
Métodos y estrategias en la clasificación de imágenes

Agustín Herrera Daniel Kolodziejczyk Reinaldo Medina Rosana Daisi Mendez

María Eugenia Villegas

Abstract

Sobre una base de datos de 6899 imágenes de índole cotidiana de 8 tipos de objetos, se realiza un estudio exploratorio de métodos y estrategias supervisadas y no supervisadas para su clasificación. Se evalúan sin éxito algoritmos de clasificación no supervisados de clustering spectral y connected components. Se considera la estrategia de *transfer learning* utilizando la red neuronal VGG-16. Sobre las *features* obtenidas al aplicar esta red se exploran varios *pipelines* de métodos no supervisados obteniendo un *accuracy* máximo de 0,9799.

Adicionalmente, se ejecuta un novedoso enfoque de *data-augmentation* generando un *dataset* paralelo con imágenes distorsionadas con ruido gaussiano. Encontramos una robustez superlativa de la red neuronal comparado con otros algoritmos no supervisados. Su *accuracy* es superior a 0,9 en un *dataset* con una población 50 % de imágenes con ruido gaussiano de varianza 50, donde otros métodos no supervisados obtienen un *accuracy* de 0,5.

A su vez, se llevó a cabo un intento de segmentación y detección de objetos utilizando una imagen de una carrera, aplicando un filtro Gaussiano y binarización de la imagen, para proceder luego a la construcción de un grafo y aplicación de algoritmo de *clustering* espectral y *connected-component labelling*.

1. Introducción

Actualmente, un sinnúmero de instrumentos, cámaras y sensores recuperan información en forma de imagen. El volumen de imágenes generado implica la necesidad de su procesamiento en forma automática. Resulta clave en este procesamiento la identificación de características sobre las cuales clasificar las imágenes. Es conveniente, entonces, el uso de un modelo que permita quitarles la información irrelevante, y obtener *features* que sean independientes de su escala, rotación u orientación.

Es aquí, donde intervienen las redes convolucionales, que han sido las más exitosas en el reconocimiento de imágenes a gran escala, desde que hay disponibles grandes repositorios públicos de imágenes y sistemas de cómputo de alto rendimiento. Dentro de las redes convolucionales, VGG se caracteriza por tener una mayor profundidad, gracias al agregado de más capas convolucionales y al empleo de filtros convolucionales muy pequeños (3x3) en las capas. [5]

Esta red ha sido entrenada en el *dataset* ImageNet [1], que contiene 1000 clases de objetos. Integra la propuesta de nuestro trabajo aplicar este modelo previamente entrenado a un conjunto de datos más pequeño y de menor diversidad.

Las clases, o grupos de objetos similares, juegan un papel esencial en cómo analizamos y describimos el mundo. Tanto es así que intuitivamente somos capaces de dividir objetos en grupos (*clustering*) y asignar nuevos objetos a estos grupos (clasificación). Ya sea que las clases capturan una estructura subyacente en la realidad o que son arbitrarias, sin duda el *clustering* es una operación útil para resumir y comprimir información, y para realizar aproximaciones en cómputos. [7]

Si se dispone de las etiquetas de las imágenes, es posible poner a prueba distintos algoritmos de *clustering*, abriendo la discusión de cuál resulta más adecuado para un *dataset* con las características del que se está analizando. Esta clasificación supervisada puede ser evaluada con métodos de validación externa y así llegar a métricas que permitan estimar el éxito de la metodología en nuestro contexto particular de aplicación. Algunos de los algoritmos de *clustering* más usados, y que integran la presente comparación, son *k-means*, *k-medoids*, *clustering* jerárquico [8] [3].

Otros aspectos de significativa importancia dentro del procesamiento de imágenes son su segmentación y la detección de objetos [3]. Este trabajo los abordará, por ser casos de aplicación de algoritmos de *clustering* pero que, a su vez, pueden ser el paso previo a la aplicación de un algoritmo de clasificación. La propuesta es aplicar dos métodos distintos de detección de objetos ampliamente utilizados: *connected-component labelling* y *clustering* spectral, para así poder comparar su *performance*.

