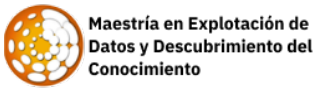
Transfer Learning, Fine-Tuning, Feature Extraction y Clustering con red VGG16.

Trabajo de especialización.



Autor: Nahuel Alba

Institución: Universidad de Buenos Aires

Fecha: 25 de febrero de 2025

Índice

1. Resumen

El presente trabajo de especialización de la Maestría en Explotación de Datos y Descubrimiento del Conocimiento tiene como un objetivo usar redes neuronales convolucionales aplicadas a problemas de clasificación multiclase y clustering.

El objeto de análisis es el dataset "Apparel Images Dataset" CC0: Public Domain, creado por Aleksandr Antonov [1].

Específicamente, se aplica *transfer learning* a partir de una red VGG16 preentrenada, a tal efecto se modifica la estructura de la red agregando una capa de clasificación especifica para el problema que se aborda. Se analizó la *performance* del modelo para la clasificación del nuevo problema planteado, a través de las medidas *loss* y *accuracy*, evaluando primero la red con todas sus capas congeladas, para luego realizar en segunda instancia *fine-tuning* descongelando las dos capas fully connected anteriores a la última capa de clasificación, evaluando sus resultados y ajustando finalmente el *lerning rate* logrando resultados con un *accuracy* máximo de 0.84 en test. Una vez encontrado su punto de máxima *performance* para el problema planteado, se utilizó la misma red entrenada para realizar extracción de características con su penúltima capa *fully connected* y con el vector (array numpy unidimensional) de 4096 dimensiones. Sobre dicho vector se aplicaron múltiples algoritmos de reducción de dimensionalidad y clusterización evaluando varias estrategias para encontrar clusters sólidos, llegando a clusters con buenos valores de Silhouette global mediante el uso de UMAP en conjunto con k-medoids (PAM) y DBSCAN que permitieron identificar características latentes que exceden a las etiquetas de clasificación.

2. Introducción

2.1. Motivación

La motivación del presente trabajo es por un lado, responder a la pregunta de si la red VGG16 al haber sido entrenada con sólo 45 de 1000 categorías relativas a prendas, puede lograr una clasificación apropiada de imágenes de un indumentaria, y por otro lado si puede clusterizar las imágenes del dataset en clusters que revelen supercategorías comunes (latent features) entre

[1] Aleksandr Antonov *https://www.kaggle.com/datasets/trolukovich/apparel-images-dataset*, License CC0: Public Domain

las prendas. Se busca responder ambas incógnitas mediante una aplicación proficiente de las técnicas enseñadas a lo largo de la cursada 2024 de las materias de Análisis Inteligente de Datos, Data Mining en Ciencia y Tecnología y Aprendizaje Automático. Específicamente:

- **Análisis Inteligente de Datos:**
  - Reducción de Dimensionalidad con PCA.
  - Clustering Jerárquico.
  - Coeficiente Cofenético como medida de validación interna para técnicas de Clustering Jerárquico.
  - Matrices de distancia.
- **Data Mining en ciencia y tecnología:**
  - Feature Extraction con VGG16.
  - Reducción de dimensionalidad con UMAP.
  - Algoritmo de clusterización K-Medoids y DBSCAN.
  - Silhouette como medida de validación interna.
  - Estadístico de Hopkins.
- **Aprendizaje Automático:**
  - Train-validation-test Split.
  - TensorFlow para manipulación de la estructura de la red VGG16 (capas Softmax, capas ReLU).
  - Funciones de pérdida y métricas de evaluación (Loss functions, Accuracy).

2.2. Antecedentes

Se tuvieron en cuenta como antecedentes para el presente trabajo además del material creado por las cátedras antedichas:

- **"Deep Learning for Computer Vision Starter Bundle" (Adrian**

Rosenbrock):

- Consolidación de conocimientos impartidos en la cátedra de Aprendizaje Automático.
- Profundización en aspectos clave de redes neuronales convolucionales aplicadas al procesamiento de imágenes.
- Estudio de la estructura del píxel, escalamiento y aspect ratio.
- Análisis de las partes de una red convolucional.
- Configuración básica de TensorFlow.
- Papel de las funciones de pérdida (Cross-Entropy Loss).
- Importancia de la visualización de la estructura de la red en TensorFlow.

- **"Very Deep Convolutional Networks for Large-Scale Image Recognition" (Karen Simonyan, Andrew Zisserman):**
  - Comprensión del funcionamiento de la red VGG16.
  - Identificación de capas convenientes para realizar fine-tuning en el caso concreto.
- **"Deep Learning for Computer Vision Practitioner Bundle" (Adrian Rosenbrock):**
  - Presenta una implementación de la VGG16, aunque no enfocada en transfer learning.
  - Útil para implementar estrategias relativas al manejo del learning rate en diferentes épocas.
- **"Comprehensive Survey of Clustering Algorithms" (Dongkuan Xu, Yingjie Tian):**
  - Análisis de diferentes medidas de distancia como alternativas a la Euclideana en espacios de alta dimensionalidad.
  - Estudio en profundidad del funcionamiento del algoritmo de clasificación DBSCAN.

- **"Análisis inteligente de datos con lenguaje R" (Débora Chan, Cristina Badano, Andrea Rey):**
  - Referencia para el análisis del algoritmo de clustering jerárquico.
  - Evaluación de diferentes tipos de coeficientes cofenéticos.
- **"Statistics in a Nutshell" (Sarah Boslaugh):**
  - Teoría de test de chicuadrado McNemar: supuestos e implmentación.

3. Objetivos

Los objetivos de este trabajo se centran en el desarrollo y evaluación de técnicas de trarnsfer learning con una red neuronal convolucional VGG16 para la clasificación de prendas de ropa. En primer lugar, se busca optimizar su desempeño en la tarea de clasificación, asegurando un adecuado entrenamiento y ajuste del modelo mediante técnicas de fine-tuning. Además, se propone utilizar la red entrenada para realizar feature extraction de las imágenes con el fin de capturar representaciones relevantes del conjunto de datos. Finalmente, dichas features serán utilizadas para la aplicación de algoritmos de clusterización, buscando la formación de clústeres robustos y significativos.

4. Metodología

Para el desarrollo del trabajo se utilizo el dataset "Apparel images dataset" de 11385 imágenes con 24 categorías. Cada categoría (Cuadro 1) corresponde a un tipo de prenda distinta entre las cuales están:

Categoría	**Porcentaje del total (%)**	**Cantidad**
black dress	3.95	450
black pants	7.65	871
black shirt	6.28	715
black shoes	6.73	766
black shorts	2.88	328
blue dress	4.41	502
blue pants	7.01	798
blue shirt	6.51	741
blue shoes	4.59	523
blue shorts	2.63	299
brown pants	2.73	311
brown shoes	4.08	464
brown shorts	0.35	40
green pants	1.99	227
green shirt	2.02	230
green shoes	4.0	455
green shorts	1.19	135
red dress	7.03	800
red pants	2.71	308
red shoes	5.36	610
white dress	7.18	818
white pants	2.41	274
white shoes	5.27	600
white shorts	1.05	120

Cuadro 1: Distribución de categorías de prendas y su porcentaje en el dataset.

