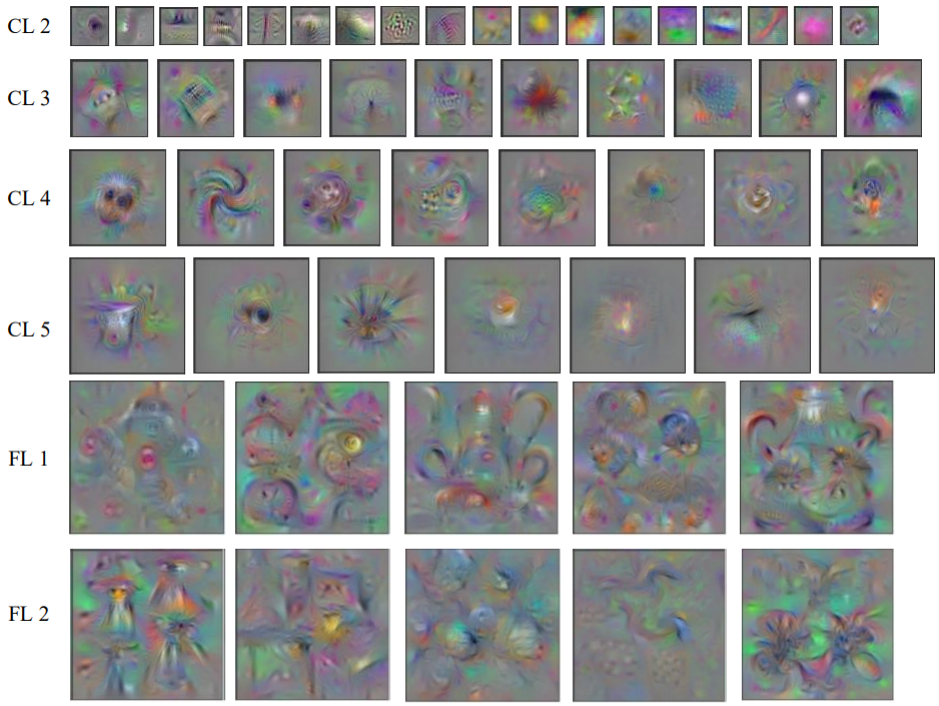

Figura 3.14: Visualización de maximizaciones generadas para capas ocultas de *CaffeNet*. Las primeras cuatro filas muestran algunos canales para las 4 capas convolucionales del modelo, mientras que las dos últimas corresponden a neuronas de capas totalmente conectadas. [2]

El problema con este método es que, a medida que las redes se vuelven más profundas, las visualizaciones generadas se vuelven más abstractas y difíciles de interpretar. Para producir imágenes más reconocibles, se ha demostrado que la utilización de métodos de regularización mejora considerablemente los resultados en este aspecto. Estos métodos introducen un parámetro de regularización en forma de sesgo, con lo que la fórmula para sintetizar la imagen se transforma en lo siguiente:

$$x^* = \max_x(a_{il}(\theta, x) - \lambda(x))$$

donde λ puede ser una función de desenfoque gaussiano, decaimiento $l2$ o eliminación de píxeles.

Otro enfoque para mejorar la interpretabilidad de este método es utilizar redes generativas adversarias (GAN). En vez de usar el regularizador $\lambda(x)$ para sesgar los ajustes del input artificial, la imagen es sintetizada por esta red, consiguiendo resultados mucho más realistas y fáciles de interpretar, como se muestra en la figura 3.15.

En la misma publicación, se analizan otros métodos de visualización, tales como *Network Inversion*, *Deconvolutional Neural Networks* y *Network Dissection*. Sin embargo, el método analizado en los párrafos anteriores (maximización de activación) es por lejos el más intuitivo de ellos. De todas formas, cada uno de estos métodos permite analizar cierto aspecto de la red de mejor manera. En general, estas técnicas sirven para analizar los modelos desde un punto de vista cualitativo y son más bien una herramienta para interpretar las representaciones que construyen las redes.

Figura 3.15: Inputs obtenidos a través del método *Activation Maximization* para neuronas de la última capa de un modelo CaffeNet, junto a imágenes generadas por una red generadora GAN.

3.4. Extracción de atributos

Clasificar atributos de prendas de vestir es un problema muy importante para el mundo del e-commerce. Extraer atributos de la ropa que visten los modelos en los catálogos (como los que se muestran en la figura 3.16) ha probado ser una tarea difícil incluso para los modelos convolucionales, principalmente por la baja calidad de los datos anotados que hay disponibles, los cuales, además, son muy engorrosos de anotar manualmente. En relación a este tema, existen trabajos como el de Sandeep Singh Adhikari et al. (2019) [3], en los que se buscan métodos para generar modelos robustos frente a datos mal etiquetados.

Figura 3.16: Atributos de diferentes prendas de vestir.

Particularmente, este trabajo propone un método progresivo de entrenamiento para modelos que clasifican atributos de prendas de vestir, diferente al entrenamiento clásico de un modelo multiclase. La idea de la propuesta es dividir la arquitectura en un modelo base y ramificaciones especializadas en clasificar cierto atributo de las prendas de vestir. Las ramificaciones son, a su vez, modelos convolucionales. Cada vez que se agrega una nueva ramificación, se entrenan la red base y la nueva rama de extremo a extremo, utilizando un conjunto de datos anotado específicamente para el atributo en cuestión. Durante este paso, los pesos de las demás ramas deben permanecer congelados. Una vez concluida una época, se entrenan durante una época todas las otras ramificaciones con los datos correspondientes a

sus atributos, pero utilizando una tasa de aprendizaje menor. Los pasos anteriores se repiten en un proceso que acaba cuando se han agregado todos los atributos que se deseen (figura 3.17).

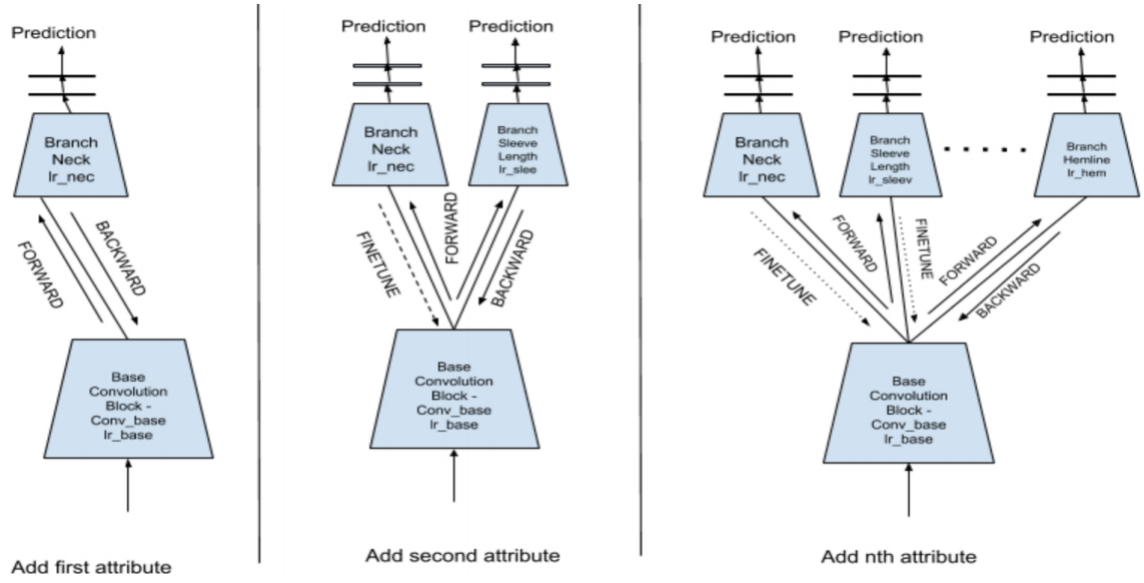

Figura 3.17: Entrenamiento progresivo del modelo. [3]

La ventaja de entrenar el modelo de esta forma es que, en el conjunto de datos de cada atributo, los otros atributos no necesariamente deben estar anotados. Es decir, la arquitectura se vuelve robusta frente a entrenamientos con conjuntos de datos débilmente etiquetados. Los resultados de los experimentos confirman la efectividad del método, en los cuales se ve reflejado claramente que el modelo resultante es al menos tan exacto como un modelo multiclase entrenado de una sola vez usando datos perfectamente anotados.

Model	Sleeve Style	Sleeve Length	Pattern	Neck	Overall
Individual	65.78	87.00	67.74	70.87	72.85
Multi-Label	65.96	88.65	64.80	70.46	72.47
Progressive	69.69	88.98	68.18	75.41	**75.57**

Model	Distress	Fade	Shade	Overall
Individual	82.60	76.25	78.58	79.14
Multi-Label	84.88	80.23	77.25	**80.79**
Progressive	84.38	79.68	77.00	80.35

Figura 3.18: Exactitud lograda para atributos de Tops y Jeans. [3]

3.5. Transferencia de aprendizaje

En el mundo del aprendizaje automático, los métodos y técnicas usadas para reutilizar modelos en dominios diferentes a los dominios para los que fueron originalmente entrenados, se agrupan bajo un concepto denominado Transferencia de aprendizaje *(Transfer Learning)*. Este concepto no se limita exclusivamente al Deep Learning, sino que también es aplicable en otras áreas del aprendizaje automático. Los beneficios de la transferencia de aprendizaje se podrían resumir principalmente en reducir los tiempos de entrenamiento al mismo tiempo que se reduce la cantidad de datos necesarios para lograr el buen desempeño de los modelos.

Una de las formas más comunes de realizar transferencia de aprendizaje para problemas de clasificación es tomar un modelo ya entrenado, quitarle las últimas capas (ver figura 3.19) y reemplazarlas con nuevas capas clasificadoras cuyas dimensiones de salida se ajusten a la cantidad de clases del nuevo problema. Una vez hecho lo anterior, se congelan los pesos de las capas del modelo original y se entrenan solo las nuevas. De esta forma, las nuevas capas solo aprenden a reorganizar los atributos extraídos por las capas convolucionales del modelo original y los utilizan para discriminar nuevas clases.

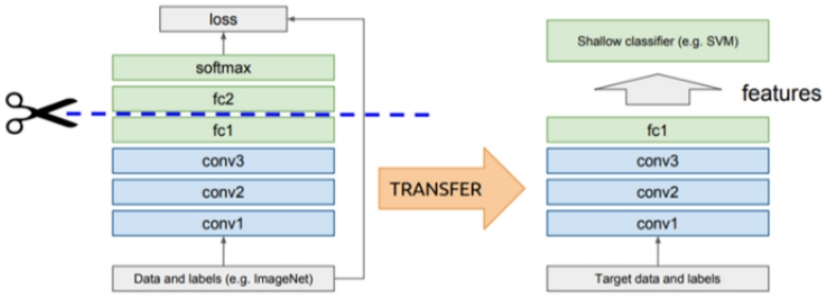

Figura 3.19: Las capas de un modelo convolucional se pueden reutilizar en otro modelo para extraer características. [22]

Es importante notar que para que esta técnica funcione, los dominios del problema original y del nuevo problema deben ser el mismo o, al menos, parecidos. Es decir, los inputs de ambos problemas deben tener características en común. Buenos ejemplos de lo anterior sería reutilizar un modelo para clasificar autos para construir un modelo que clasifique camiones u otro tipo de vehículos con ruedas.

Dado que para entrenar modelos de Deep Learning se suele requerir de una enorme cantidad de datos anotados, las técnicas de Transfer Learning son muy convenientes cuando no se cuenta con suficientes datos para entrenar un modelo desde cero. Mientras que entrenar un modelo desde cero es un proceso lento y que puede requerir cientos de miles de imágenes etiquetadas para alcanzar resultados de buena calidad, reutilizar parte de un modelo previamente entrenado puede reducir estos requerimientos de manera drástica.

3.6. Mapas de activación de clase

Una de las herramientas que más se han utilizado para analizar el comportamiento de las capas internas de los modelos convolucionales son los mapas de activación de clase (Class

Activation Mapping) [10]. A través de este método, es posible visualizar qué regiones de las imágenes son las que más influencian al modelo para asociarlas a cada clase, como se muestra en la figura 3.20. El método CAM está diseñado para funcionar sobre una CNN en la que, justo antes de la salida de la última capa convolucional, hay una capa de reducción que utiliza la función de promedio global (GAP) para reducir los mapas de características a un solo escalar. Para identificar las regiones que la red utiliza para asignar cada clase y generar los mapas de activación de clase, los pesos de la capa de salida son proyectados sobre los mapas de características de la última capa convolucional.

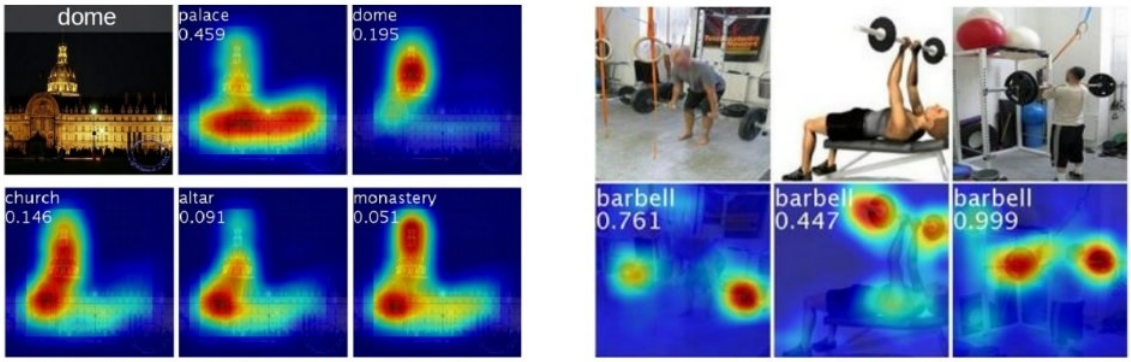

Figura 3.20: Mapas de activación de clase de un modelo convolucional para distintas clases.

Para producir estas visualizaciones, los mapas de características de la última capa convolucional del modelo son ponderados por los pesos que conectan la salida de cada canal con la neurona de salida correspondiente a una de las clases del modelo que se quiera analizar. La idea de esto es discriminar qué regiones de la imagen utilizó el modelo para hacer su predicción. Luego, los mapas de características ponderados son promediados en un solo mapa y se aplica algún método de upsampling, de modo que este nuevo mapa sea del mismo tamaño que el input. El resultado se puede visualizar como un mapa de calor, en el cual las zonas más calientes indican una mayor activación de las neuronas.

3.7. Técnicas para visualizar datos con dimensiones altas

3.7.1. Aproximación y proyección con colector uniforme para reducción de dimensión (UMAP)

Al trabajar con conjuntos de datos, frecuentemente es útil recurrir a alguna técnica de reducción de dimensión para generar visualizaciones que permitan entender de forma intuitiva cómo se distribuyen los datos del conjunto.

UMAP funciona creando un complejo simplicial difuso *(fuzzy simplicial complex)*, el cual se puede pensar como un grafo ponderado en el que las aristas representan la probabilidad de que dos puntos estén conectados. Para determinar esta probabilidad, se escoge un parámetro *n_neighbors* para el algoritmo y luego, para cada punto, se determina un radio tal que este

intersecte con el radio de exactamente *n_neighbors* vecinos más cercanos (figura 3.21). A continuación, para cada punto, la probabilidad de conexión con cada uno de sus *n_neighbors* vecinos más cercanos se calcula a partir de una función que decrece con la distancia, comenzando con probabilidad 1 para el vecino más próximo.