La calidad de las imágenes también constituye un factor relevante en el contexto de su procesamiento. Las redes neuronales como VGG-16 se entrena en *datasets* de alta calidad y, sin embargo, en el contexto de aplicación las imágenes suelen tener afectada su calidad [2]. Es conocido que las redes neuronales convolucionales pueden ser alteradas por una perturbación imperceptible, también que un mismo artefacto puede afectar específicamente a una red neuronal y no a otra. Szegedy et al encontraron que el mapeo de input-output de las redes neuronales profundas puede ser significativamente discontinuo [6]. Dada la relevancia que puede tener un error de clasificación, y la necesidad de obtener modelos parsimoniosos, consideramos evaluar la robustez de los modelos presentados, tanto su efecto en la predicción como su efecto al dañar el set de entrenamiento.

2. Métodos

Dataset

Para el presente trabajo se empleó un *dataset* previamente compilado de imágenes realistas de alta resolución divididas en 8 objetos bien diferenciados: natural-images (<https://www.kaggle.com/datasets/prasunroy/natural-images>) [4]. El *dataset* contiene 6899 imágenes de aviones (727), autos (968), gatos (885), perros (702), flores (843), frutas (1000), motocicletas (788) y personas (986).

Las imágenes del *dataset* variaban en resolución desde 43x114 hasta 1719x2655 píxeles. Las imágenes originales estaban en formatos ‘PNG’ (*Portable Network Graphics*) y ‘JPG’ (*Joint Photographic Experts Group*). Durante el proceso de preparación, todas las imágenes se convirtieron al formato de color RGB, con valores de píxeles en el rango de 0 a 225, garantizando consistencia en el tratamiento del color.

Cada imagen se redimensionó a 224x224 píxeles y se normalizaron los valores de los píxeles a un rango de 0-1. Este proceso aseguró que todas las imágenes fueran comparables en términos de color, valor, rango y tamaño. Para verificar el procesamiento, se seleccionó una imagen representativa de cada objeto, mostrando sus dimensiones y formato de color (Figura 1). La verificación final confirmó que todas las imágenes procesadas tienen un tamaño uniforme de 224x224 píxeles, asegurando la comparabilidad entre ellas.

Arquitectura de la red neuronal profunda

Se empleó la red neuronal VGG-16 [5], que es un modelo de *transfer learning* de 16 capas (con pesos) pre-entrenado para la clasificación de imágenes. Se empleará la implementación en **Keras** (<https://keras.io/api/applications/vgg/>). La entrada de esta red son imágenes RGB de 224x224 píxeles, y luego de las capas de convolución y *pooling*, se encuentran tres capas, dos de ellas con 4096 neuronas (*features*) y una capa *softmax* para clasificar. No se utilizará esta última capa (*softmax*), y se trabajará sobre las características (*features*) de las capas intermedias, que servirán de entrada cada uno de los algoritmos de *clustering* que se pondrán a prueba.

Algoritmos de *clustering*

El análisis de *clustering* y reducción de dimensionalidad se realizó completamente desde **Python**. Para la obtención de los *features* desde las imágenes originales se utiliza, específicamente, la librería **keras.applications.vgg16** y **tensorflow.keras.utils**, la que se utiliza en general para deep