Sobre dicho dataset, primero se procedió a preprocesar las imágenes que lo integran a través de la función "load_img" que incorpora la librería keras para la carga del archivo. Esta función transforma la imagen a formato .pil, luego se la transforma en un array de numpy y luego load_img modifica la forma del array de la fotografía para que cumpla con el input especifico necesitado para ingresar a la red VGG16 convirtiendo la imagen de "RGB" (canales RED-GREEN-BLUE) a "BGR" (canales BLUE-GREEN-RED) con dimensiones 224 de ancho por 224 de largo (224x224 BGR).

Una vez preprocesado la totalidad del dataset, se procedió a dividirlo en dos datasets de entrenamiento y testeo en una proporción de 70 % para train y 30 % para test. Cabe destacar que luego en el entrenamiento se utilizó el 20 %

del dataset utilizado en train para validación.

La red VGG16 (Figura 1) posee en primer lugar una parte plenamente convolucional de 13 capas donde se usan solamente bloques convolucionales de dimensiones “3x3”, apilados uno sobre otros en cada vez mayor profundidad, a los efectos de capturar características cada vez más abstractas de la imagen. La reducción de las dimensiones de la imagen a lo largo de cada una de las capas se produce en las capas “max pooling”. Estas capas (de “2x2” con stride 2) reducen progresivamente la resolución espacial de la imagen mientras mantienen la información relevante de los filtros convolucionales, hasta antes de llegar a la segunda parte de la red convolucional, donde luego del ultimo max pooling de “7x7x512”, se transforman los tensores en un vector de características unidimensional a través de la capa Flatten. En este vector, cada elemento del mismo representa una dimensión de la observación. En segundo lugar, la red VGG16 posee una parte integrada por dos capas fully connected y un clasificador “sofmax”. Consecuentemente, la primera de estas últimas capas fully connected con 4,096 nodos recibe un vector de dimensiones “7x7x512 = 25088”. Luego, se pasa a otra capa fully connected también de 4096 nodos donde esta penúltima capa finalmente se conecta a una capa de clasificación softmax. Esta última capa, al haber sido la red entrenada para el problema de clasificación que plantea el dataset “ImageNet”, está configurada para la clasificación de 1000 categorías.

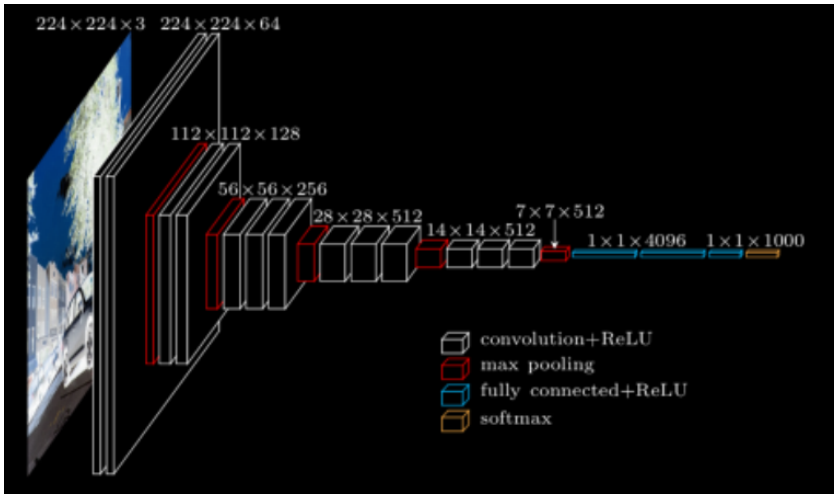


Figura 1: Estructura VGG 16

Teniendo en cuenta las estructura de la red, resulta claro que este modelo de aproximadamente 138 millones de parámetros no es un modelo “liviano”

y que genera un alto consumo de memoria RAM o VRAM. Consecuentemente, si se pretendiesen implementar el modelo para casos de inferencia en tiempo real en despositivos con recursos limitados o en aplicaciones web, hay opciones más apropiadas, como implementaciones de redes convolucionales como*MobileNet, SqueezeNet o EfficientNet-lite*.

A partir de esto se adapta la última capa de clasificación softmax para el problema de clasificación presentado en el dataset "Apparel Images Dataset", modificando la estructura de la red, removiendo la última capa y utilizando una capa dense con activación "softmax" con 24 neuronas correspondientes a las 24 tipo de categorías del dataset.

El segundo eje del presente trabajo, busca clusterizar diferentes grupos de categorías presentes en el dataset "Apparel images dataset", partiendo de las representaciones latentes de las imágenes del dataset.

Para llegar a dichas representaciones latentes y obtener buenas representaciones de clustering, se utilizó como base la CNN VGG16 para hacer feature extraction de las imagenes, utilizando el output de la penúltima capa (segunda capa fully-connected) en forma de vector con 4076 dimensiones. Luego, se aplicaron varios algoritmos de reducción de dimensionalidad (PCA, UMAP) a los efectos de facilitar la obtención de resultados concluyentes en el proceso de clusterización, y finalmente se aplicaron los algoritmos de clasificación, jerárquico, k-medoids (PAM) y DBSCAN, buscando clusters sólidos validados por el coeficiente de Hopkins y por el coeficiente de Silhouette como técnica de validación interna.

El coeficiente de Hopkins será calculado con la siguiente fórmula de manera tal que valores cercanos a 1 indican alta tendencia a la clusterización.

$$H = \frac{\sum d(z, \mathrm{NN}(z))}{\sum d(z, \mathrm{NN}(z)) + \sum d(x, \mathrm{NN}(x))}$$

5. Resultados

5.1. Transfer Learning VGG16

Centrándonos en el primer eje de este trabajo. Para el primer entrenamiento se entrenó dejando congeladas todas las capas para ver el desempeño de la red sobre el dataset. Específicamente, se setearon los callbacks `CSVLogger` y `EarlyStopping` a los efectos de detener el entrenamiento cuando no ha-

ya mejora por 5 epochs seguidos con el parámetro `restore_best_weights` activo para volver la red al epoch que mejor desempeño obtuvo. Se utilizan los hiperparámetros `epochs=10`, `batch_size=64`, y un optimizador `ADAM` con `learning_rate=1e-4`.