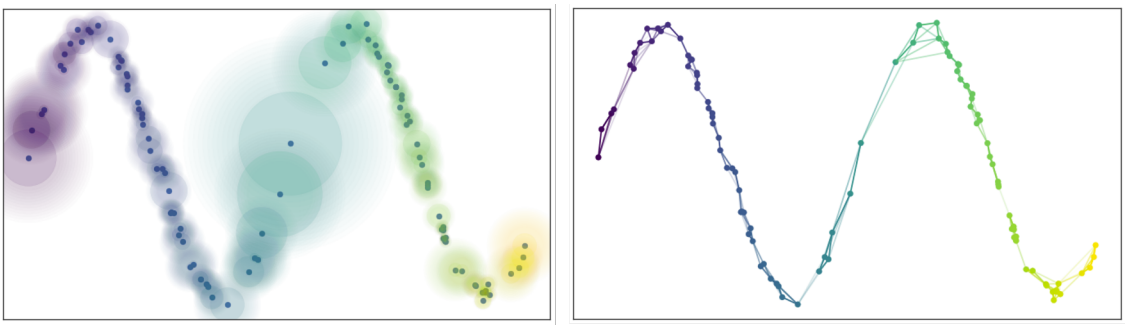

Figura 3.21: [26] A la izquierda, se calcula el radio de cada punto dado el parámetro *n_neighbors*. A la derecha, las aristas representan la probabilidad de conexión entre los datos.

Una vez se construye este grafo, UMAP optimiza iterativamente la topología de un grafo de dimensiones reducidas para parecerse lo más posible a la topología del grafo de dimensiones originales, proceso para el que utiliza la técnica de descenso de gradiente estocástico. Este proceso se realiza minimizando la entropía cruzada de los pesos de las conexiones entre los vértices de los grafos.

3.7.2. GridCut

Como se hablaba en la sección anterior, es común recurrir a técnicas de reducción de dimensión como UMAP cuando se quieren analizar conjuntos de datos que tienen muchas dimensiones. Sin embargo, a pesar de que estas técnicas logran capturar correctamente las diferencias y similitudes entre los datos de un conjunto, las visualizaciones creadas a partir de estos métodos suelen contener muchos elementos visuales solapados, elementos que pueden ser los puntos que representan la posición de los datos dentro del espacio reducido. Naturalmente, esto genera problemas de oclusión que pueden complicar los análisis exploratorios.

Para abordar este problema, los autores de GridCut [27] proponen un método de dos pasos para asignar la posición de cada elemento del espacio multidimensional a una celda de una grilla, a la vez que se intentan mantener las agrupaciones presentes en el espacio multidimensional.

Los resultados de este método producen visualizaciones en que los elementos visuales que representan a los datos no se ocluyen unos con otros, al mismo tiempo que se agrupan de forma muy similar a como lo están los datos en el espacio multidimensional.

En la figura 3.22 se puede ver una comparación entre los resultados obtenidos con este método y otras técnicas de reducción de dimensión y visualización.

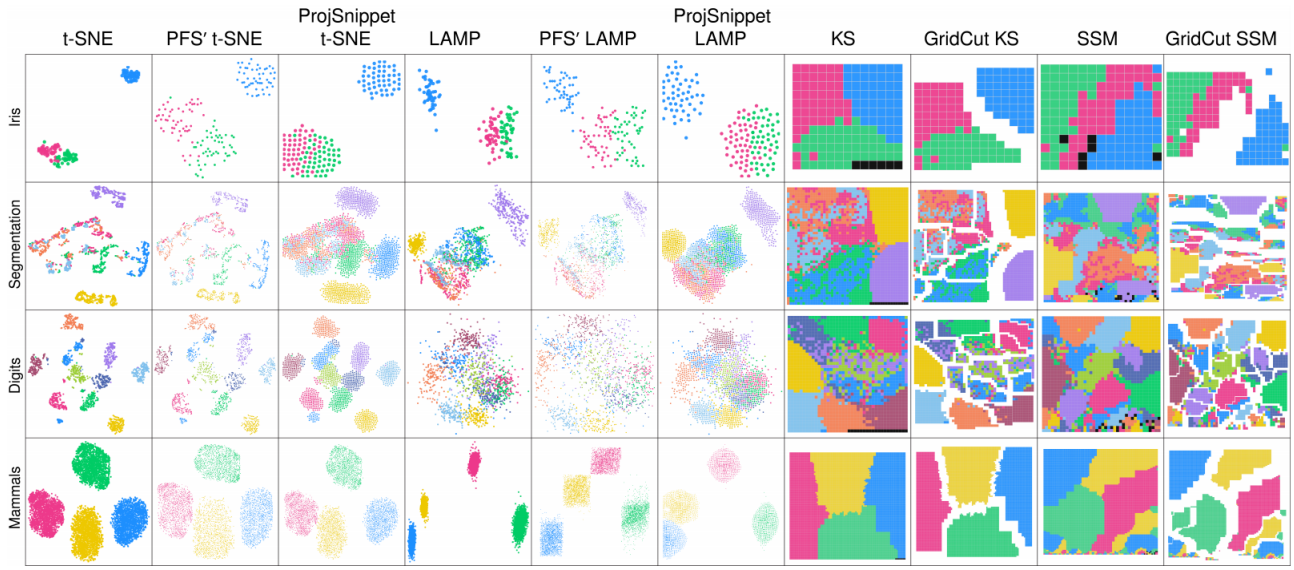

Figura 3.22: [27] Visualizaciones creadas con GridCut sobre distintos conjuntos de datos y su comparación con algunas técnicas de reducción de dimensión y otros métodos de visualización basados en grillas.

3.8. Métricas para evaluar clusterings

3.8.1. Adjusted Rand Index

Una métrica bien conocida para comparar clusterings es el índice Rand (*Rand Index*). Esta métrica toma en cuenta todos los posibles pares de datos del conjunto y mide la cantidad de coincidencias entre dos particiones del mismo conjunto. En este contexto, una coincidencia se da cuando ambos datos de un par de datos están en el mismo subconjunto tanto en la primera como en la segunda partición, o bien, están separados tanto en la primera como en la segunda partición. De esta manera, el índice Rand es la división entre la suma de coincidencias positivas (a) y negativas (b) y el total de pares posibles:

$$RI = \frac{a+b}{\binom{n}{2}}$$

El problema con esta métrica es que, a medida que crece el número de subconjuntos de las particiones, una agrupación al azar de los datos puede producir fácilmente valores altos del índice. Esto se debe a que, conforme aumenta la cantidad de subconjuntos, se vuelve más probable que un par de datos que no estén juntos en la primera partición tampoco lo estén en la segunda.

Para abordar este problema, se propone una versión ajustada del índice Rand (*Adjusted Rand Index*) que considera la probabilidad de agrupamiento por azar. Supongamos que se tiene la siguiente tabla de contingencia para comparar dos particiones:

Clase \Cluster	v_1	v_2	...	v_C	Sumas
u_1	n_{11}	n_{12}	...	n_{1C}	$n_{1.}$
u_2	n_{21}	n_{22}	...	n_{2C}	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
u_R	n_{R1}	n_{R2}	...	n_{RC}	$n_{R.}$
Sumas	$n_{.1}$	$n_{.2}$	...	$n_{.C}$	$n_{..} = n$

Dada esta tabla de contingencia, la versión ajustada del índice Rand se calcula de la siguiente forma:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] / \binom{n}{2}}$$

Donde n_{ij} es el número de objetos que pertenecen a la clase j y están en el cluster j. n_i y n_j son el número de objetos de la clase i y del cluster v_j, respectivamente. El término de la derecha en el numerador corresponde al valor esperado del índice para la partición, si esta fuera una partición aleatoria que sigue una distribución hipergeométrica. La fórmula anterior produce valores en el rango $[-1, 1]$, siendo 1 el valor que indica mayor similitud entre ambas particiones.

3.8.2. Adjusted Mutual Information Score

Otra métrica conocida para evaluar similitud entre clusterings es el puntaje de información mutua (*Mutual Information Score*). Este puntaje se calcula a partir de la siguiente fórmula:

$$MI(U, V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N\,|U_i \cap V_j|}{|U_i|\,|V_j|}$$

Donde U y V son las particiones U_i y V_j son sus respectivos clusters. Al igual que en el caso del índice Rand, esta métrica tiende a tener valores más altos cuando el número de clusters de la partición es mayor. La versión ajustada al azar de esta métrica, asumiendo una distribución hipergeométrica de particiones aleatorias, se calcula de la siguiente forma:

$$AMI(U, V) = \frac{MI(U, V) - E\{MI(U, V)\}}{\max\{H(U), H(V)\} - E\{MI(U, V)\}}$$

Nuevamente, un resultado igual a 1 indica que ambas particiones son idénticas.

3.9. Resumen del capítulo

El aprendizaje profundo es un área dentro del campo de la inteligencia artificial en la que se utilizan redes neuronales profundas para abordar problemas de visión computacional, reconocimiento del lenguaje natural y otros.

Las redes convolucionales son una de las arquitecturas de redes neuronales más usadas en

esta área, y serán el principal sujeto de investigación en los experimentos de esta memoria. Para analizar estas redes, se utilizarán algunas de las técnicas de visualización (como maximización de activación y mapas de activación de clases) y de análisis de activaciones (como indexación de selectividad de neuronas) que se presentaron en el capítulo, las que sirvieron como herramienta y punto de partida para la exploración y los experimentos que se describen más adelante.

Capítulo 4

Diseño experimental

En este capítulo se describirán aspectos sobre las redes neuronales que fueron utilizadas para los experimentos, así como también se hablará sobre los conjuntos de datos que se usaron para entrenar los modelos y analizar los resultados.

4.1. Arquitectura de las redes

Para realizar los experimentos y los análisis, se escogió usar una arquitectura ResNet50, la cual tiene 49 capas convolucionales agrupadas en 5 bloques. En los experimentos siguientes se hablará de estos bloques para indicar en qué lugar de la arquitectura se están analizando las activaciones, haciendo referencia a la capa convolucional de salida del bloque en cuestión.

Los motivos para elegir esta arquitectura (ResNet50) fueron, en primer lugar, porque se ha demostrado en múltiples trabajos que los modelos con esta arquitectura funcionan muy bien como extractores de características. De esta forma, es común encontrar arquitecturas de detección de objetos y segmentación semántica, por ejemplo, que utilizan la arquitectura ResNet50 para implementar su columna vertebral o *backbone*.

El otro motivo es que, al ser tan profunda, es posible analizar con mayor detalle cómo evolucionan las representaciones internas de los datos a lo largo de las capas que componen la arquitectura.

4.2. Conjuntos de datos

Dado el contexto del problema que se abordó, se buscó un conjunto de datos adecuado para entrenar un modelo para clasificar prendas de vestir según su tipo. Finalmente, se decidió utilizar el dataset Fashion Product Images [12], disponible en el sitio web Kaggle.

El dataset mencionado contiene alrededor de 44.000 imágenes de productos de moda (como los que se ven en la figura 4.1), incluyendo ropa, accesorios y otros como perfumes y maquillaje. Para cada imagen, existen varias etiquetas que clasifican al producto en categorías con distintos niveles de especificidad. Las 4 categorías son, de menos a más específica, «gender», «master-category», «subcategory» y «article type». Por ejemplo, una imagen de una polera puede tener asociadas simultáneamente etiquetas tales como «Men», «Apparel»,

«Topwear» y «T-shirts». Las imágenes son a color (3 canales) y tienen una resolución de 1080 x 1440 pixeles.

Figura 4.1: Imágenes del dataset Fashion Product Images.

Para realizar experimentos relacionados con determinar la capacidad de las neuronas para detectar características como colores y texturas, no se encontró un conjunto de datos público con suficientes clases relacionadas a las características de las prendas que se buscaban aislar. En particular, los conjuntos que se encontraron tenían pocas clases con anotaciones relacionadas a los patrones y texturas de las prendas, tenían muchas anotaciones inconsistentes (como se vio al analizar el dataset DeepFashion) o bien la presencia de estas características en las imágenes era insuficiente. Por estos motivos, se decidió construir dos pequeños conjuntos de datos, uno con prendas etiquetadas según su color y otro con etiquetas para los patrones de la tela.

El tamaño final de cada conjunto acabó siendo de 1000 imágenes, las cuales fueron obtenidas a partir de una combinación de otros conjuntos y también desde sitios web que contenían imágenes con derechos de uso libre. En cada conjunto, las imágenes fueron etiquetadas manualmente en 10 clases (100 imágenes por clase), las cuales dividen al primer conjunto según el color de la prenda y, en el caso del segundo conjunto, según el patrón de la tela. En la figura 4.2 se pueden observar algunos ejemplos de patrones de tela. Las clases de cada conjunto se resumen en las siguientes listas:

- Color
  - Amarillo
  - Azul
  - Café
  - Gris
  - Morado
  - Naranjo
  - Negro
  - Rojo
  - Rosado
  - Verde
- Patrón
  - Agryle
  - Cuadros
  - Flores
  - Lentejuelas
  - Leopardo
  - Paisley
  - Pata de gallo
  - Plano
  - Polka
  - Rayas

Figura 4.2: Patrones, de izquierda a derecha: Agryle. Paisley, Pata de gallo y Polka.

4.3. Entrenamiento

Para entrenar el modelo de clasificación de prendas de vestir, se utilizó la biblioteca Keras, la cual contiene funciones de alto nivel para trabajar con redes neuronales. Además, contiene implementaciones de distintas arquitecturas convolucionales, incluyendo ResNet50.

A su vez, se utilizó la versión de Keras que funciona sobre Tensorflow, la cual también es una librería para crear y entrenar redes neuronales en Python. La ventaja de usar Keras en vez de Tensorflow directamente es que Keras contiene muchos métodos auxiliares que simplifican aún más la implementación, el entrenamiento y la evaluación de modelos.

Como se mencionó en las secciones anteriores, se escogió utilizar una arquitectura ResNet50, cuyas 49 capas convolucionales suman un total de 23,534,592 parámetros entrenables. El conjunto de datos se dividió en 36.000 imágenes para la fase de entrenamiento y 8.000 para la validación. La categoría de clasificación que se utilizó fue «articleType», la cual divide al dataset en 144 clases. Dado que se contaba con una GPU, se utilizó la versión de Tensorflow compatible con este hardware (Tensorflow-GPU). El objetivo de aquello fue aprovechar la GPU para acelerar los procesos de entrenamiento y clasificación.

Las imágenes originales del conjunto de datos tenían una resolución de 1080 x 1440 píxeles, por lo que el proceso de entrenamiento resultó ser bastante lento, demorando alrededor de 20 horas por época y alcanzando solo un 40 % de exactitud después de la primera época. Como la idea era entrenar la red durante varias épocas y obtener mayores niveles de exactitud en un tiempo razonablemente pequeño, se reescalaron las imágenes a un tamaño de 390 x 520 píxeles.

Los parámetros de entrenamiento que se usaron son los siguientes:

- Tasa de aprendizaje: 0.01
- Tamaño de batch (lotes): 2

- Número de épocas: 10
- Optimizador: Adam
- Función de pérdida: Entropía cruzada

4.4. Entrenamiento del modelo de clasificación de ropa

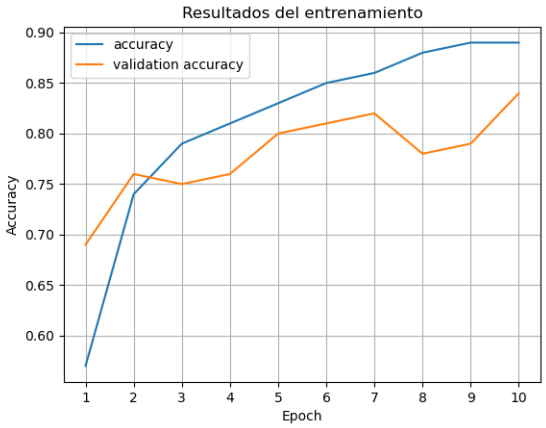

Figura 4.3: Entrenamiento de la red ResNet50 sobre el conjunto Fashion Product Images de Kaggle.