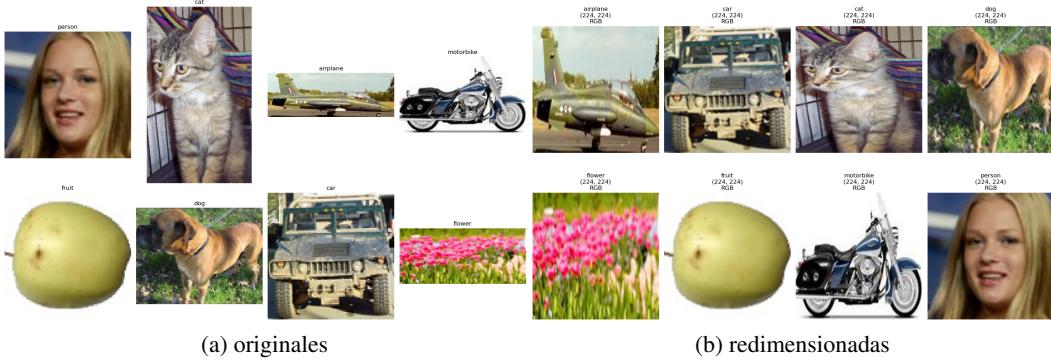


Figura 1: Imágenes del *dataset*, originales y redimensionadas a 224x224 pixeles

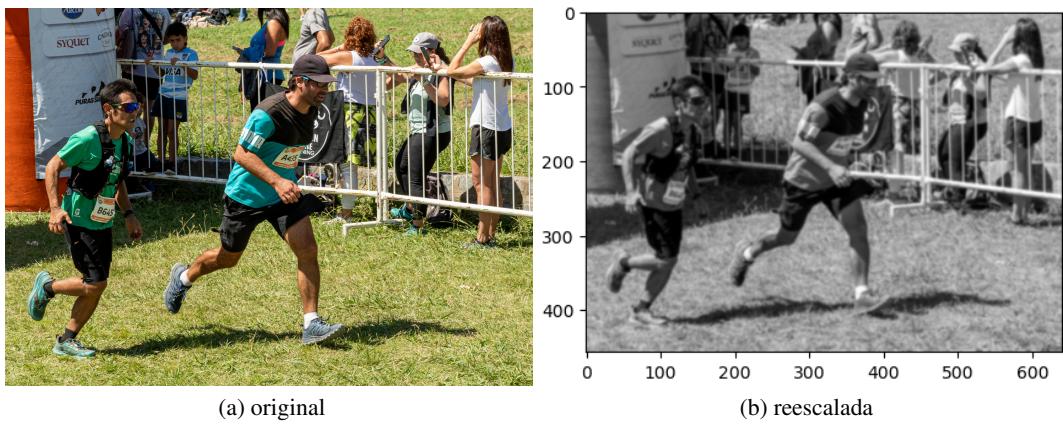


Figura 2: Imagen original (2a) y en escala de grises, reescalada y con filtro gaussiano (2b).

learning. Los algoritmos de clustering se implementan desde **sklearn.cluster** (*k-means* y pruebas con DBSCAN), **sklearn_extra.cluster** para *k-medoids* y **scipy.cluster.hierarchy** para *clustering* jerárquico.

Las reducciones de dimensionalidad se llevan a cabo usando **sklearn.decomposition** para PCA, **sklearn.manifold** para t-SNE (y pruebas con MDS) y la librería **umap** para UMAP. Acompañan estas operaciones otras herramientas como **cv2** para visualización y evaluación de imágenes, **sklearn.metrics** para métricas como Indice de Rand, matrices de confusión, análisis de Silhouette, etc.

Detección de objetos

Se llevó a cabo la segmentación y detección de objetos utilizando una fotografía de una carrera (Figura 2a). Este tipo de imagen es interesante debido a la cantidad de figuras y el fondo complejo, lo cual presenta un buen desafío para el método de *clustering* spectral. El proceso se dividió en varias etapas: la carga y preprocesamiento de la imagen, la aplicación de filtros y umbrales, la construcción de un grafo de similaridades y, finalmente, la segmentación mediante *clustering* espectral.

Primero, se procedió a cargar la imagen, la cual fue convertida a escala de grises para facilitar el procesamiento. Esta conversión simplifica los datos y reduce el ruido que podría afectar la calidad de la segmentación. La imagen original tenía un tamaño de 1143x1600 píxeles, pero para su mejor procesamiento, se redujo al 40 % de su tamaño original, quedando en 457x640 píxeles.

Para mejorar la detección de los objetos presentes en la imagen, se aplicó un filtro Gaussiano que la suaviza, reduciendo el ruido y resaltando las estructuras más significativas (Figura 2b). El filtro gaussiano aplicado es de sigma = 3. Posteriormente, se utilizaron diferentes umbrales para binarizar la imagen. Se probaron umbrales de 90 y 100.