En este primer entrenamiento (Figura 2) se obtuvieron los siguientes valores:

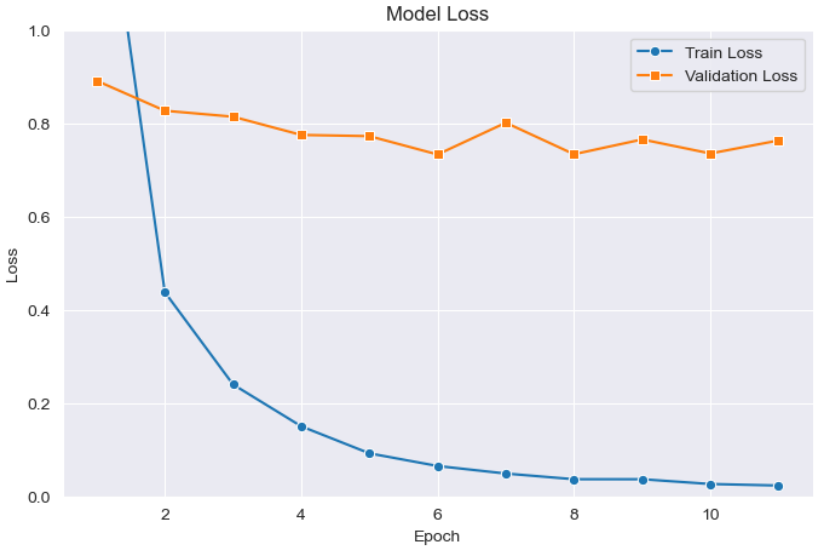


Figura 2: Loss Function - Primer entrenamiento

Así, podemos observar que el modelo alcanza su mejores valores en el epoch 6, momento a partir del cual se produce la bifurcación entre el train y validation de manera que en validation se entiende que el modelo comienza a sobreajustar.

Cuadro 2: Primer entrenamiento valores epoch 6

Métrica	**Valor**
Val Accuracy	0.7933
Val Loss	0.7331
Test Loss	0.6814
Test Accuracy	0.7961

Se observa en principio (Cuadro 2) buena capacidad de generalización con todas las capas congeladas pero dejando lugar a la mejora que puede operar mediante el fine-tuning.

En un segundo entrenamiento (Figura 3), se empieza el proceso de fine-tuning, descongelando las últimas dos capas dejando los hiperparámetros del modelo en iguales valores con respecto al epoch, batch size, y learning rate, obteniendo los siguientes valores:

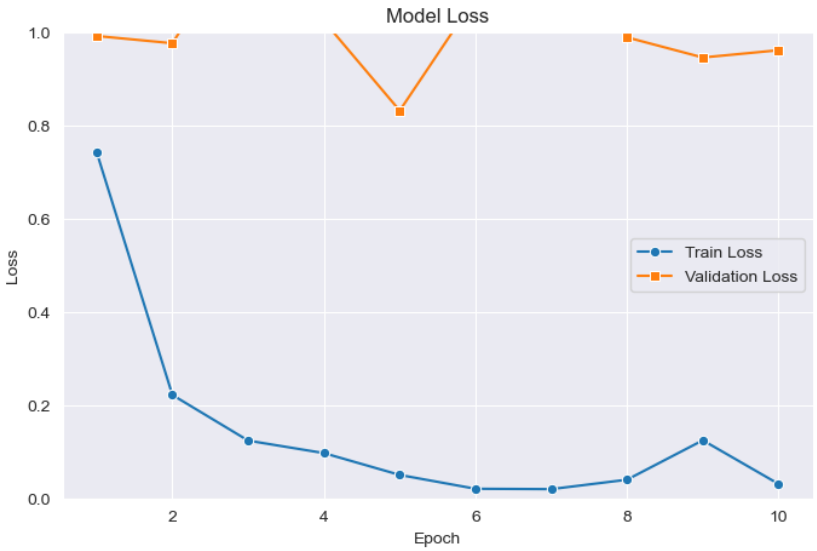


Figura 3: Loss Function - Segundo entrenamiento

Se puede apreciar en la figura 3 que la bifurcación se produce en el epoch 5 obteniendo los valores más bajos de validación. Asimismo se observa (Cuadro 3) un aumento de los valores de loss tanto en test como en validación pero con una mejora del accuracy.

Cuadro 3: Segundo entrenamiento valores epoch 5

Métrica	**Valor**
Val Accuracy	0.8367
Val Loss	0.8314
Test Loss	0.9024
Test Accuracy	0.8224

Finalmente se modifican los hiperparámetros por última vez más bajando el learning rate a los efectos facilitar la convergencia a un mínimo local que pudiese mejorar los resultados. Obteniendo los siguientes resultados:

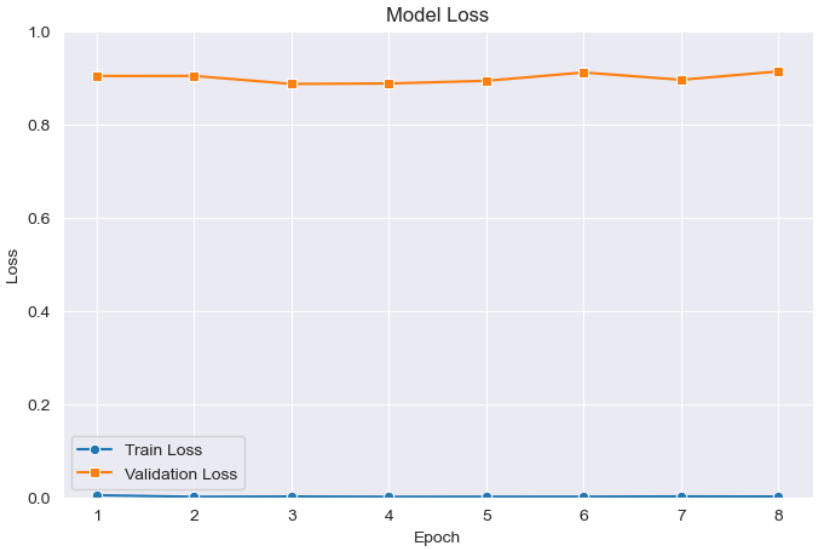


Figura 4: Loss Function - Tercer entrenamiento

Se observa (Figura 4) que el modelo alcanza su mejor validación en el epoch 3 con val_accuracy = 0.8500 y val_loss = 0.8865. A partir del epoch 4, el modelo comienza a sobreajustar, ya que el loss en training sigue bajando, pero el loss en validación se mantiene constante o empeora.

.

Cuadro 4: Resultados de validación y test

Métrica	**Valor**
Val Accuracy	0.8467
Val Loss	0.8932
Test Loss	0.9875
Test Accuracy	0.8418

De esta manera, se visualiza (Cuadro 4) que el accuracy mejora pero a costas de un loss en test más alto. Lo cual implica que la red "pierde certeza" sobre el resultado correcto, de manera que la probabilidad que corresponde a la etiqueta correcta disminuye.

Finalmente, se procedió a comparar el desempeño del primer modelo con el modelo final al que se le aplico fine-tuning y manipulación del hiperparámetro learning-rate, a través del test de McNemar.

El test de McNemar "es un tipo de prueba de chi-cuadrado utilizada cuando los datos provienen de muestras pareadas, también conocidas como muestras emparejadas o muestras relacionadas(...). Por ejemplo, podríamos usar el test de McNemar para examinar los resultados de una encuesta de opinión sobre algún tema antes y después de que un grupo de individuos vea un anuncio político. En este caso, cada persona aportaría dos opiniones: una antes y otra después de ver el anuncio. No podemos tratar ambas opiniones sobre el mismo tema como independientes, por lo que no podemos utilizar un chi-cuadrado de Pearson. En su lugar, asumimos que dos opiniones recogidas de la misma persona estarán más relacionadas entre sí que dos opiniones recogidas de personas distintas."[2].