Como se puede ver en el gráfico de entrenamiento (figura 4.3), luego de 10 épocas, se logró una exactitud de un 84 % sobre el conjunto de validación. Esto quiere decir que, después de entrenar el modelo recorriendo completamente 10 veces el conjunto de entrenamiento, el modelo pudo clasificar correctamente un 84 % de las imágenes del conjunto de validación. El conjunto de validación no se utiliza para entrenar el modelo, sino que sólo para evaluar los resultados del entrenamiento. De esta forma se descarta que el modelo tan solo haya aprendido a memorizar las clases de cada input, comprobando así su capacidad de generalización.

En el gráfico, la pendiente de la curva que muestra la exactitud de validación sugiere que el modelo no ha terminado de converger. Es decir, si se ejecutaran épocas adicionales de entrenamiento, probablemente producirían una disminución del error.

Además, en la figura 4.4 se puede ver el desempeño del modelo para cada clase. Para medir el desempeño, se utilizaron las métricas de precisión y exhaustividad. En este contexto, la precisión de cada clase significa la proporción entre la cantidad de inputs de la clase a los que el modelo clasificó correctamente y la cantidad total de inputs a los que el modelo asignó tal clase. Por otro lado, la exhaustividad o *recall* de cada clase significa la proporción entre la cantidad de inputs de la clase que fueron clasificados correctamente y la cantidad de inputs correspondientes a la clase.

En el gráfico se pueden ver las 10 clases con mayor índice f1, el cual se calcula a partir del promedio armónico entre precisión y exhaustividad *(recall)*. Este índice suele utilizarse para el desempeño general del modelo para cada clase del problema.

Figura 4.4: Clases con mejores resultados según su índice f1.

Las métricas resultantes muestran que, para varias clases del conjunto de datos, el modelo fue capaz de predecir correctamente su clase en más del 95 % de los casos. Sin embargo, la medición de exactitud después de la última época de entrenamiento indicó un valor de 84 %, por lo que deben haber varias clases en las que el modelo tuvo un mal desempeño.

4.5. Modelo entrenado con ImageNet

Luego de analizar los resultados de los primeros experimentos (5.1.1, 5.1.2 y 5.1.3), se determinó que el modelo entrenado para clasificar prendas de vestir no era adecuado para buscar canales y neuronas especializadas en texturas y colores. Como consecuencia, se optó por utilizar un segundo modelo con la misma arquitectura que el modelo anterior (ResNet50), pero esta vez entrenado sobre ImageNet.

En este caso, no fue necesario entrenar el modelo, ya que Keras permite crear instancias de ResNet50 y otras arquitecturas inicializadas con los pesos resultantes de haberlas entrenado sobre ImageNet. Este modelo se usó desde el experimento 5.1.3 en adelante.

4.6. Resumen del capítulo

Para realizar los experimentos de esta memoria se decidió utilizar la arquitectura convolucional ResNet50, dada su buena reputación en aplicaciones de extracción de atributos. Se prepararon dos modelos basados en esta arquitectura, uno entrenado con el conjunto de datos Fashion Product Images de Kaggle y otro preentrenado con ImageNet. Además, se construyeron dos pequeños conjuntos de datos para evaluar la capacidad de los modelos de discriminar colores y texturas de prendas de vestir.

Capítulo 5

Experimentos y resultados

En este capítulo se describirán varios experimentos agrupados en tres grupos:

- Visualización de mapas de características
- Análisis de activaciones
- Clasificación usando descriptores intermedios

El primer grupo corresponde a experimentos realizados con el objetivo de visualizar las activaciones en mapas de características de distintos bloques de la red. El objetivo de estos experimentos es realizar un primer análisis cualitativo del comportamiento interno de la red. En el segundo grupo se encuentran experimentos realizados para entender cómo se diferencian las activaciones de la red y determinar qué tan distintivas son respecto a cada clase de los conjuntos de datos utilizados. Finalmente, el tercer grupo contiene experimentos en los que se busca evaluar la capacidad de los bloques intermedios para realizar tareas de clasificación.

Para cada experimento se realizará una descripción de qué se hizo y con qué propósito, se mostrarán los resultados obtenidos y se terminará con la discusión de estos últimos. Es importante mencionar que, en los experimentos 5.1.1 y 5.1.2 se utilizó el modelo entrenado con el conjunto de datos Fashion Product Images de Kaggle. El experimento 5.1.3 se repitió para ese mismo modelo y también para el modelo entrenado con ImageNet, mientras que desde el experimento 5.1.4 en adelante se usó solo el modelo entrenado con ImageNet.

5.1. Visualización de mapas de características

5.1.1. Class Activation Mapping

5.1.1.1. Descripción del experimento

Una vez entrenado el modelo con el conjunto de datos Fashion Product Images de Kaggle, se comenzaron a hacer pruebas con el método Class Activation Mapping, con el objetivo de observar la activación de los mapas de características de la última capa convolucional del modelo. Para esto, se utilizó una implementación para TensorFlow disponible en GitHub [11]. En el código, dada una predicción del modelo, se selecciona la neurona correspondiente a la clase predicha y se obtienen los pesos que utilizaron para ponderar sus inputs. Luego se utilizan esos pesos para ponderar todos los mapas de características de la última capa convolucional. Una vez ponderados, estos mapas se suman en uno solo para luego producir un mapa de calor, en el que se pueden discriminar las regiones de la imagen que produjeron una mayor activación (figura 5.1). Estas regiones se pueden interpretar como aquellas en las que el modelo puso mayor atención para realizar la clasificación.

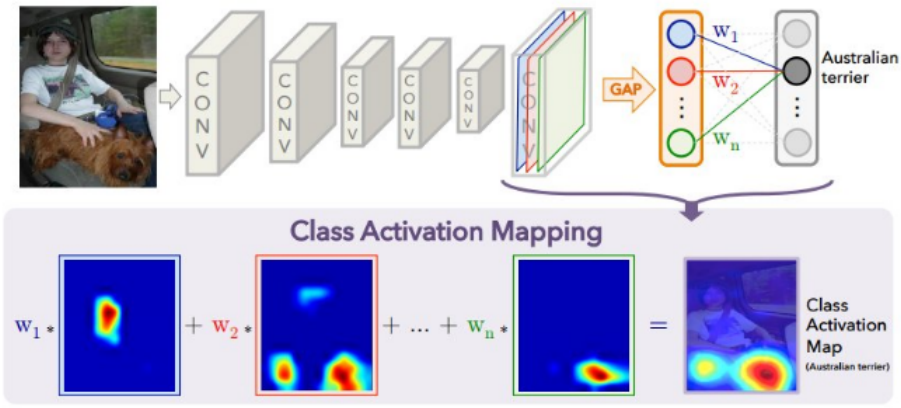

Figura 5.1: Funcionamiento del método CAM. [10]

5.1.1.2. Resultados

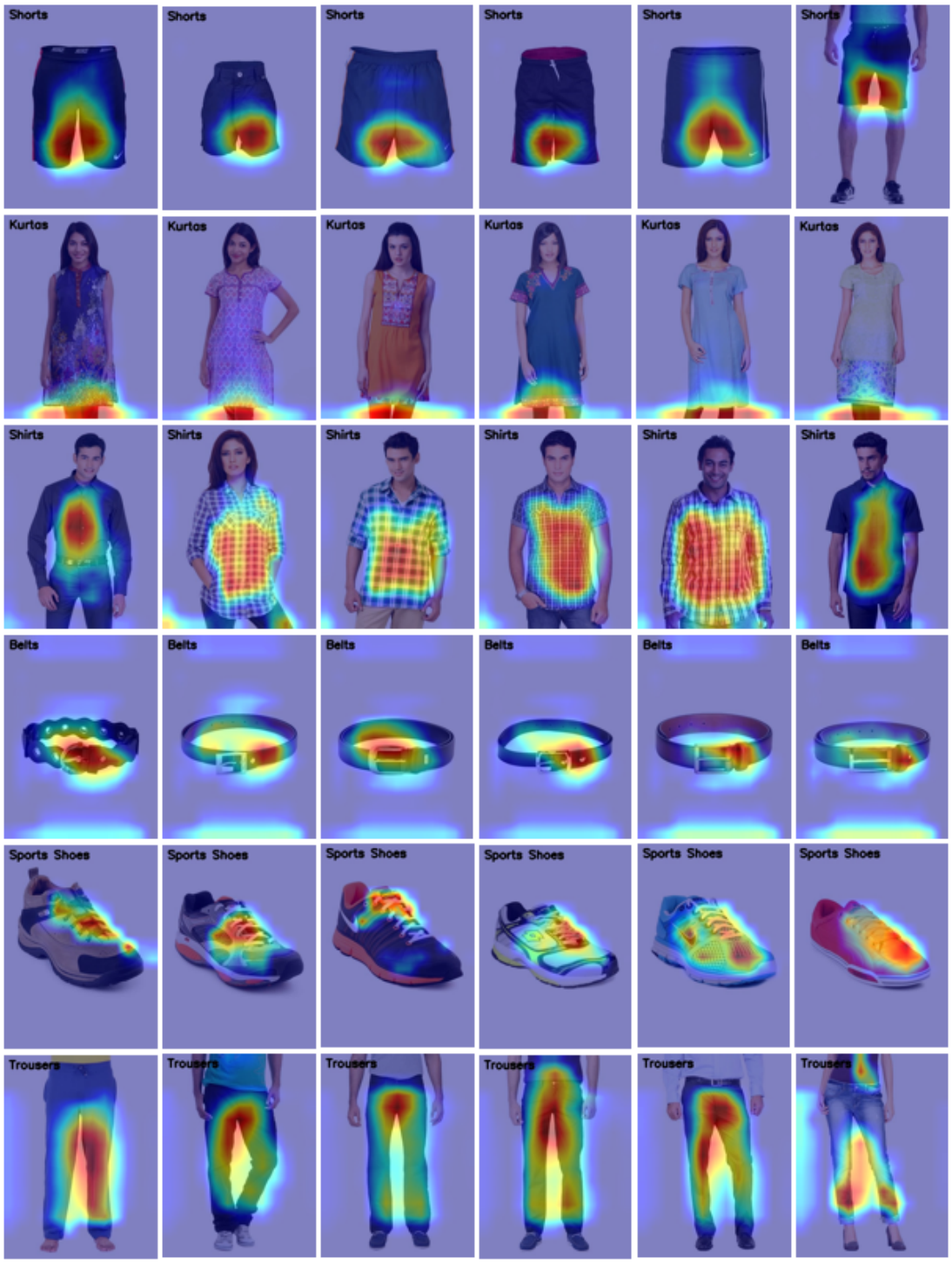

Figura 5.2: Resultados del método Class Activation Mapping para inputs de 3 clases diferentes.

5.1.1.3. Discusión de los resultados

En la figura 5.2 podemos ver los resultados del método Class Activation Mapping para inputs de 6 de las 144 clases del conjunto de datos Fashion Product Images de Kaggle.

En el caso de la clase «Shirts», se puede observar que el modelo pone atención en el torso de las personas, lo que es esperable para esta clase. En los resultados para las clases «Trousers» y «Shorts», se ve que, más que en la prenda, el modelo pone atención en el espacio que hay entre las piernas. Algo parecido sucede con la clase «Kurtas», en las que se ve que la mayor ponderación se le da a la sección inferior de la imagen, donde termina la prenda y se ven las piernas de las modelos.

Estos resultados sugieren que, en algunos casos, el modelo aprende a clasificar utilizando elementos y características ajenas al objeto con el que intuitivamente se pueden asociar a la clase. Por ejemplo, en el caso del pantalón, dado que la probabilidad de que la imagen contenga un pantalón es muy alta cuando existe una cuña vacía en el centro del input, el modelo seguramente aprendió a darle mucha relevancia a esa característica para decidir si la imagen se trata de un pantalón.

En el caso de las Kurtas, también es algo común en las imágenes de esa clase que la prenda termine con un corte casi horizontal en la parte inferior de la imagen, luego del cual se muestra siempre una pequeña sección de las piernas de las modelos. Esta característica distintiva de las imágenes de esta clase pudo haber empujado al modelo a detectar ese patrón para distinguir esa clase de otras.

Lo anterior representa un obstáculo para el objetivo de este trabajo, ya que si para la red no es necesario aprender a reconocer las características de interés, como color y patrones, entonces es poco probable que existan neuronas y canales especializados en detectar esas características.

5.1.2. Mascaras binarias

5.1.2.1. Descripción del experimento

Continuando con la visualización de activaciones, se implementó un método para filtrar las regiones de las imágenes que no sobrepasaran un cierto umbral de activación. Para realizar lo anterior, se comienza generando una imagen de color negro del mismo tamaño que el input. Luego, para cada neurona del canal bajo análisis, se proyecta su campo receptivo sobre la imagen negra y, si la activación de la neurona supera el umbral definido, se reduce la opacidad de esa zona de la imagen a cero. De este modo, se obtiene una máscara que al ser superpuesta con la imagen de entrada, cubre en negro todo lo que no activa lo suficiente a ese canal y deja al descubierto las zonas que sí causaron una alta activación.

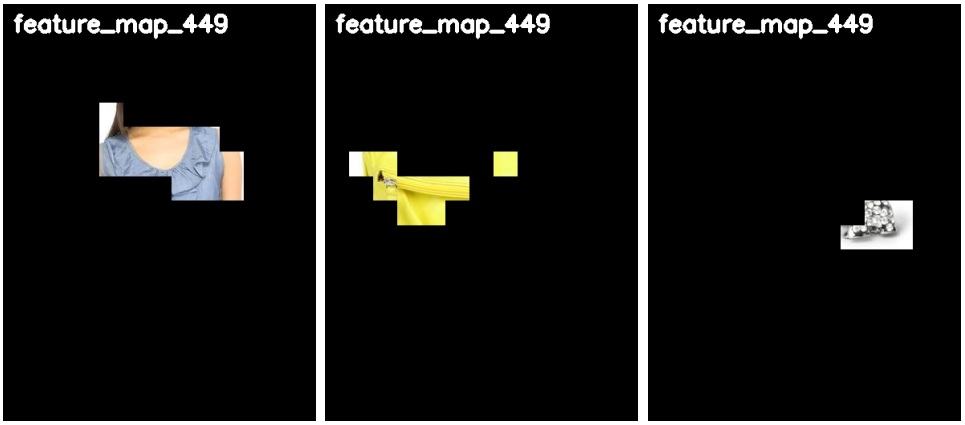

Figura 5.3: Ejemplos del método de máscaras binarias. Las regiones visibles indican que la activación del canal en aquella zona fue superior al umbral.

5.1.2.2. Resultados

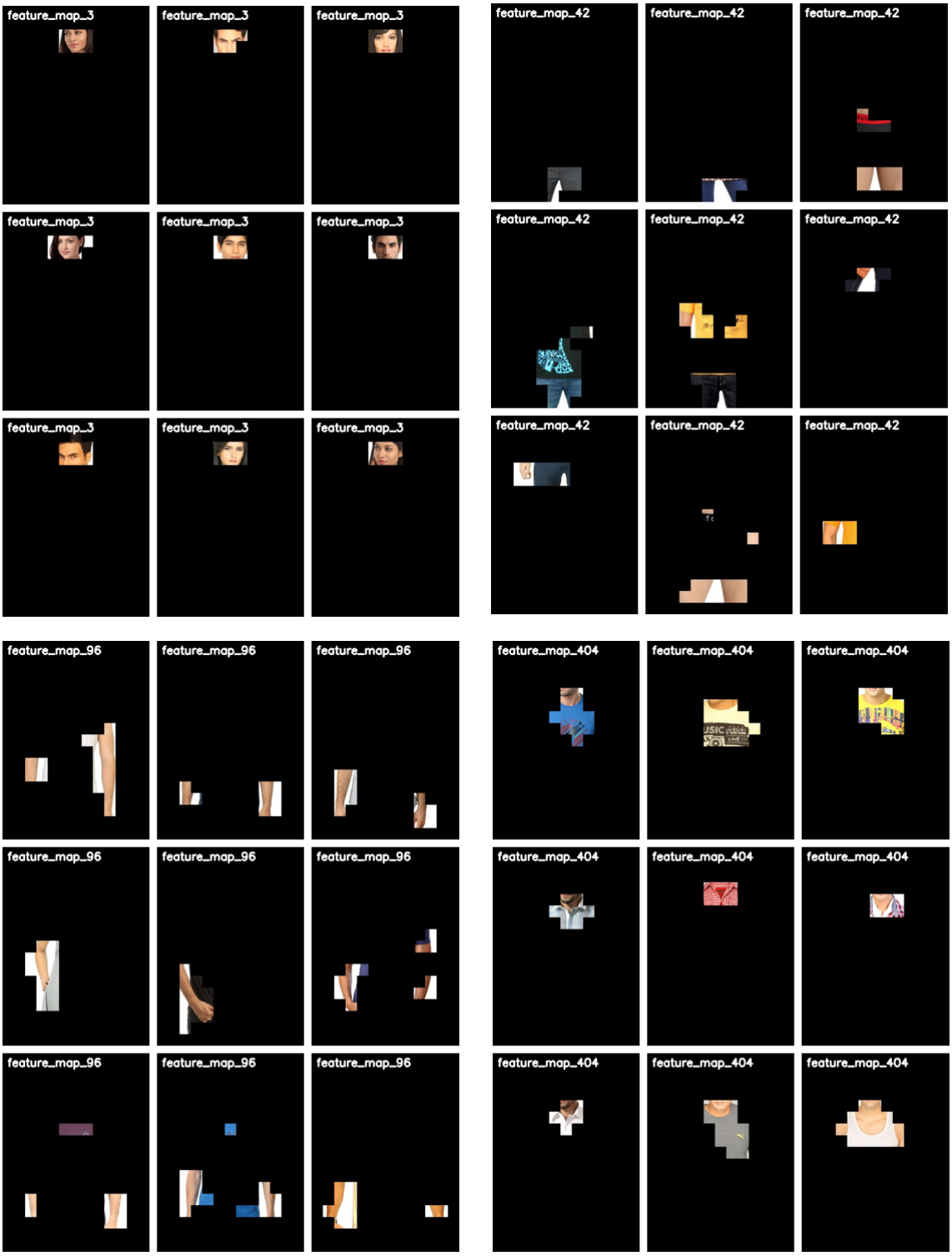

Figura 5.4: Resultados del experimento de máscaras binarias, para algunos canales de la última capa convolucional del modelo.

5.1.2.3. Discusión de los resultados

Los resultados del experimento de máscaras binarias muestran que hay ciertos canales en los que hay una clara tendencia a detectar un cierto elemento o característica. Sin embargo, en la mayoría de los casos, las características que se detectan poco y nada tienen que ver con las prendas en sí. Por ejemplo, en las imágenes de la figura 5.4, se puede ver que varios canales se activan claramente frente a elementos del cuerpo, como la cara, los brazos, las piernas y el cuello.

Una vez más, es claro que el modelo está utilizando información ajena a las prendas de vestir para realizar la clasificación. La causa más probable es que, para este conjunto de datos (y probablemente para la mayoría de los catálogos de ropa), es posible determinar la clase del producto indirectamente, como por ejemplo infiriendo que si en la imagen hay piernas descubiertas, no hay cara ni brazos pero sí hay pies, entonces la imagen probablemente corresponde a un pantalón. De esta manera, un modelo entrenado a partir de imágenes de catálogo no es el apropiado para extraer características como el color y los patrones de la ropa.

Peor aún. Es razonable pensar que, en general, los modelos entrenados para clasificar prendas de vestir no terminan conteniendo canales especializados en detectar características como el color y los patrones de la tela (que es lo que se busca en este caso), ya que estos atributos no son inherentes a ningún tipo de prenda y, por lo tanto, no ayudan a determinar de qué tipo de prenda se trata. Supongamos que el modelo se enfrenta a la imagen de un vestido azul con puntos blancos y debe predecir su clase. Como el color azul y el patrón de puntos pueden encontrarse otros tipos de prenda, el modelo no aprenderá a detectar ese tipo de características, sino que más probablemente se enfocará en discriminar a través de la silueta de las prendas y elementos característicos de ellas, como podrían ser los cordones de un zapato o los botones de una camisa.

Dado que este experimento y el anterior confirman que el modelo de clasificación de ropa no es adecuado para los objetivos de este trabajo, se decidió dejar de lado el modelo entrenado con el conjunto Fashion Product Images. Para los siguientes experimentos, se utilizó una red ResNet50 (la misma arquitectura utilizada hasta ahora), pero esta vez entrenada sobre ImageNet.

El motivo es que ImageNet contiene clases mucho más generales, varias de ellas relacionadas a la naturaleza, donde el color y la textura de los objetos sí pueden servirle al modelo para discernir la clase del input.

5.1.3. Maximización de activación

5.1.3.1. Descripción del experimento

También se decidió usar *Maximización de activación*, otra técnica para analizar modelos convolucionales visualmente. La principal diferencia con los métodos anteriores es que, en este caso, el análisis se hace sobre un input ficticio, el cual es generado por el método para mostrar a qué patrones de una imagen responde mayormente un cierto canal.

Dado que la arquitectura que se utilizó para el modelo (ResNet50) es bastante profunda, se decidió usar la versión del método que introduce regularización al proceso de generación del input ficticio. En otros trabajos, se ha visto que la versión original del método no funciona bien para arquitecturas demasiado profundas, dado que las representaciones de los datos se vuelven muy abstractas y las visualizaciones resultantes no contienen patrones visuales interpretables de forma intuitiva.

5.1.3.2. Resultados

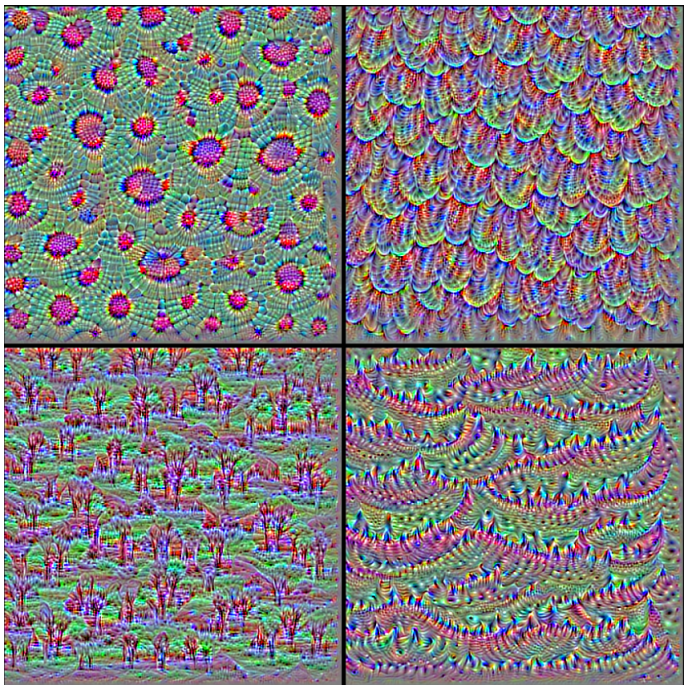

Figura 5.5: Maximización de activación sobre 4 canales de la última capa convolucional del modelo ResNet50 entrenado sobre ImageNet.

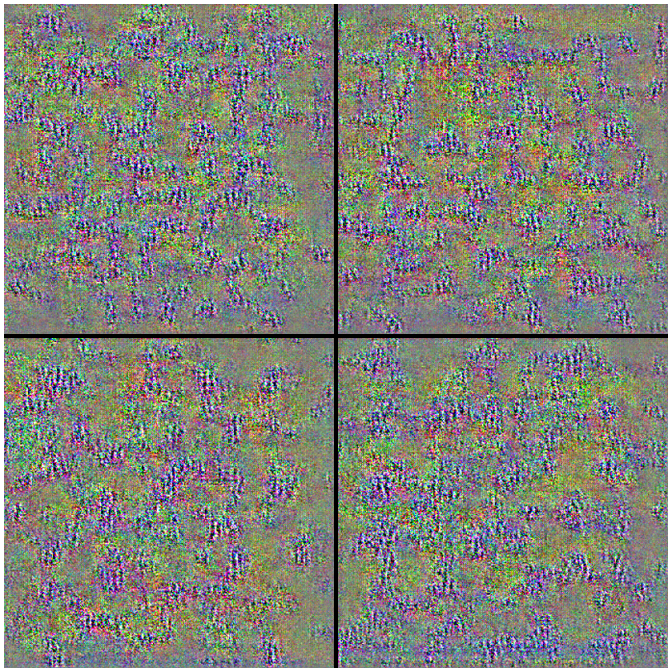

Figura 5.6: Maximización de activación sobre 4 canales de la última capa convolucional de modelo ResNet50 entrenado sobre Kaggle.

5.1.3.3. Discusión de los resultados

Las imágenes sintetizadas por este método muestran resultados muy diferentes para los dos modelos. En el caso del modelo entrenado con ImageNet (figura 5.5), las imágenes muestran patrones bastante definidos, siendo posible incluso asociarlos a ciertas texturas que se ven en la naturaleza, como escamas, ramas, dientes y manchas redondas. Estos resultados son alentadores, ya que varios de estos canales podrían también detectar ciertos patrones de tela, como podría ser para el caso de un polerón de leopardo, un vestido con puntos o un top de lentejuelas.

En contraste, el mismo experimento sobre el modelo entrenado sobre Kaggle no generó imágenes con patrones identificables (figura 5.6). El motivo de esto queda pendiente de determinar, pero puede deberse a falta de entrenamiento, pocos datos o bien que el problema para el cual fue entrenado no haya sido lo suficientemente exigente como para inducir la detección de patrones complejos.

5.2. Análisis de activaciones

Dado que la hipótesis es que los canales de las capas de un modelo convolucional se especializan en detectar ciertas características, se realizaron varios experimentos para intentar asociar cuantitativamente patrones de activación de los canales con características. En el contexto de estos experimentos, nos referiremos con el término *descriptor intermedio* al vector que se obtiene al aplicar *Global Average Pooling* a la salida de un bloque intermedio de la arquitectura luego de alimentar el modelo con un input (o un conjunto de inputs).

Al aplicar Global Average Pooling sobre la salida de un bloque, las activaciones del mapa de características de cada canal se reducen a un valor de activación promedio para ese canal. De esta forma, si una capa convolucional tiene 512 canales, podemos ver al descriptor intermedio de esa capa como un vector de 512 dimensiones, en el cual las activaciones del conjunto de neuronas de cada canal se resumen en un solo valor por canal, como se ilustra en la figura 5.7.

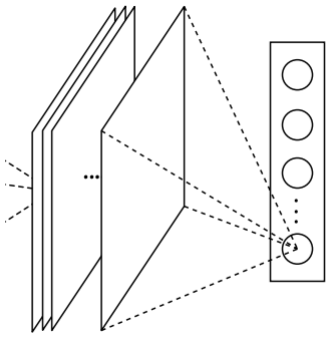

Figura 5.7: Obtención de un descriptor intermedio mediante la aplicación de Global Average Pooling.

Para los experimentos de esta sección, se usaron los dos conjuntos de datos que se construyeron para anotar el color y la textura de las prendas.

5.2.1. Promedio de activaciones por clase

5.2.1.1. Descripción del experimento

El primer experimento que se realizó en esta línea fue, para cada clase de los conjuntos de prendas anotadas según color o textura, promediar las activaciones de los distintos inputs y analizar aquellos canales que acumularon una mayor activación.

Aún si la hipótesis de que ciertas neuronas se especializan en detectar atributos concretos es correcta, es importante notar que las imágenes del conjunto, además de la característica etiquetada, generalmente contienen otros elementos que también pueden gatillar activaciones de otros canales, como podrían ser objetos en el fondo de la imagen o, en el caso de las prendas de vestir, características del modelo que viste la prenda. Por este motivo, se espera que no solo se activen aquellos canales que detecten la característica correspondiente a la etiqueta, sino también otros que detecten otras características igualmente presentes en la imagen. Sin embargo, como todas las imágenes del subconjunto correspondiente a la clase comparten al

menos la característica correspondiente a la clase, se espera ver que, en comparación a la mayoría de los canales, al menos algunos canales tengan un peak de activación evidente.

5.2.1.2. Activaciones promediadas para prendas anotadas por color

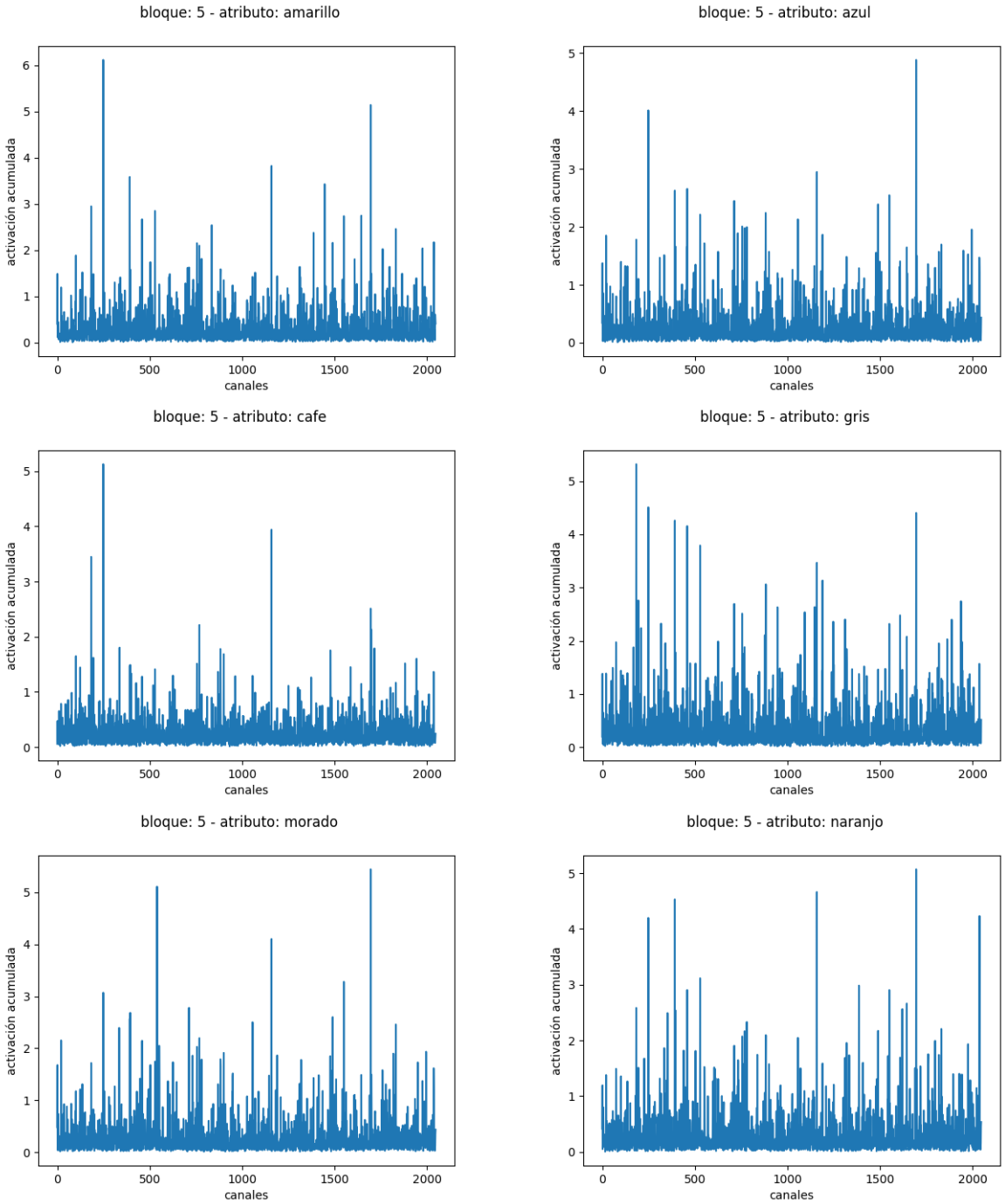

Figura 5.8: Activación promedio de los canales de salida del bloque 5 de ResNet50 para inputs de 6 clases diferentes del conjunto de imágenes prendas por color.