En el caso concreto el test de McNemar tiene sentido porque compara el mismo conjunto de datos con dos modelos diferentes, específicamente compara dos clasificadores en términos de errores cometidos en las mismas muestras. El test evalúa si la diferencia entre b y c (Cuadro 5) es significativa. Si hay una diferencia significativa, significa que un modelo es mejor que el otro en la misma muestra.

	Modelo B Correcto (1)	**Modelo B Incorrecto (0)**
Modelo A Correcto (1)	a (Ambos correctos)	c (A correcto, B incorrecto)
Modelo A Incorrecto (0)	b (A incorrecto, B correcto)	d (Ambos incorrectos)

Cuadro 5: Tabla de contingencia para el Test de McNemar

A razón de ello se corre el test antedicho obteniendo un estadístico con valor 7.9670 y un p-valor de 0.004. Este resultado sugiere que el modelo fine-tuned mejora estadísticamente significativa respecto al modelo base.

5.2. Clustering

Esta sección se focaliza en el segundo eje de este trabajo centrado en realizar extracción de features de las imágenes del dataset utilizado y tomar dichas features para aplicar algoritmos de clusterización.

El primer intento infructuoso, fue intentar aplicar técnicas de clusterización partiendo de una reducción de dimensionalidad utilizando PCA (Principal Component Analysis), teniendo como objetivo lograr el 95 % de varianza acumulada en sus componentes. Producto de ello, se obtuvieron 1454 com-

[2]Sarah Boslaugh *"Statistics in a Nutshell"*, p. 134 *(Traducción propia)*

ponentes principales mostrando (Figura 5) baja capacidad de los embeddings para aglomerar la varianza de las imágenes.

Figura 5: Componentes principales

Se remarca que cada embedding aglomera poca varianza siendo en el componente principal 1 de 0.047. Se infiere de esto que los vectores procesados por la CNN se caracterizan por una falta de linealidad entre sus features que no es susceptible de ser correctamente captada por PCA. Esto, será posteriormente ratificado al comparar los valores obtenidos con técnicas aptas para relaciones no lineales en los datos como UMAP.

No obstante, se analizo el coeficiente de Hopkins obteniendo valores de 0.9123, motivo de ello se infiere una tendencia a la clusterización y por ende se procedió a aplicar algoritmos de clusterización.

Se realizó una clusterización jerárquica utilizando una matriz de distancias basada en la métrica euclidiana. Los coeficientes cofenéticos obtenidos (Cuadro 6) indican que el método de enlace centroidal es la mejor opción.

Cuadro 6: Coeficientes Cofenéticos por método con medida de distancia Euclidiana

Método	**Coeficiente Cofenético**
Complete	0.8810
Average	0.8698
Weighted	0.7772
Centroid	0.8862
Median	0.7587
Ward	0.5606

Se elige en consecuencia como mejor método, el centroidal y se genera el clustering jerárquico. Se procede entonces, a analizar el número idóneo de clusters a partir del SSE (suma de errores cuadrado) que se calcula como la suma de las distancias al cuadrado entre cada punto dentro de un cluster y su centroide (Figura 6). Específicamente, se utiliza el "Elbow Method" o método del codo para analizar cuando los valores de SSE y sus cambios de cluster a cluster dejan de ser significativos. Asimismo, se usa el Silhoutte score promedio donde también se hace uso el método del codo, para analizar el punto donde la mejora en la separación entre clusters deja de ser significativa (Figura 7).

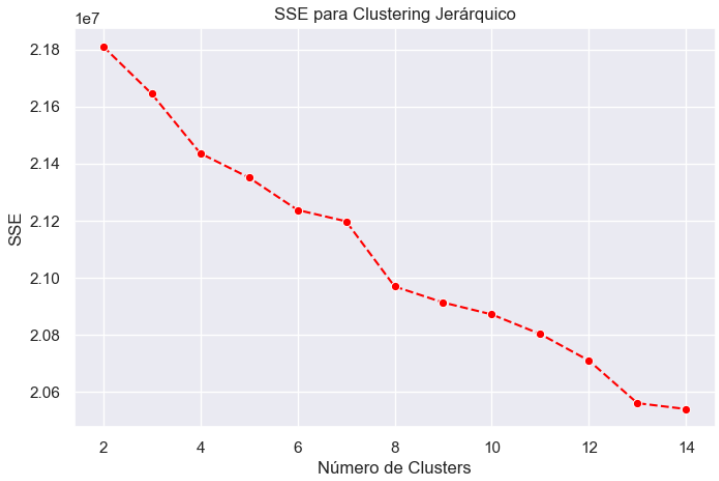


Figura 6: SSE para clustering jerárquico (PCA)

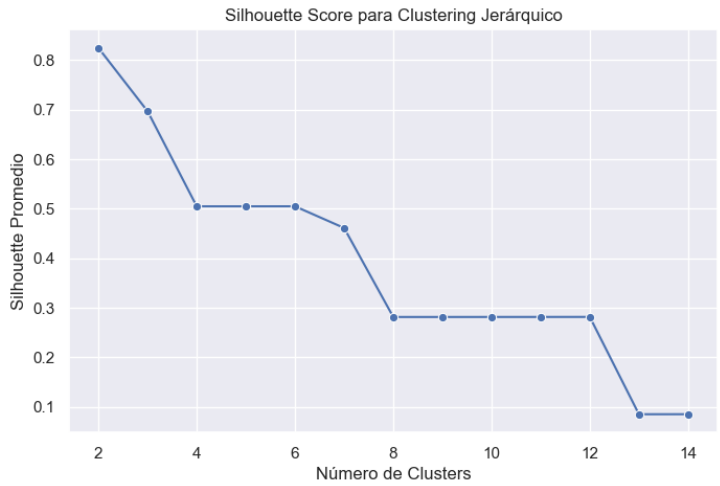


Figura 7: Silhouette Score para clustering jerárquico (PCA)

Desde el gráfico de Silhouette (Figura 7) se infiere que podría ser un buen punto de corte 4 en la medida que continuar separando los clusters no aporta diferencias significativas, sin embargo si miramos el gráfico de SSE (Figura 6) el punto idóneo debería ser 7 u 8.

Si tomamos como punto de corte el inferido del gráfico de Silhouette y analizamos la conformación de los clusters tomando cuatro como punto de corte obtenemos el gráfico de la Figura 8.