5.2.1.3. Activaciones promediadas para prendas anotadas por textura

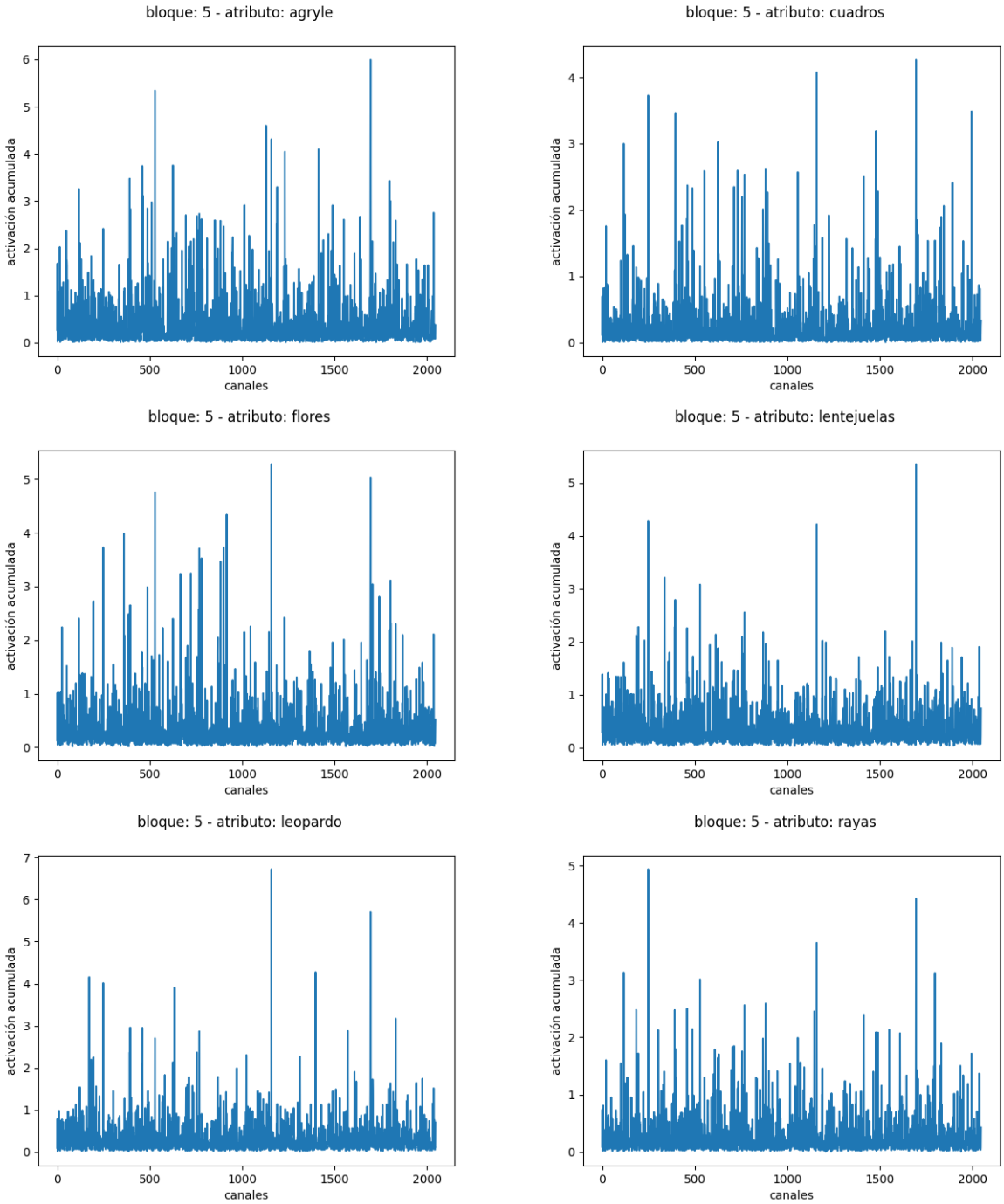

Figura 5.9: Activación promedio de los canales de salida del bloque 5 de ResNet50 para inputs de 6 clases diferentes del conjunto de imágenes prendas por textura.

5.2.1.4. Discusión de los resultados

En los gráficos de este experimento (figuras 5.8 y 5.9), se puede observar que las activaciones de cada clase son bastante diferentes. Para cada clase, existen peaks muy marcados que no están presentes en las activaciones promediadas de las otras clases. Esto es un buen augurio para el objetivo de caracterizar las clases según el valor de sus descriptores intermedios, ya que si la distribución de activaciones promedio es lo suficientemente distintiva para cada clase, quiere decir que las activaciones que producen imágenes de una misma clase deben parecerse.

También se puede ver que las intensidades de las activaciones promedio se encuentran en el mismo orden de magnitud, fluctuando entre 0 y 8, aproximadamente. Esta homogeneidad sugiere que utilizar la distancia euclidiana entre los descriptores podría entregar buenos resultados para clasificar inputs.

Además, se puede ver que existen ciertos canales que se activan fuertemente en todas las clases, como es el caso del canal 1731. Esto podría deberse a que ese canal se activa frente algún atributo compartido por todas las imágenes, como podría ser el fondo blanco o la piel de los modelos.

5.2.2. Reducción de dimensión y clusterización

5.2.2.1. Descripción del experimento

Para determinar qué tan característicos de cada clase del conjunto de datos son los diferentes descriptores intermedios de cada capa, se redujeron de dimensión las activaciones de distintas capas del modelo a varios espacios de distinta cantidad de dimensiones. En particular, para generar visualizaciones de las activaciones, se redujo el espacio a 2 dimensiones. Sin embargo, también se realizaron experimentos con más dimensiones. La técnica utilizada para reducir la dimensión fue *Uniform Manifold Approximation and Projection for Dimension Reduction* (UMAP).

Una vez reducidas las dimensiones de los descriptores, se generaron visualizaciones para analizar cuantitativamente la distribución de las activaciones en el espacio. El objetivo de esto es determinar si es que los descriptores intermedios de inputs pertenecientes a la misma clase efectivamente se encuentran más cercanos entre sí en el espacio reducido. Además, se utilizó el algoritmo *HDBSCAN* para crear clusters y las métricas *rand-index* y *mutual-information-score* para evaluar la similitud entre la partición resultante de la clusterización y la partición original del conjunto de datos dada por las clases.

5.2.2.2. Resultados

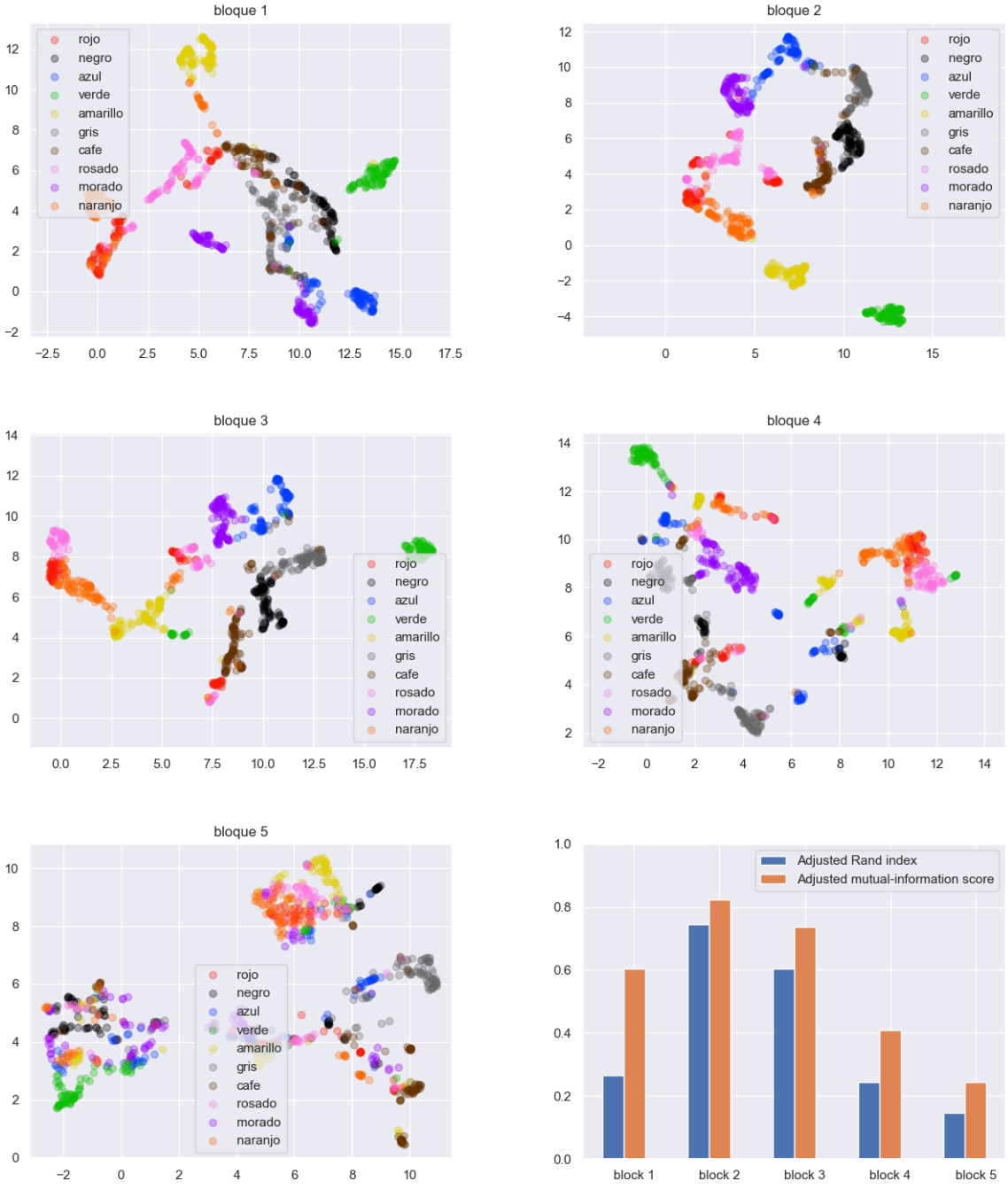

Figura 5.10: Espacio reducido con activaciones de la salida del segundo bloque convolucional del modelo entrenado con ImageNet para inputs del conjunto de imágenes anotadas por color. Además, se muestran las métricas adjusted Rand index y mutual-information score para la distribución de activaciones de cada bloque en el espacio reducido.

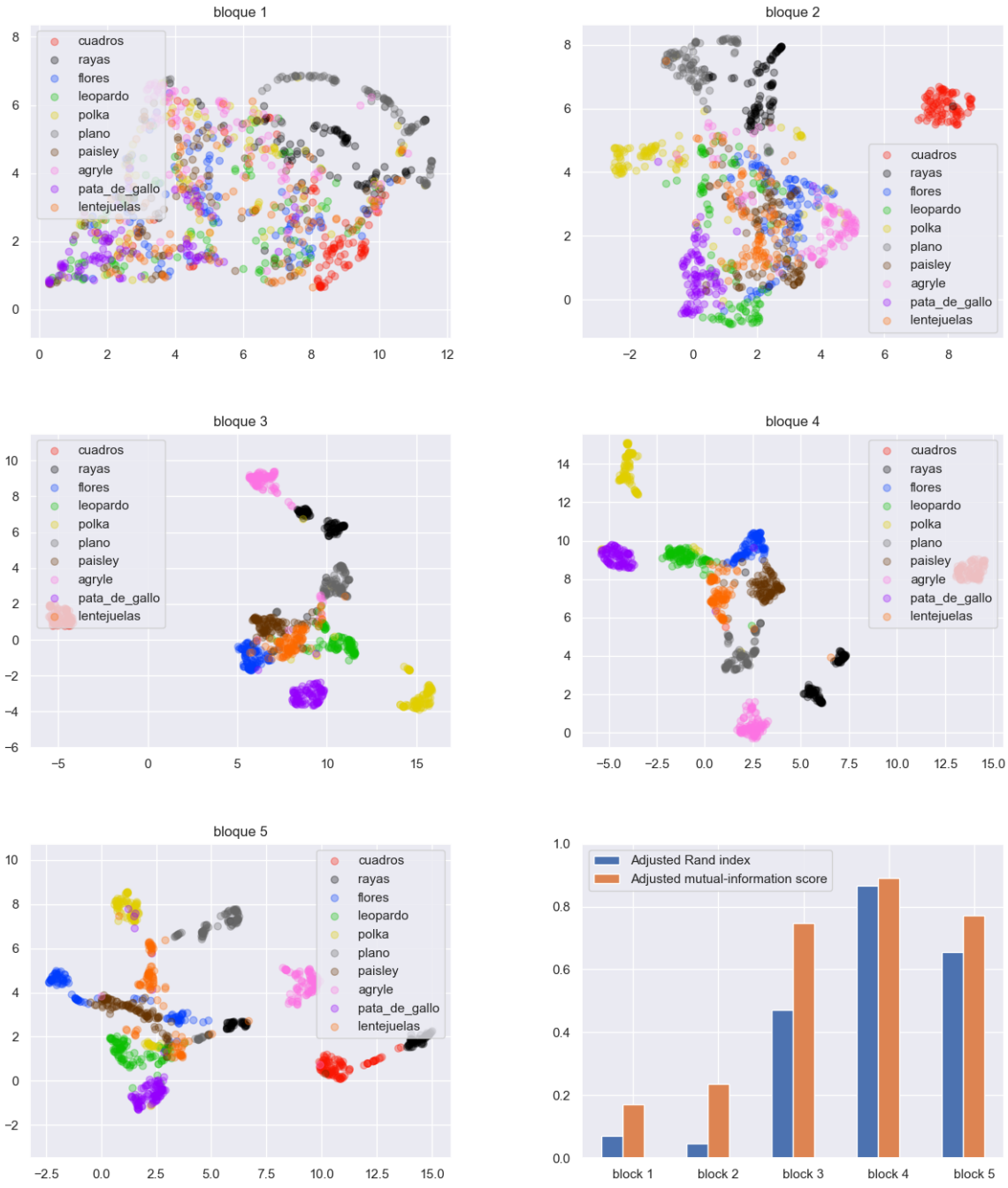

Figura 5.11: Espacio reducido con activaciones de la salida del cuarto bloque convolucional de ResNet50 del modelo entrenado con ImageNet para inputs del conjunto de imágenes anotadas por textura. Además, se muestran las métricas adjusted Rand index y mutual-information score para la distribución de activaciones de cada bloque en el espacio reducido.