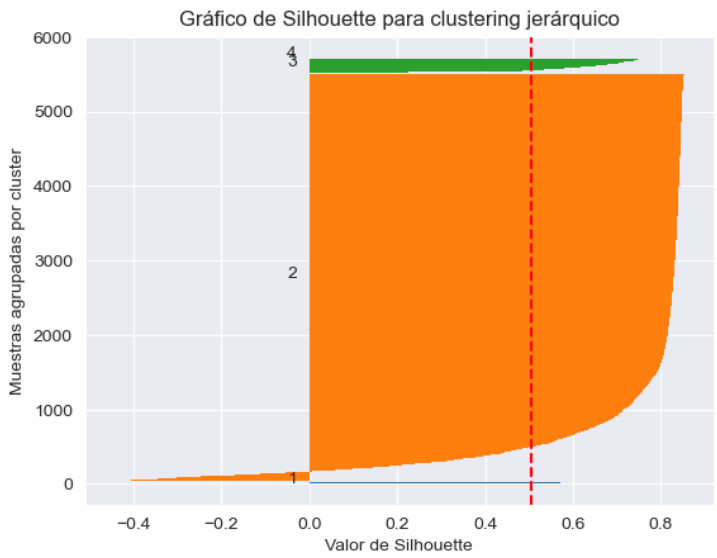


Figura 8: Silhouette Score para clustering jerárquico con distancia Euclidiana, corte en 4 (PCA)

Podemos observar entonces que la distribución de los clusters resulta altamente desigual con un cluster 2 que aglutina la mayoría de valores y 3 clusters de tamaño mínimos con valores que no llegan o apenas alcanzan a pasar los valores de Silhouette global.

Por otro lado, si consideráramos al gráfico de SSE (Figura 6), tomando 8 clusters como punto de corte tenemos a la figura 9 mostrando la composición de los clusters.

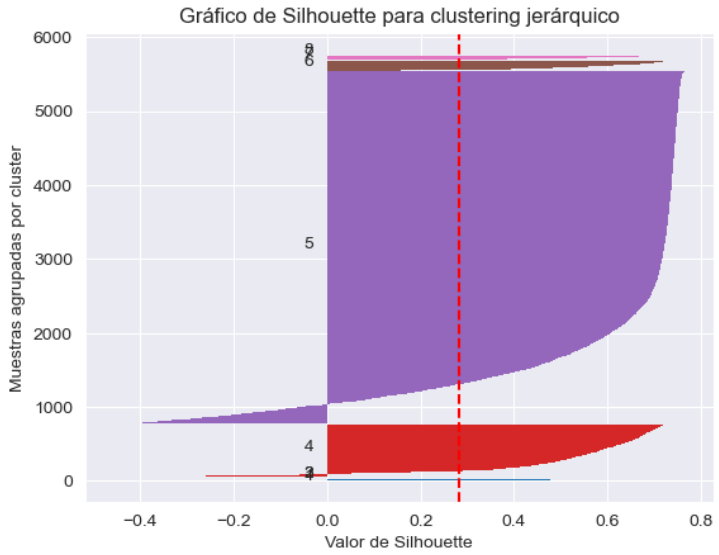


Figura 9: Silhouette Score para clustering jerárquico con distancia Euclidiana, corte en 8 (PCA)

En este caso, si bien en algunos clusters se sobrepasa el puntaje global de Silhouette seguimos encontrando el problema de que se observa una desigualdad significativa en la cantidad de muestras por cluster, lo que puede afectar la estabilidad del clustering. Asimismo, cabe remarcar que el valor de Silhouette promedio global es 0.2813, lo cual es un valor bajo que no índica clusters correctamente conformados.

Motivo de ello, se infiere que no resulta un método idóneo para clusterizar el presente dataset. Se intentó asimismo la clusterización jerárquica partiendo de otras matrices de distancia como Manhattan (Cuadro 7) o Chebyshev (Cuadro 8), pero se infiere que por la alta dimensionalidad ninguna llevó a clusters coherentes.

Cuadro 7: Coeficientes Cofenéticos para Matriz de Distancia Manhattan

Método	**Coeficiente Cofenético**
Single	0.6885
Complete	0.6618
Average	0.7447
Weighted	0.5990
Centroid	0.7235
Median	0.6290
Ward	0.3890

No se observan de esta manera (Cuadro 7) mejores resultados para la matriz de distancia calculada con la medida de distancia Manhattan y sus correspondientes coeficientes cofenéticos.

Se probo como medida de distancia Chebyshev, obteniendo los valores consignados en el Cuadro 8. El máximo coeficiente cofenético obtenido a partir de la matriz de distancias calculada con la medida de distancia antedicha, fue el coeficiente cofenético para complete: 0.8630.

Cuadro 8: Coeficientes Cofenéticos para Matriz de Distancia Chebyshev

Método	**Coeficiente Cofenético**
Single	0.7261
Complete	0.8631
Average	0.8552
Weighted	0.8181
Centroid	0.8567
Median	0.8325
Ward	0.6545

Si bien los resultados obtenidos a partir de la matriz de distancia de Chebyshev resultaban prometedores, se observaron los mismos problemas que con la distancia euclidiana una vez elegido el punto de corte óptimo. En consecuencia, los clusters presentan una conformación irregular en el gráfico (Figura 10), no logrando superar los valores de Silhouette promedio global.

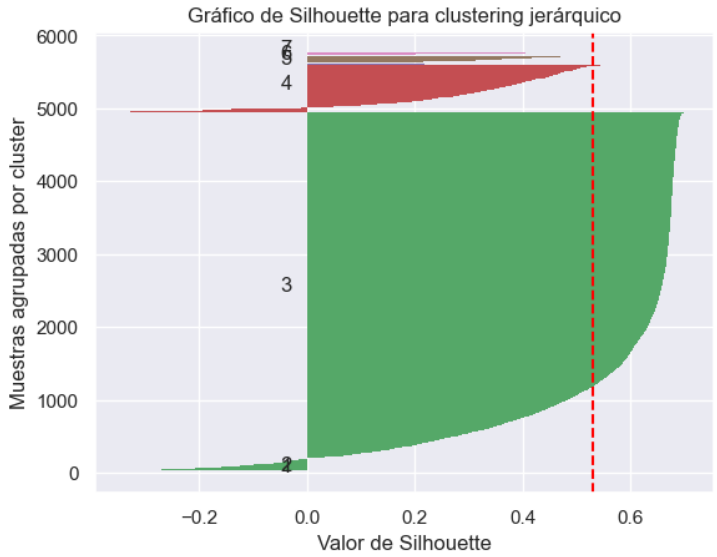


Figura 10: Silhouette Score para clustering jerárquico, corte en 8 matriz de distancias Chebishev (PCA)

Con motivo de los resultados obtenidos al aplicar PCA y clustering jerárquico, se infiere entonces que PCA no resulta una método de reducción de dimensionalidad idóneo para captar las relaciones no lineales de las features extraídas de las imágenes por la VGG16.

Se utilizó entonces UMAP (Uniform Manifold Approximation and Projection) que construye un grafo de los datos en alta dimensión, y luego optimiza una representación en baja dimensión cuyo grafo sea lo más similar posible al de alta dimensión. UMAP preserva la estructura del manifold, lo que lo hace superior a PCA en ciertos casos, siendo un algoritmo más apropiado para captar relaciones no lineales entre las dimensiones de las features extraídas de las imágenes.