5.2.2.3. Discusión de los resultados

Los espacios reducidos visualizados en los gráficos 5.10 y 5.11 muestran que los descriptores de los inputs de una misma clase sí se encuentran más cercanos en el espacio. Sin embargo, las imágenes muestran que la capacidad del modelo para aglutinar descriptores de una misma clase y distanciar los de clases diferentes se ve afectada según qué tan profundo está el bloque desde el cual se toman los descriptores.

Comprobando lo que se planteó en la hipótesis, las capas más superficiales diferencian mejor los descriptores de las clases de colores, al estar más especializadas en características básicas. Por otro lado, se puede observar que las capas más profundas capturan mejor la información de mayor complejidad. Esto se refleja en que los descriptores de las clases de texturas están mejor agrupados en la salida del bloque 4.

Estas observaciones coinciden con los gráficos que muestran los índices Rand y de información mutua, en los que se ve que en el caso de los colores, el máximo para ambos índices se encuentra en el bloque 2, mientras que en el caso de las texturas, estos máximos se alcanzan utilizando los descriptores del bloque 4.

5.2.3. Visualización de activaciones de los conjuntos de datos usando GridCut

5.2.3.1. Descripción del experimento

Este experimento consiste en generar visualizaciones en las que se pueda apreciar cómo se distribuyen las imágenes del conjunto de datos de forma más intuitiva que con UMAP. En vez de visualizar puntos en el espacio reducido, se visualiza una grilla en la que cada celda corresponde a una imagen del conjunto.

Como el método GridCut preserva las agrupaciones que existen en el espacio multidimensional, se esperaría ver que las imágenes se agrupen según su clase.

5.2.3.2. Resultados

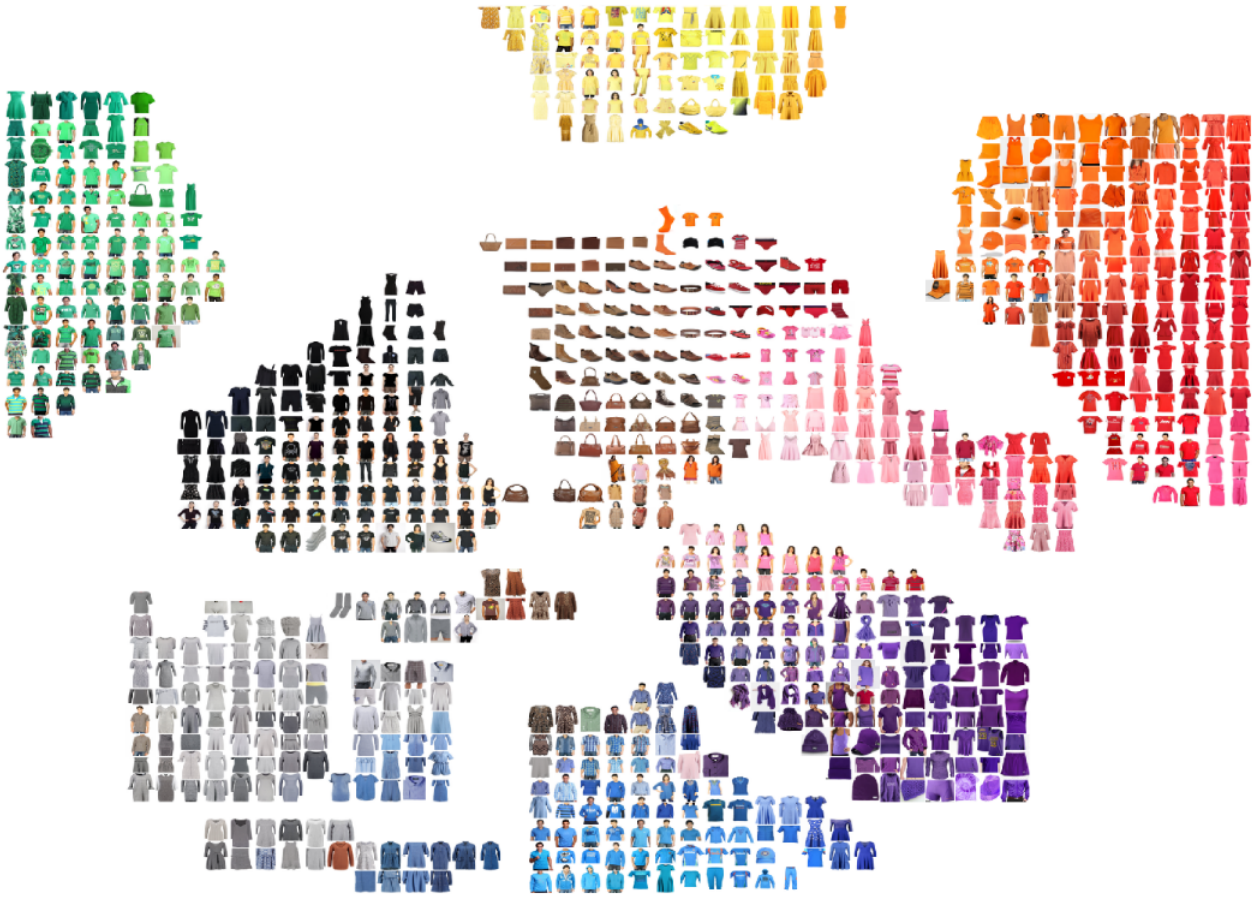

Figura 5.12: Visualización de GridCut para el conjunto de prendas anotadas por color, utilizando las activaciones del bloque 2 del modelo con arquitectura ResNet50 entrenado con ImageNet.

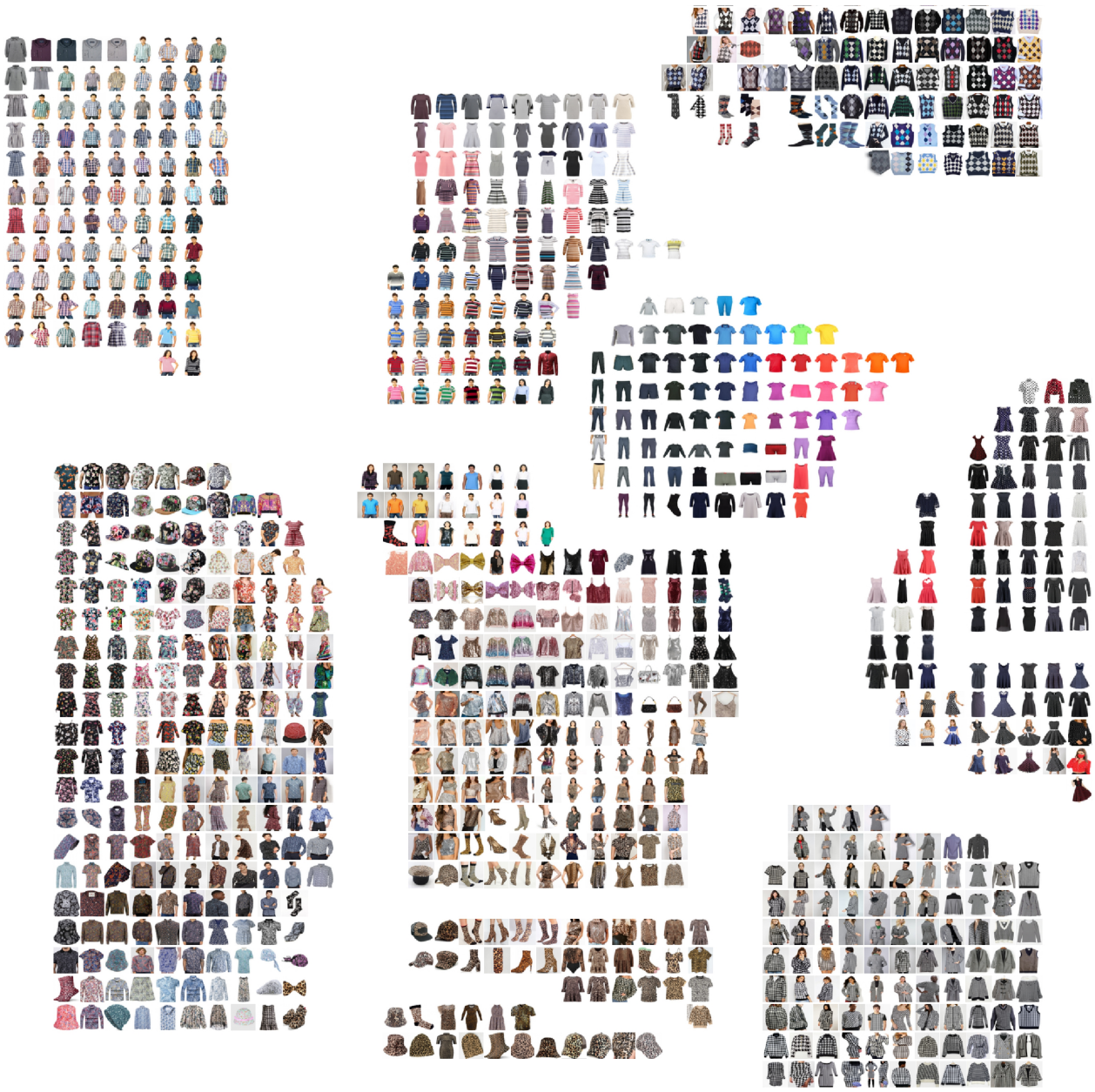

Figura 5.13: Visualización de GridCut para el conjunto de prendas anotadas por textura, utilizando las activaciones del bloque 4 del modelo con arquitectura ResNet50 entrenado con ImageNet..

5.2.3.3. Discusión de los resultados

Las visualizaciones muestran que, a grandes rasgos, tanto en el caso de las imágenes anotadas por color (figura 5.12) como para aquellas anotadas por textura (figura 5.13), las activaciones sí se agrupan según la clase de sus inputs.

Sin embargo, existen ciertos clusters que agrupan a imágenes de clases distintas, pero que en la práctica se parecen bastante. Este es el caso de las prendas naranjas y rojas, cuyo tono es bastante similar. Algo parecido ocurre en el segundo gráfico con las prendas floreadas y

con patrón Paisley. Una vez más, ambas clases son visualmente parecidas.

5.2.4. Distribución de distancias entre pares de descriptores

5.2.4.1. Descripción del experimento

Este experimento tiene por objetivo visualizar las distancias entre descriptores de imágenes de una misma clase (pares positivos) y compararlas con las distancias entre descriptores de imágenes de clases diferentes (pares negativos). Se tomaron aleatoriamente 3000 pares positivos y 3000 pares negativos por cada clase, sumando un total de 30.000 pares positivos y 30.000 pares negativos para las imágenes anotadas según color y lo mismo para las imágenes anotadas según textura.

Además, se repitió el experimento en espacios reducidos por UMAP con distintas dimensiones, con el objetivo de analizar cómo afecta a estas distribuciones la dimensionalidad de los datos. El interés por realizar lo anterior proviene del hecho de que, a medida que los datos tienen más dimensiones, tienden a ser más dispersos, lo que se conoce comúnmente como *Maldición de la dimensionalidad*. Por lo tanto, podría darse que en menores dimensiones, las distribuciones tuvieran un menor solapamiento que en el espacio original. La métrica de distancia utilizada en este experimento fue distancia euclidiana.

5.2.4.2. Resultados

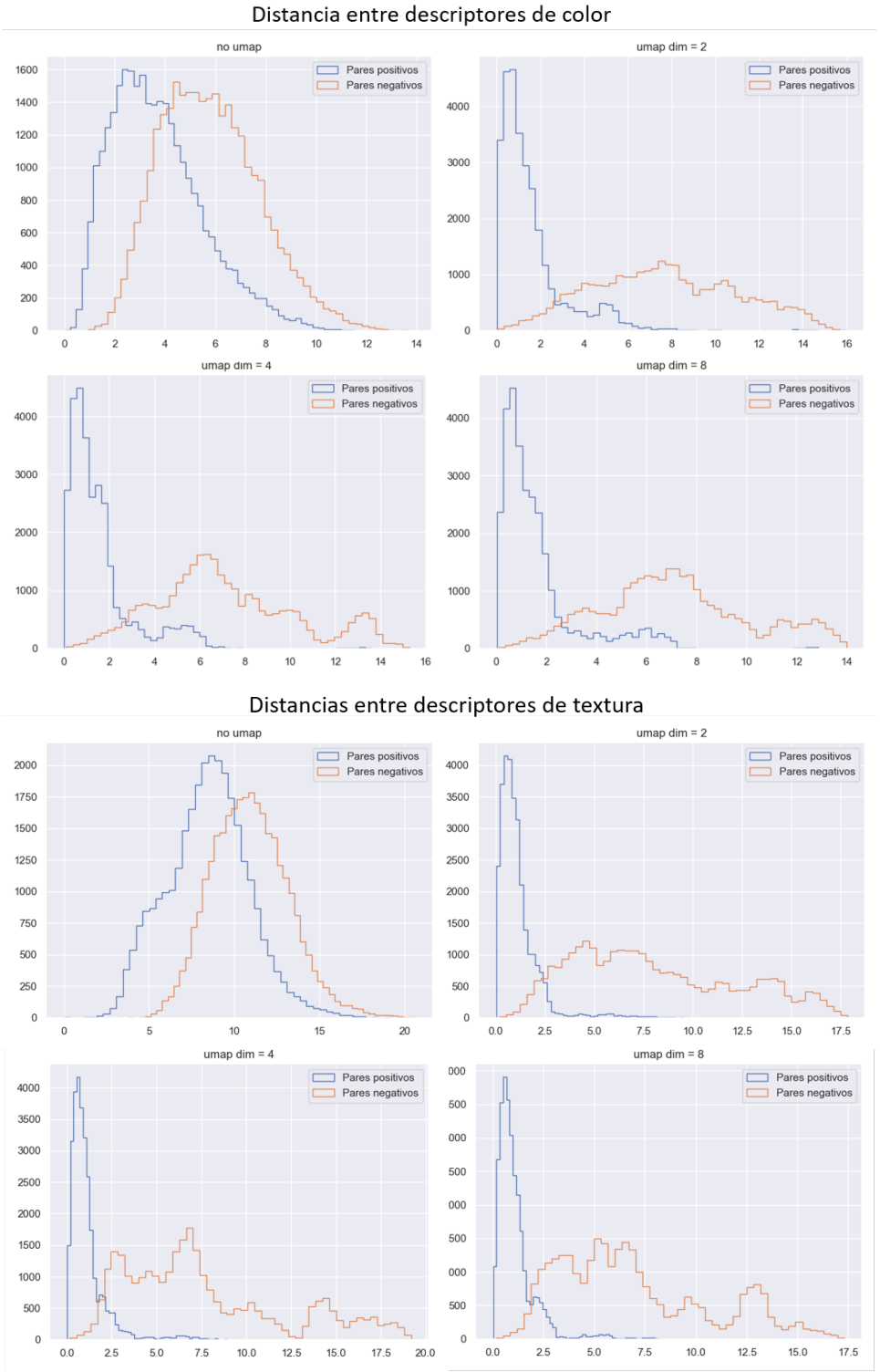

Figura 5.14: Los histogramas muestran las distancias euclidianas entre 30.000 pares positivos y entre 30.000 pares negativos en distintos espacios reducidos con UMAP.