Este algoritmo se empleó en conjunto con DBSCAN (Density-Based Spatial Clustering of Applications with Noise), algoritmo que "define clusters como regiones continuas de baja densidad (...) por cada instancia, el algoritmo cuenta cuantas instancias están localizadas dentro de una pequeña distancia (épsilon) de esta. Esta región de la instancia es llamada e-neighborhood . Si una instancia tiene al menos un determinado número (parametro "min_samples") de instancias en su e-neighborhood (incluyéndose a si misma), entonces se considera como una core instance (semilla). En otras palabras, las semillas son aquellas localizadas en regiones densas. Todas las instancias en un neigh-

borhood de una semilla pertenecen al mismo cluster. Esto puede incluir otras instancias semilla, formando entonces una secuencia de instancias semillas en un mismo cluster. Cualquier distancia que no es una instancia semilla y no tiene una en su neighborhood es considerada una anomalía. Este algoritmo funciona bien si los clusters son lo suficientemente densos y están bien separados en regiones de baja-densidad" [3]

Entonces, al poder DBSCAN detectar clusters con formas complejas que en alta dimensión serían difíciles de identificar, sin ser necesario además especificar la cantidad de ellos, se lo elige como primera opción para usar junto con UMAP. Asimismo, para facilitar el desempeño de DBSCAN y buscar características latentes en los datos que pudiesen exceder a las etiquetas se configuró UMAP en sus parámetros buscando lograr mejores comunidades locales que fuesen susceptibles de ser fácilmente clusterizadas. Sus parámetros fueron: n_neighbors=70; min_dist=0.10; n_components=100; metric='cosine'; random_state=68).

Cabe destacar que se utilizo la métrica "cosine" dado que el algoritmo por default utiliza la euclideana. La medida de distancia euclidiana, se centra en la diferencia absoluta entre valores de los vectores, viéndose entonces afectada por la magnitud de los valores, lo que puede hacer que dos imágenes similares tengan distancias altas solo por diferencias en escala. Es decir, una imagen con muchos bordes y texturas generará activaciones más fuertes que una imagen uniforme. Un fondo oscuro en comparación con un fondo claro puede hacer que ciertos filtros respondan más o menos.

Por ende, no todas las dimensiones del embedding son activadas por igual. Algunas dimensiones del embedding pueden ser muy activas para algunas imágenes y casi 0 para otras. Esto significa que el vector resultante tendrá diferente magnitud dependiendo de la imagen de entrada. Pudiendo presentarse la situación de que las imágenes que generen embeddings con valores grandes estarán más alejadas, aunque sean visualmente similares. Teniendo en consideración lo dicho, la medida de distancia cosine resulta mejor opción dado que solo importa la dirección del vector, no su magnitud. Esto, evita que imágenes similares se separen artificialmente solo porque sus activaciones fueron más altas o más bajas.

Por otro lado, el parámetro "n_neighbors" de UMAP determina cuántos veci-

[3]Aurélien Géron *"Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlowÇoncepts, Tools, and Techniques to Build Intelligent Systems*, p. 256 *(Traducción propia)*

nos cercanos se consideran al construir el grafo de similitud en alta dimensión antes de reducir la dimensionalidad. Por ello, se eligió el valor 70 debido a que valores menores a 100 (n_neighbors=70, en el caso concreto) y distancias mínimas cercanas a 0 en el parámetro min_dist (min_dist=0.10 en el caso concreto), favorecen la creación de comunidades locales.

Una vez que se aplico el algoritmo sobre el dataset de features extraídas por la red VGG 16, se analizó el coeficiente de Hopkins obteniéndose valores favorables de 0.9846. Se remarca que devuelve en este análisis mejores valores que los obtenidos con los embeddings de PCA (0.9123)

Así entonces, la aplicación de ambos algoritmos dio como resultado los clusters representados en la Figura 11 y 12.