5.2.4.3. Discusión de los resultados

En los gráficos de la figura 5.14 se muestra, para distintos espacios reducidos con UMAP, la distribución de distancias entre 30.000 pares positivos de descriptores y, al mismo tiempo, la distancia entre 30.000 pares negativos.

Lo primero que llama la atención es que, en las dimensiones originales (i.e. sin aplicar UMAP), tanto para el caso de los descriptores de color como los de textura, hay un solapamiento importante entre la distribución de distancia pares positivos y la distribución entre pares negativos. Esto quiere decir que existe una cantidad importante de pares negativos cuya distancia es menor que la de varios pares positivos.

Si bien lo anterior inicialmente puede parecer inconsistente con los resultados anteriores en los que se mostraba que los descriptores se agrupaban en clusters bastante definidos, en realidad no son situaciones incompatibles.

Supongamos que en el espacio original hay dos clusters claramente definidos (esto es, que para la mayoría de sus puntos, los vecinos más cercanos sean de su misma clase). Si el diámetro de los clusters es lo suficientemente grande y la distancia entre sus centroides es lo suficientemente pequeña, es perfectamente posible que muchos pares positivos tengan una distancia mayor que la distancia de varios pares negativos.

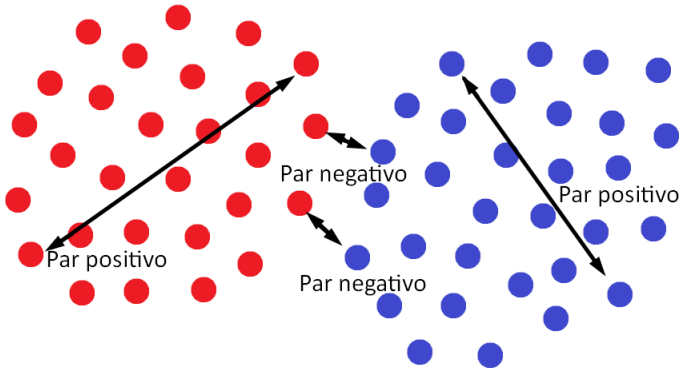

Figura 5.15: Ejemplo de clusters bien definidos en los que se pueden formar varios pares positivos con distancia mayor a algunas distancias entre pares negativos

Además, dada la tendencia de los datos a volverse más dispersos a medida que tienen más dimensiones, es esperable que, al disminuir las dimensiones del espacio, el solapamiento de ambas distribuciones disminuya. Esto se ve reflejado claramente en los histogramas que muestran las mismas distancias pero en espacios reducidos con UMAP.

5.3. Clasificación usando descriptores intermedios

Una vez concluidos los experimentos de análisis sobre las activaciones, se procedió a realizar experimentos de extracción de características, tema que es el objetivo principal de este trabajo. Estos experimentos buscan realizar tareas de clasificación a partir de los descriptores intermedios de los que se ha hablado en las secciones interiores.

Para los experimentos de esta categoría, se realizaron dos versiones de cada experimento, una utilizando el conjunto de prendas anotadas por color y la otra utilizando el conjunto anotado por textura. El motivo de esto es que existe bastante solapamiento entre las clases de cada grupo. Por ejemplo, una camisa de cuadros puede ser, a su vez, una camisa verde. En un caso así, no tiene sentido forzar al modelo a elegir entre un color o una clase, sino que es más razonable que el modelo pueda elegir una característica de cada grupo para cada input. De esta forma, en algunos experimentos se midió la exactitud aisladamente para cada caso.

Además, para evitar sesgos inducidos por la partición escogida para separar los conjuntos de entrenamiento y de validación, se utilizó la técnica de validación cruzada estratificada con 5 splits (figura 5.16). Esto quiere decir que, para cada experimento, se realizaron 5 iteraciones con 5 particiones diferentes del conjunto. Cada partición se forma dividiendo al conjunto en 5 subconjuntos, de los cuales se toma un subconjunto como conjunto de validación y los otros 4 se unen para formar el conjunto de entrenamiento. En cada iteración, se toma un subconjunto de validación distinto y, finalmente, se promedian las métricas obtenidas con las 5 iteraciones.

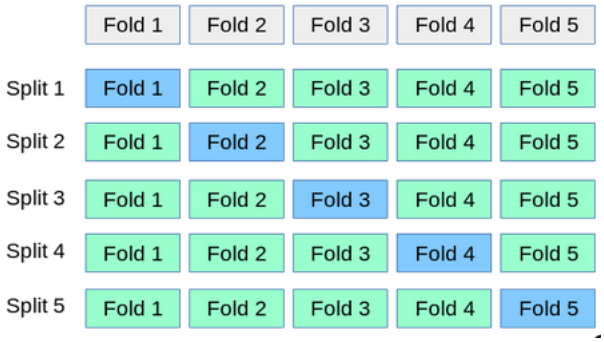

Figura 5.16: Validación cruzada con 5 splits.

5.3.1. Clasificación usando distancia al descriptor promedio de cada clase

5.3.1.1. Descripción del experimento

El primer intento de realizar clasificación con descriptores intermedios se hizo construyendo, para cada clase, un descriptor promedio. Específicamente, para cada clase del conjunto de datos, se calcularon los descriptores de 80 imágenes con esa clase y se promediaron en un solo descriptor, el que se consideró como el descriptor representante de la clase. Luego, para cada una de las otras 20 imágenes de cada clase, se calculó su descriptor intermedio en cada bloque del modelo y se le asignó una clase dependiendo de a qué descriptor representante se encontraba más cercano su descriptor. Para la métrica de exactitud, se consideró como acierto cada vez que la clase predicha coincidía con la de la etiqueta del input. Este experimento se repitió para dos métricas de distancia diferentes:

- Distancia euclidiana
- Distancia coseno

5.3.1.2. Resultados

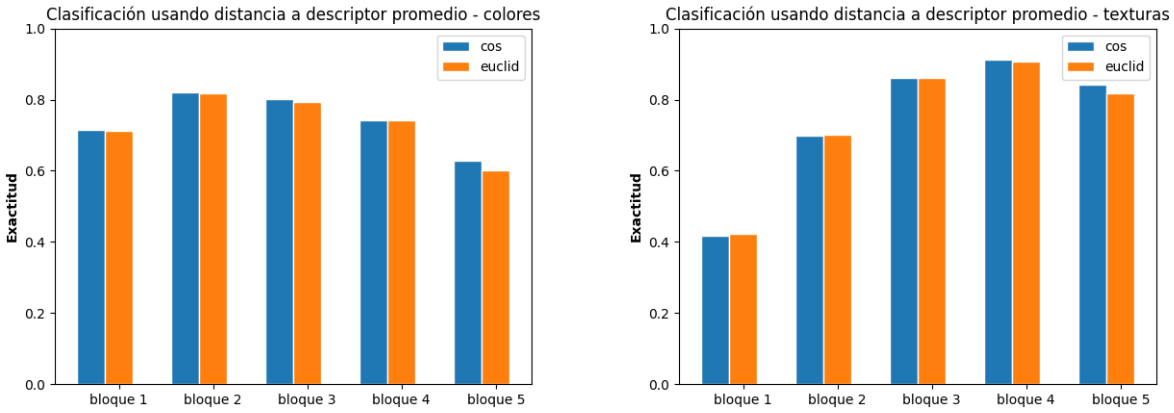

Figura 5.17: Exactitud lograda por el método en distintos bloques de la red. A la izquierda, los resultados para imágenes anotadas por color. A la derecha, los resultados para imágenes anotadas por textura.

Tabla 5.1: Exactitud lograda en cada bloque según métrica de distancia para imágenes anotadas según color.

Bloque	Euclidiana	Coseno
1	0.715	0.711
2	**0.821**	**0.821**
3	0.801	0.793
4	0.742	0.743
5	0.629	0.602

Tabla 5.2: Exactitud lograda en cada bloque según métrica de distancia para imágenes anotadas según textura.

Bloque	**Euclidiana**	**Coseno**
1	0.423	0.417
2	0.701	0.698
3	0.860	0.862
4	**0.907**	**0.912**
5	0.818	0.842

5.3.1.3. Discusión de los resultados

En este experimento se midió la exactitud lograda al clasificar el input según su cercanía a los descriptores representantes de cada clase, los cuales se construyeron promediando los descriptores de 80 imágenes pertenecientes a la clase. El experimento se repitió para distintos bloques de la red y, a la vez, con distintas funciones de distancia (distancia coseno y euclidiana).

En primer lugar, los resultados indicados en la figura 5.17 muestran un desempeño general bastante similar para ambas métricas de distancia. Además, en las tablas 5.1 y 5.2 se puede ver que, en el caso de las imágenes anotadas por color, la mayor exactitud (82 %) se logró utilizando los descriptores de la salida del bloque 2 , mientras que en el caso de las imágenes anotadas por textura, el máximo de exactitud (91 %) se obtuvo utilizando los descriptores de la salida del bloque 4.

5.3.2. Clasificación usando K vecinos más cercanos

5.3.2.1. Descripción del experimento

En vez de construir descriptores representantes para cada clase, en este caso la clasificación se hizo a partir de la clase predominante entre los k descriptores más cercanos al descriptor intermedio de cada input, de la misma forma que en el algoritmo de los k vecinos más cercanos. Se repitió este experimento para varios valores de k.

5.3.2.2. Resultados

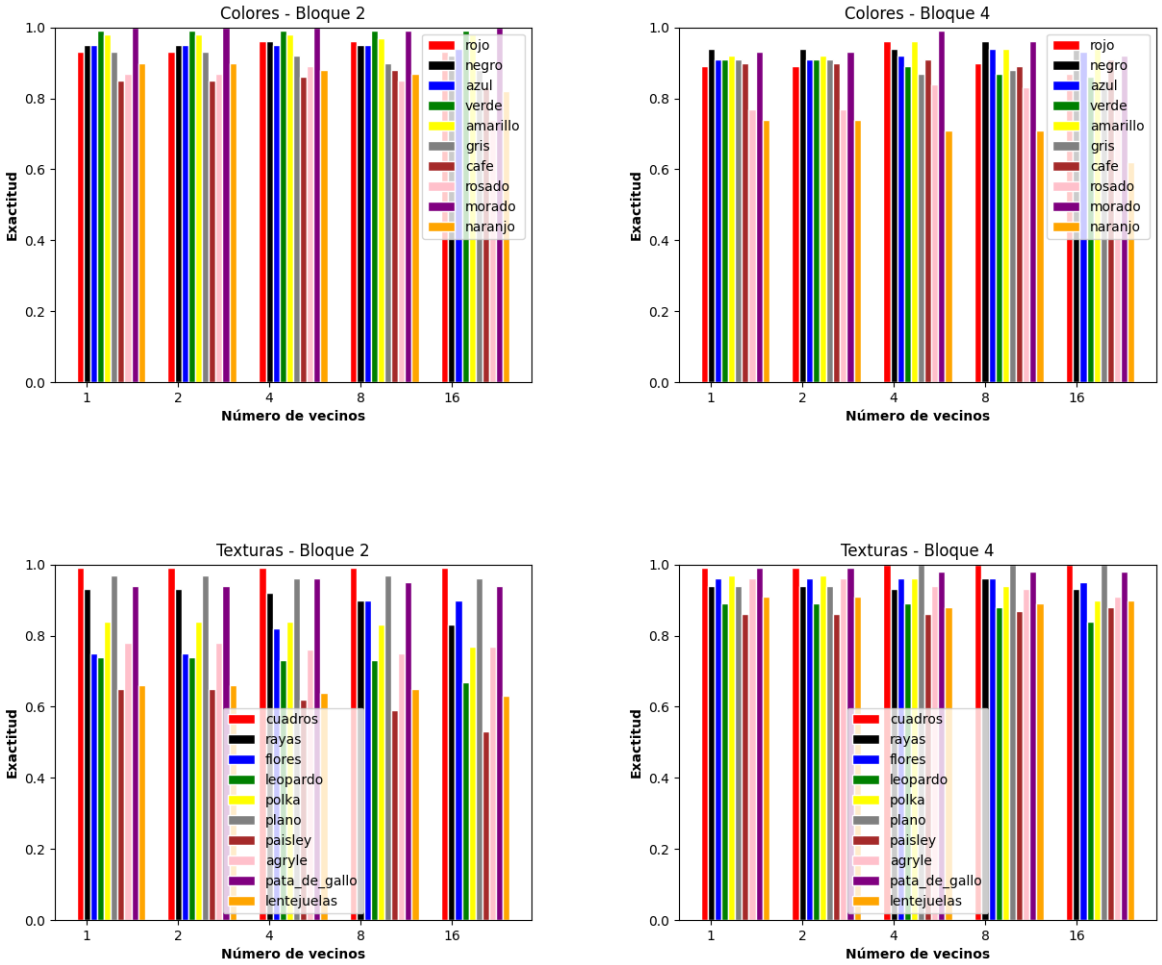

Figura 5.18: Exactitud desagregada por clases para distintos valores de k

Tabla 5.3: Exactitud promedio para imágenes anotadas por color (izquierda) y textura (derecha)

k	**Bloque 2**	**Bloque 4**
1	0.935	0.882
2	0.935	0.882
4	**0.939**	0.899
8	0.931	0.888
16	0.903	0.866

k	**Bloque 2**	**Bloque 4**
1	0.825	0.941
2	0.825	0.941
4	0.824	940
8	0.826	**0.941**
16	0.799	0.929

5.3.2.3. Discusión de los resultados

En los resultados de este experimento, en el que la clase del input se deduce de la clase de los k descriptores más cercanos, se puede ver que se logra un aumento generalizado en la exactitud lograda en comparación a los métodos de clasificación anteriores (figura 5.18 y tabla 5.3).

En el caso de las prendas anotadas por color, se logró una exactitud promedio de 94 % en la salida del segundo bloque de la red, mientras que en la salida del bloque 4 (más profundo) la exactitud promedio alcanzó un 90 %.

Para las prendas anotadas por textura, se alcanzó una exactitud también de 94 %, aunque, a diferencia del caso de las prendas anotadas por colores, esto se logró utilizando los descriptores del bloque 4. Esto es coherente con los resultados del experimento anterior.

Además, el valor del parámetro k que tuvo un mejor desempeño en el caso de los colores fue 4. En este caso, la clase del input se deduce de la clase del descriptor más cercano. Para los descriptores de texturas, el valor de k que obtuvo mejores resultados fue el 1, empatando con 2 y 8. De todas formas, se puede ver que para valores de k iguales o menores a 8, las exactitudes logradas variaron muy poco

5.3.3. Clasificación en espacio reducido

5.3.3.1. Descripción del experimento

Este experimento consiste en replicar el experimento anterior, pero dentro de un *espacio reducido*. Un espacio reducido es un espacio al cual se pueden trasladar vectores de más dimensiones, reteniendo (idealmente) parte de la topología del espacio original. El motivo de este experimento es que, dado que se espera que los descriptores intermedios de los inputs de cada clase estén agrupados cerca unos de otros, podría darse que el algoritmo de reducción de dimensión descarte aquellas características que no contribuyen a mantener esa cercanía, con el fin de obtener una topología similar a la original. Esto podría, eventualmente, mejorar la clasificación basada en distancia.