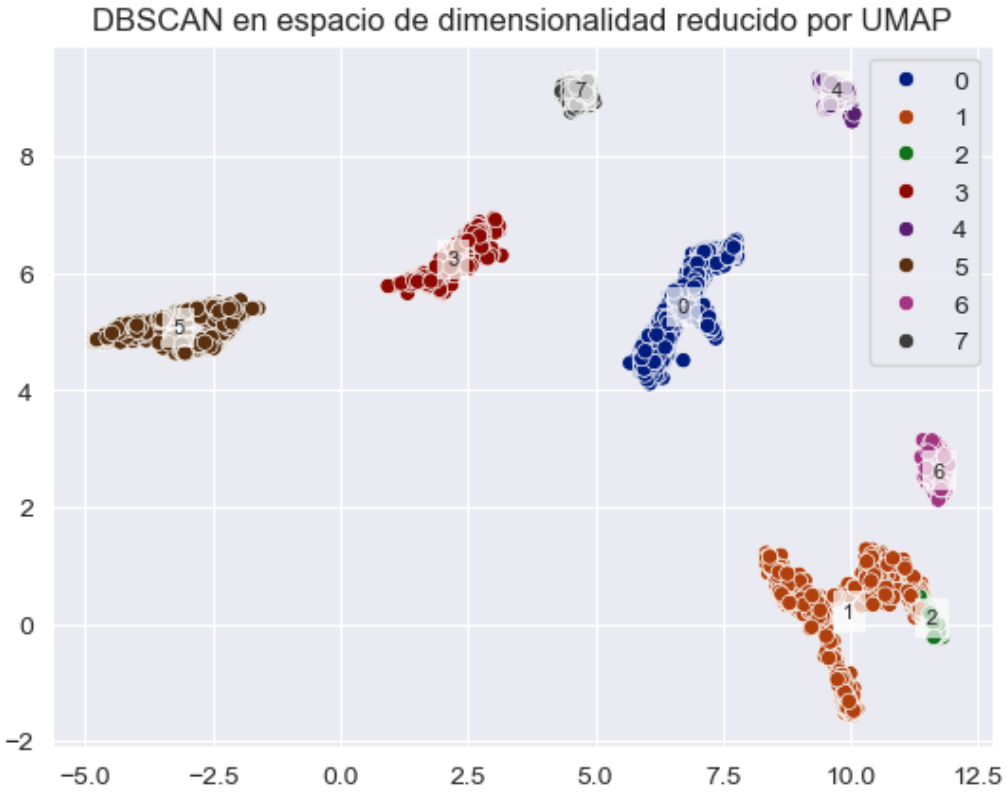


Figura 11: DBSCAN en espacio de dimensionalidad reducido bidemensional por UMAP

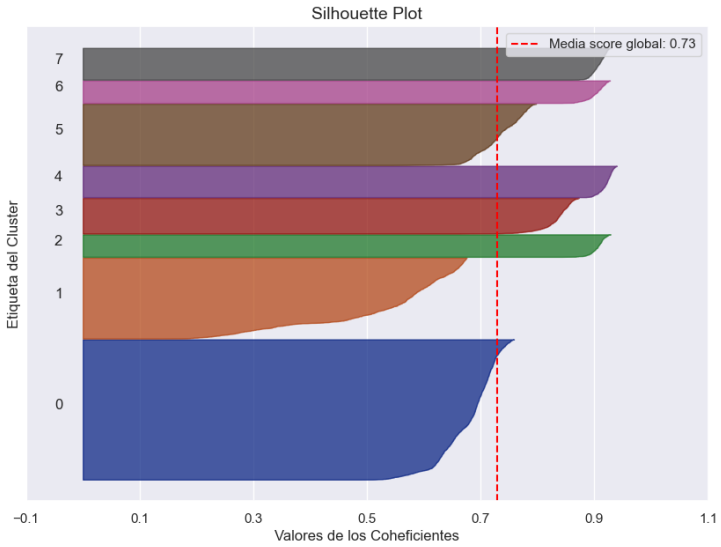


Figura 12: Gráfico Silhouette para clusters obtenidos por DBSCAN

Encontramos entonces una buena clusterización con valores globales de Silhouette de 0.73 (Figura 12), lo cual resulta indicativo de la mejor clusterización encontrada. Con clusters con forma más uniformemente distribuidos a excepción del cluster uno que sin embargo presenta valores de Silhouette altos (superiores a 0.5) en la mayoría de sus puntos, razón por la cuál todos los clusters resultan internamente validados.

Finalmente, se procedió a la construcción de una matriz de distancias a partir de los embeddings generados por UMAP a los efectos de ver si con esta técnica de dimensionalidad reducida podían devolver clusters más sólidos que con PCA y por ende se construyó una matriz de distancias en base a la medida de distancia euclidiana la cuál devolvió los clusters analizados en las figuras 13, 14 y 15.

Figura 13: Gráfico SSE para clusters obtenidos por K-Medoids

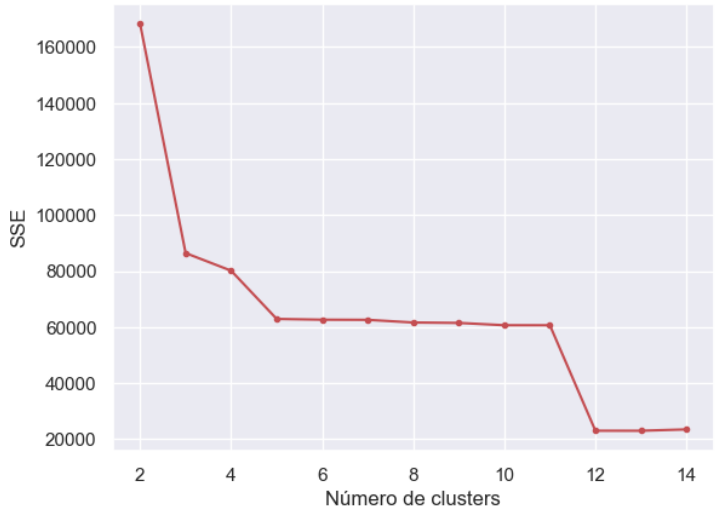


Figura 14: Gráfico Silhouette para clusters obtenidos por K-Medoids

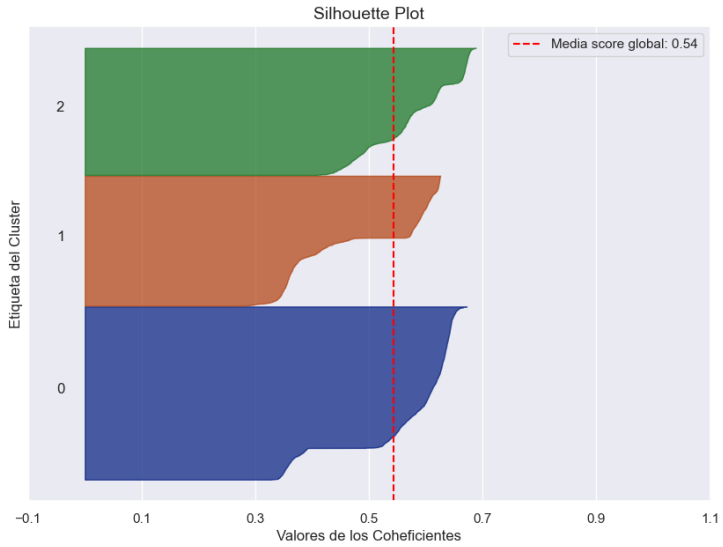


Figura 15: Gráfico Silhouette K-Medoids

Habiendo elegido como punto de corte 3 para la conformación de los clusters por k-medoids, teniendo en cuenta que es el punto con máximo puntaje global de Silhouette (Figura 12) y donde se "quiebra" el codo del gráfico de SSE (Figura 13), llegamos a 3 clusters correctamente conformados pero con un puntaje de global de Silhouette de 0.54 (Figura 15) no tan alto como la clusterización realizada con DBSCAN (0.73) pero igualmente sólidos, en la medida que la mayoría de los puntos pasan el coeficiente global y la mayoría de las observaciones en los clusters poseen una gran cantidad de observaciones con valores de Silhouette lejanos a 0.

5.3. Insights sobre la composición y perfilado de los clusters

Para analizar la composición de los clusters se procedió a agrupar las observaciones del dataset según la categorías asignadas por los algoritmos de clustering. Específicamente, se analizó dentro de cada cluster el porcentaje de categorías reales (labels pertenecientes a cada tipo de prenda) para elaborar conclusiones de qué tipo de características latentes comunes aglomera cada cluster.

5.3.1. Análisis de clusters DBSCAN

Primero, se analiza la composición de los clusters obtenidos por DBSCAN partiendo de los embeddings de UMAP .

Cuadro 9: Distribución de Clases en el Cluster 0

Clase	**Porcentaje (%)**
Black Pants	0.23
Blue Pants	0.21
Black Shorts	0.09
Brown Pants	0.08
Red Pants	0.08
Blue Shorts	0.08
White Pants	0.07
Green Pants	0.06
Green Shorts	0.04
White Shorts	0.03
Brown Shorts	0.01
Black Shoes	0.01

El **cluster 0** (Cuadro 9) agrupa principalmente prendas inferiores, es decir, pantalones y shorts, sin centrarse en el color. La red parece haber captado patrones estructurales o de silueta que distinguen a las prendas inferiores de otras categorías, mostrando que la forma (alargada) es una feature latente importante. El hecho de incluir mínimas proporciones de otros items (como black_shoes) podría deberse a similitudes en texturas o bordes, pero el dominante es el patrón de pantalones/shorts.

Cuadro 10: Distribución de Clases en el **Cluster 1**

Clase	**Porcentaje (%)**
Black Shoes	0.34
Blue Shoes	0.23
Brown Shoes	0.21
Green Shoes	0.20

El **Cluster 1** (Cuadro 10) se centra en zapatos, donde la forma y estructura características del calzado son determinantes. La red ha captado la silueta y

los detalles propios de los zapatos, diferenciándolos de prendas con cortes o texturas distintas, como pantalones o vestidos.