Para construir el espacio reducido, se utilizó UMAP para reducir 800 descriptores intermedios (80 por clase) de cada tipo de característica. Luego, al clasificar, se calculó el descriptor de cada input, se redujo a través de UMAP y se le asignó la clase más frecuente entre los k descriptores más cercanos dentro del espacio reducido.

5.3.3.2. Resultados

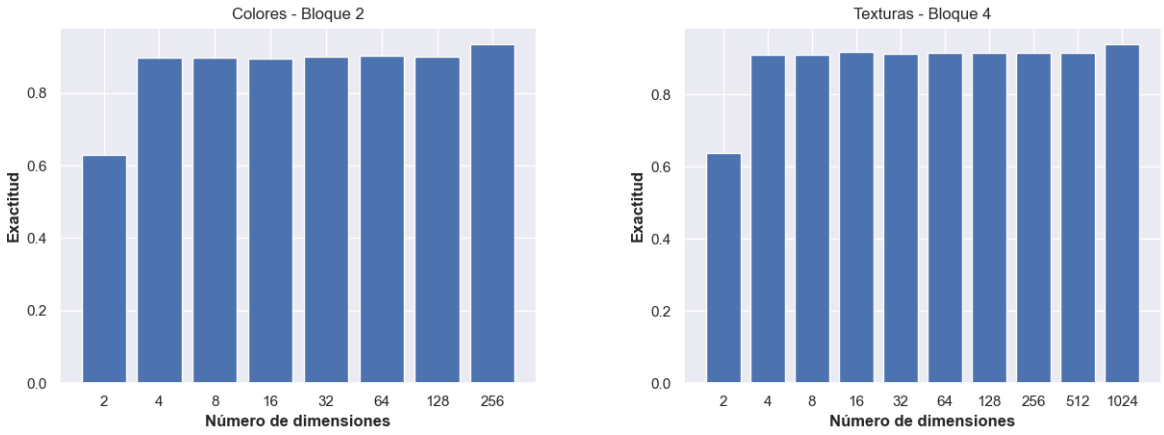

Figura 5.19: Exactitud lograda usando la técnica del vecino más cercano dentro de espacios reducidos de distinta dimensión generados con UMAP. En ambos gráficos, el valor de exactitud para la dimensión más alta corresponde a la exactitud lograda en las dimensiones originales.

Tabla 5.4: Exactitud lograda utilizando la salida del bloque 2 y el bloque 4 para color y textura, respectivamente, reduciendo los descriptores a espacios de diferentes dimensiones usando UMAP.

Dimensiones	**Color**	**Textura**
2	0.650	0.676
4	0.892	0.903
8	0.901	0.910
16	0.899	0.913
32	0.904	0.912
64	0.894	0.916
128	0.897	0.916
256	0.934	0.915
512	–	0.916
1024	–	0.939

5.3.3.3. Discusión de los resultados

En la figura 5.19 se puede ver que, tanto para las características de patrón y color, hacer la clasificación dentro del espacio reducido no genera mejores resultados que al hacerlo en el espacio original. El objetivo de este experimento era explorar la posibilidad de que el algoritmo de reducción disminuyera la importancia de aquellos canales que no incidían mayormente en la cercanía o lejanía de las activaciones, lo que podría haber mejorado el rendimiento del algoritmo de clasificación en base a los k vecinos más cercanos. Sin embargo, es evidente que, independiente del número de dimensiones ocupado para la reducción, los resultados son inferiores.

De todas formas, es interesante ver que, para el caso del espacio reducido a 32 dimensiones, la exactitud lograda para ambos conjuntos es solo un 3 % inferior a la exactitud lograda en el espacio original, lo que muestra la capacidad del método de funcionar en bajas dimensiones.

5.3.4. Recuperación de imágenes similares

5.3.4.1. Descripción del experimento

Este experimento tiene por objetivo evaluar la capacidad de los descriptores intermedios para realizar tareas de recuperación de imágenes similares dado un input. En este caso, se calculan dos descriptores del input, uno para el color y otro para la textura, cada uno a través del bloque que tuvo el mejor desempeño clasificando el correspondiente tipo de característica. Es decir, si se ve que el bloque i obtuvo los mejores resultados clasificando color y que el bloque j clasificó mejor las texturas, entonces se calculan los descriptores respectivos en cada uno de estos bloques.

Luego, para cada imagen del conjunto de datos (excluyendo a la imagen de input), se calculan ambos descriptores. Una vez calculados los descriptores de todas las imágenes, se generan tres ordenamientos de las imágenes del conjunto de datos. Primero, se ordenan según las distancias entre los descriptores de color y el descriptor de color del input. Luego, se ordenan según la distancia entre los descriptores de textura y el descriptor de textura del input.

Una vez realizados los ordenamientos, se recuperan primeras k imágenes de cada ordenamiento. El método recibe como parámetro cuántas imágenes se quieren recuperar y retorna, para cada ordenamiento, esa cantidad de imágenes, ordenadas de más a menos cercanas en términos de distancia euclidiana entre los descriptores.

El experimento se repitió para 500 imágenes seleccionadas al azar dentro del conjunto de prendas anotadas por color y lo mismo para el conjunto de prendas anotadas por textura. En cada iteración se calculó la cantidad de imágenes relevantes dentro de las primeras diez seleccionadas, llamando relevantes a aquellas imágenes que fueron de la misma clase que el input. Esta métrica es conocida como *precision@k*, siendo k el número de imágenes que se consideran.

5.3.4.2. Resultados generales

- precision@10 promedio para imágenes anotadas por color: 0.90
- precision@10 promedio para imágenes anotadas por textura: 0.89

5.3.4.3. Ejemplos de buen desempeño para colores

5.3.4.4. Ejemplos de buen desempeño para texturas

5.3.4.5. Ejemplos de mal desempeño para colores

5.3.4.6. Ejemplos de mal desempeño para texturas

5.3.4.7. Discusión de los resultados

Las visualizaciones muestran las imágenes recuperadas a partir de los conjuntos de prendas anotadas por color y textura. Los resultados de este experimento indican que, en la mayoría de los casos, las primeras imágenes recuperadas efectivamente tienen el mismo atributo que el de la imagen del input, ya sea color o textura. Además, se puede ver que, en algunos casos, solo las primeras imágenes tienen ambas características similares, mientras que las siguientes son similares en solo uno de los dos aspectos. Esto es razonable, ya que, para cada imagen del conjunto de datos, no necesariamente hay 36 prendas cuyo patrón y color coincidan.

En los casos en los que la técnica no dio buenos resultados, se puede ver que, en varias ocasiones, el error se debe a una similitud entre las clases. Por ejemplo, en los ejemplos en los que la técnica no funcionó bien recuperando prendas del mismo color, se puede ver que se confunden prendas rosadas con prendas moradas y rojas, prendas grises con prendas celestes o prendas amarillas con prendas de color café claro.

En otros casos en los que funcionó mal es cuando la característica de la clase no se encontraba tan claramente presente en la imagen del input. Esto se ve reflejado claramente en algunos de los ejemplos de mal desempeño para recuperar imágenes de textura similar.

De todas formas, observando el desempeño general del método, se obtuvo un valor promedio de *precision@10* de 0.92 para el caso de colores y de 0.89 para el caso de las texturas. En otras palabras, en promedio, 9 de las 10 primeras imágenes recuperadas por el método correspondían a la clase del input, lo que muestra el potencial de esta técnica para una eventual aplicación en buscadores por imágenes.

5.4. Resumen del capítulo

Los primeros experimentos de visualización, en los que se usó la red entrenada con el conjunto de datos Fashion Product Images de Kaggle, mostraron que si bien la red era capaz de clasificar correctamente las prendas de vestir según qué tipo de prenda eran, los canales de los bloques intermedios de la red no se especializaron en extraer atributos relacionados al color ni la textura de las prendas. Como el objetivo de este trabajo es utilizar la salida de los bloques intermedios para clasificar prendas según atributos como color y textura, se decidió utilizar otro modelo entrenado con ImageNet, el cual contiene clases de un contexto mucho más amplio que la ropa.

Los resultados de los experimentos de visualización y análisis de activaciones realizados con el segundo modelo mostraron que, en este caso, sí existían canales que capturaban información sobre color y textura. En particular, se confirmó que las capas más superficiales del modelo extraían de mejor forma la información relacionada al color de las prendas, mientras que los canales de las capas más profundas detectaban texturas con mayor exactitud.

En coherencia con estos resultados, los experimentos que utilizaban bloques intermedios para clasificar prendas según color y textura mostraron que el método es eficaz para extraer los atributos seleccionados utilizando solo unos pocos ejemplos por clase, logrando además mantener un buen desempeño en espacios con dimensiones reducidas usando UMAP.

Finalmente, el experimento de recuperación de imágenes basado en activaciones de bloques intermedios mostró que es posible utilizar la salida de estos bloques para encontrar imágenes parecidas al input, tanto en color como textura.

Capítulo 6

Conclusiones

Los resultados de los experimentos muestran que es posible determinar en qué secciones de la red se detectan diferentes tipos de características. Las capas más superficiales demostraron ser mejores clasificando colores, mientras que las más profundas obtuvieron una clara ventaja detectando los patrones de la tela de las prendas. Uno de los resultados más reveladores de este trabajo fue que el primer modelo que se entrenó, aquel que clasificaba prendas de vestir, tuvo un muy mal desempeño detectando colores y patrones. Luego de los análisis cualitativos de sus activaciones, resultó evidente que el modelo no estaba utilizando esas características para hacer sus predicciones, motivo por el que modelo acabó siendo reemplazado por un modelo entrenado sobre ImageNet.

Estas observaciones sirvieron para clarificar el hecho de que, para encontrar canales especializados en detectar un cierto tipo de característica, es crucial que esas características le sumen poder discriminativo al modelo. De otra forma, el proceso de optimización de la red no tiene incentivos para aprender a detectarlos y, por lo tanto, la red no contendrá neuronas que se activen frente a ellos. Este razonamiento fue corroborado por los resultados mucho mejores obtenidos al usar una red entrenada sobre ImageNet, modelo al que sí le sirve reconocer texturas y colores.

Más generalmente, lo anterior permite ver que, si se quieren utilizar las capas ocultas de un modelo entrenado para resolver un problema de clasificación, primero se debe realizar un análisis cualitativo de las activaciones. Incluso si el dominio de ambos problemas el mismo, como lo fue en el caso de este trabajo (donde el modelo original se entrenó para clasificar ropa y el propósito nuevo también era aquel), puede darse el caso de que el modelo que intentamos reutilizar no utilice características propias del dominio para hacer sus predicciones. Como se vio en los resultados, el modelo clasificador de ropa aprendió más bien a detectar características de las personas que vestían las prendas y a intuir en base a ellas la clase del input.

Una vez se determina la aptitud de un modelo para ser reutilizado, el proceso de atribuir a canales específicos de las capas ocultas la detección de características de interés comienza exponiendo al modelo a diferentes estímulos. Si bien este proceso requiere de datos y de procesamiento, que es lo que se quería evitar, la cantidad de ejemplos necesaria para hacer los análisis es muy inferior a la que se requiere para entrenar un modelo desde cero. Además, los resultados obtenidos mediante algunas de las técnicas que se evaluaron mostraron tener

un excelente desempeño, superando en algunos casos un 90 % de exactitud. Llegar a esta cifra teniendo menos de 100 ejemplos por clase puede ser algo muy conveniente.

La ventaja de esta metodología frente a las técnicas de transferencia de aprendizaje es que no es necesario ningún proceso de entrenamiento adicional. Esto es especialmente importante para aplicaciones cuyos modelos deben actualizarse con frecuencia, ya que permite para agregar nuevas clases sin tener que intervenir su arquitectura, lo que muestra cómo el método es fácilmente escalable.

En conclusión, en este trabajo se demostró que la metodología propuesta para la extracción de atributos permite abordar problemas de clasificación y recuperación de imágenes para los que se tienen pocos datos anotados, aprovechando el conocimiento contenido en modelos convolucionales entrenados en contextos más generales de clasificación.

Es importante mencionar que parte de este trabajo fue aceptado en la modalidad de Abstract Extendido en el workshop LXCV (LatinX at CV) de CVPR 2021, bajo el título de *Scalable Visual Attribute Extraction through Hidden Layers of a ResidualConvNet*.

Bibliografía

[1] I. Rafegas, M. Vanrella, L. A. Alexandre, and G. Arias, "Understanding trained cnns by indexing neuron selectivity," 2019.

[2] Z. Qin, "How convolutional neural networks see the world - a survey of convolutional neural network visualization methods," 2018.

[3] S. S. Adhikari, S. Singh, A. Rajagopal, and A. Rajan, "Progressive fashion attribute extraction," 2019.

[4] S. Lee, Y. Lee, J. Kim, and K. Lee, "Extraction of visual attributes from large-scale fashion dataset,"

[5] G. E. H. Alex Krizhevsky, Ilya Sutskever, "Imagenet classification with deep convolutional neural networks," 2012.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognitiony," 2015.

[7] S. A. G. S. E. W. Jie Hu, Li Shen, "Squeeze-and-excitation networks," 2019.

[8] P. D. Z. T. K. H. Saining Xie, Ross Girshick, "Aggregated residual transformations for deep neural networks," 2017.

[9] Q. V. L. Mingxing Tan, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019.

[10] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization.," *CVPR*, 2016.

[11] nickbiso, "Keras-Class-Activation-Map." https://github.com/nickbiso/Keras-Class-Activation-Map/blob/master/Class%20Activation%20Map(CAM).ipynb, 2008. [Online; accessed 19-July-2008].

[12] P. Aggarwal, "Fashion product images dataset." https://www.kaggle.com/paramaggarwal/fashion-product-images-dataset, 2019.

[13] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer Science Business Media, 2011.

[14] G. Cybenko, *Mathematics of Control, Signals, and Systems*. 1989.

[15] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, "Subject independent facial expression recognition with robust face detection using a convolutional neural network," *Neural Networks*, vol. 16, no. 5, pp. 555–559, 2003. Advances in Neural Networks Research: IJCNN '03.

[16] *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[17] I. Shafkat, "Intuitively understanding convolutions for deep learning." https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1, 2018.

[18] D. Mishra, "Translational invariance vs translational equivariance." https://towardsdatascience.com/translational-invariance-vs-translational-equivariance-f9fbc8fca63a, 2020.

[19] M. Yani, S. Irawan, and M. S.T., "Application of transfer learning using convolutional neural network method for early detection of terry's nail," *Journal of Physics: Conference Series*, vol. 1201, p. 012052, 05 2019.

[20] D. Soni, "Translation invariance in convolutional neural networks." https://divsoni2012.medium.com/translation-invariance-in-convolutional-neural-networks-61d9b6fa03df, 2019.

[21] Y. Li, J. Yosinski, J. Clune, H. Lipson, and J. Hopcroft, "Convergent learning: Do different neural networks learn the same representations?," 2016.

[22] V. Roman, "Cnn transfer learning fine tuning." https://towardsdatascience.com/cnn-transfer-learning-fine-tuning-9f3e7c5806b2, 2020.

[23] M. Rivera, "La red residual." http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/resnet/resnet.html, 2019.

[24] "Plotting umap results." https://umap-learn.readthedocs.io/en/latest/plotting.html.

[25] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020.

[26] L. McInnes, "Understanding umap." https://pair-code.github.io/understanding-umap/, 2018.

[27] "Gridcut: Improving grid layouts with minimal graph cuts," 2021.