El **cluster 2** contiene 0.98 por ciento de imágenes de la clase red_shoes. La alta homogeneidad en cuanto a la clase (red_shoes) indica que, además de las características de forma del calzado, el color rojo es un factor muy distintivo en la representación latente. Aquí, se observa una especialización en una combinación muy concreta de forma y color.

El **cluster 3** contiene 0.48 por ciento de imágenes de la clase blue_dress, 0.46 por ciento de imágenes de la clase black_dress, Este cluster agrupa vestidos, en particular en tonos azul y negro. La similitud en la forma (la silueta de un vestido) es la feature dominante, mientras que las variaciones de color se mantienen en una gama cercana, lo que sugiere que la red ha aprendido a identificar el patrón estructural de vestidos y luego ha segmentado ligeramente por color.

El **cluster 4** contiene 0.97 por ciento de imágenes de la clase white_dress, la casi exclusividad de vestidos blancos señala que el color es un atributo muy fuerte en la representación latente para este cluster. La red ha logrado aislar una combinación específica: la forma del vestido junto con la característica del color blanco.

Cuadro 11: Distribución de Clases en el **Cluster 5**

Clase	**Porcentaje (%)**
Blue Shirt	0.44
Black Shirt	0.42
Green Shirt	0.14

En el **cluster 5** (Cuadro 11) se agrupan camisas. La red parece haber captado detalles propios de esta prenda, como la presencia de cuellos, botones o patrones de manga, que la distinguen de otros tipos de ropa. Aun cuando hay variaciones de color, la estructura de la prenda es lo que predomina.

El **cluster 6** contiene 0.96 por ciento de imágenes de la clase white_shoes, similar al cluster 2 pero para zapatos de color blanco, se evidencia que la red ha captado una combinación muy precisa de forma de calzado y el atributo cromático (blanco), lo que permite que este cluster se especialice en una clase muy concreta.

El **cluster 7** contiene 0.94 por ciento de imágenes de la clase red_dress, 0.03 por ciento de imágenes de la clase blue_dress. Este cluster se centra en vestidos predominantemente rojos. La pequeña proporción de blue_dress podría representar casos fronterizos, pero claramente el color rojo es el factor definitorio junto con la forma característica del vestido.

En conclusión, el análisis de los 8 clusters sugiere que la red VGG16 ha aprendido a representar la ropa no solo en función de su categoría semántica (pantalones, zapatos, vestidos, camisas), sino también en función de atributos visuales finos, tales como la forma, la textura y el color. Esto permite una separación clara entre diferentes tipos de prendas y, dentro de cada tipo, la diferenciación basada en características cromáticas específicas.

5.3.2. Análisis de clusters K-medoids (PAM)

Finalmente se analiza el caso de los clusters encontrados con k-medoids.

El **cluster 0** (Cuadro 12) agrupa como posible característica latente a las prendas inferiores (pantalones y shorts), lo cual indica que las features latentes extraídas capturan de forma destacada características estructurales como la silueta alargada y ciertos patrones de textura o forma asociados a estas prendas.

Cuadro 12: Distribución de Clases en el **Cluster 0**

Clase	**Porcentaje (%)**
Black Pants	0.19
White Dress	0.18
Blue Pants	0.17
Black Shorts	0.07
Brown Pants	0.07
Red Pants	0.07
Blue Shorts	0.06
White Pants	0.06
Green Pants	0.05
Green Shorts	0.03
White Shorts	0.03
Brown Shorts	0.01
Black Shoes	0.00

Así el **cluster 1** (Cuadro 13) contiene principalmente imágenes de camisas

y vestidos. Incluye colores como rojo, azul, negro y verde. No hay presencia de pantalones ni zapatos. Posible característica latente: prendas de la parte superior del cuerpo (camisas y vestidos) con predominio de colores oscuros y saturados (negro, rojo, azul, verde).

Cuadro 13: Distribución de Clases en el Cluster 1

Clase	**Porcentaje (%)**
Red Dress	0.23
Blue Shirt	0.21
Black Shirt	0.21
Blue Dress	0.14
Black Dress	0.13
Green Shirt	0.07

Finalmente, el **cluster 2** (Cuadro 14) está compuesto exclusivamente por imágenes de zapatos, diferenciados solo por el color. La alta coherencia intra cluster indica que la red ha aprendido a capturar con precisión las características estructurales y relativas a la textura que definen al calzado, diferenciándose de las de prendas superiores o inferiores. La variación cromática se mantiene dentro de un mismo grupo, lo que refuerza la idea de que la forma, el contorno y otros detalles específicos del calzado son atributos latentes muy distintivos.

Cuadro 14: Distribución de Clases en el Cluster 2

Clase	**Porcentaje (%)**
Black Shoes	0.22
Red Shoes	0.18
White Shoes	0.18
Blue Shoes	0.15
Brown Shoes	0.14
Green Shoes	0.13

6. Conclusiones

La red VGG16 resulta un instrumento idóneo para realizar transfer learning en el problema de clasificación de prendas de vestir. A pesar de haber sido entrenada con 45 de 1000 categorías en prendas, la red a logrado consolidar

filtros que resultan útiles para el problema de clasificación planteado en el dataset utilizado en el presente trabajo. Se logran buenos resultados para el dataset sin descongelar ninguna capa y mejora su accuracy al hacer fine-tuning. Se deja como premisa para futuras investigaciones lograr valores más bajos de loss utilizando técnicas de ajuste progresivo del learning rate como por ejemplo "ReduceLROnPlateau", que es implementado en la librería TensorFlow.

Asimismo, la red permite extraer las features de cada imagen de manera tal de constituir inputs útiles para lograr clusters sólidos con algoritmos de clusterización. Si bien resultan viables los dos algoritmos de clusterización utilizados con UMAP (DBSCAN y K-medoids), resulta particularmente útil a los efectos de interpretación de las características latentes los resultados obtenidos por k-medoids. Consecuentemente, estos clusters permiten descubrir categorías latentes como prendas superiores, prendas inferiores y calzados que de esta manera congloban información congruente, permitiendo eventualmente solucionar nuevos problemas de semi-supervised learning o supervised-learning.

7. Referencias

[1] Boslaugh, S. (2012). *Statistics in a nutshell* (2nd ed.). O'Reilly Media.

[2] Chan, D., Badano, C., & Rey, A. (2019). *Análisis inteligente de datos con lenguaje R*. Editorial de la Universidad Nacional de La Plata.

[3] Rosenbrock, A. (2020). *Deep learning for computer vision starter bundle*. PyImageSearch.

[4] Rosenbrock, A. (2020). *Deep learning for computer vision practitioner bundle*. PyImageSearch.

[5] Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition*. arXiv preprint arXiv:1409.1556.

[6] Xu, D., & Tian, Y. (2015). *A comprehensive survey of clustering algorithms*. Annals of Data Science, 2(2), 165-193.

8. Repositorio de respaldo

Se incluye link al repositorio de Github personal utilizado como respaldo a lo largo del presente trabajo de especialización. ***Clothes-Classification-with-CNNs***.