Luego se construyó un grafo donde cada píxel es un nodo y las conexiones entre los nodos representan las disimilaridades de intensidad entre los píxeles vecinos. Utilizando una función exponencial decreciente, se transformaron estas disimilaridades en medidas de similaridad, ajustando los parámetros para enfocarse en similaridades más exigentes. El enlace entre los nodos se determinó según el gradiente de intensidades, transformando estas disimilaridades en medidas de similaridad.

Para aplicar *spectral clustering* se realizaron testeos sobre imágenes sin filtro, con filtro gaussiano y con imágenes binarizadas.

Finalmente, se aplicó el algoritmo de *clustering* espectral para segmentar la imagen en diferentes regiones. Se predefinió el número de regiones a segmentar y se utilizó el método *k-means* para asignar las etiquetas a cada región. Se comprobó que el *clustering* espectral es un algoritmo que lleva mucho tiempo de procesamiento. Esto conllevó a la reevaluación del tamaño de la foto, la cuál fue reducida para obtener diversidad de resultados. La misma se redujo a 114x160 píxeles.

Parámetros

Para imágenes binarizadas, con método *k-means*, se emplearon los parámetros beta = 40, eps = $1e - 6$, eigen tol = $1e - 4$, regiones = 10. Se empleó el método *discretize* con los mismos parámetros.

Para las imágenes en escala de grises sin filtro gaussiano los parámetros empleados fueron beta = 30, eps = $1e - 6$, eigen tol= $1e - 7$, regiones=15, método=*k-means*. Para las imágenes en escala de grises con filtro gaussiano los parámetros empleados fueron beta = 80, eps = $1e - 6$, eigen tol= $1e - 7$, regiones = 8, método=*k-means*.

Librerías de Python utilizadas

pillow 9.4.0, numpy 2.0.0, scikit-learn 1.5.1, scikit-image 0.24.0, matplotlib 3.8.4

Red neuronal

La red neuronal fue definida utilizando las librerías Tensorflow 2.10. Contiene una secuencia de capas densas de activación ReLu de 256, 128, 32 neuronas y finalizando con una capa densa de 8 neuronas con activación *softmax*.

En cada implementación fue entrenada de la misma forma con un optimizador adam, perdida 'sparse_categorical_crossentropy', batch_size 64 y 20 epochs. La primera capa y sus subsiguientes sufrieron una regularización por dopout de 30, 10 y 10 %.

Se generaron 9 datasets paralelos al original (Figura 3) con las mismas imágenes distorsionadas con ruido gaussiano de media cero y varianza creciente en steps de 10 (de 0 hasta 90). Cada aplicación de ruido gaussiano fue randomizada, sin repetir el ruido en ninguna de las aplicaciones. En cada experimento se generó un nuevo set de imágenes combinando las imágenes originales y un set de imágenes distorsionadas con ruido gaussiano de una misma varianza. Los sets de test y train fueron divididos al azar en proporción 20-80, las imágenes originales y sus correspondientes imágenes distorsionadas fueron tomadas de pares.

Se compararon tres *pipelines* de clasificación:

1. VGG-16 + *k-means* (8 clusters)
2. VGG-16 + UMAP (100 vecinos, 10 componentes) + *k-means* 8 clusters
3. VGG-16 + red neuronal

Los tres *pipelines* fueron ajustados y testeados con los datos originales y distorsionados de los datasets.

3. Resultados y discusión

En principio, y al margen de tener el número de grupos asignados, se obtuvo por coeficiente de Silhouette, y por error cuadrático medio (SSE) (Figura 4) que entre 7 y 8 *clusters* respectivamente se describirían mejor las agrupaciones. Hecho que se vislumbra notoriamente cuando se realiza una reducción de la dimensionalidad tanto con PCA o t-SNE (Figura 5).



Figura 3: Superior izquierda: imagen del *dataset* sin cambios. Subsiguiente imágenes distorsionadas con ruido gaussiano de media cero y varianza creciente.

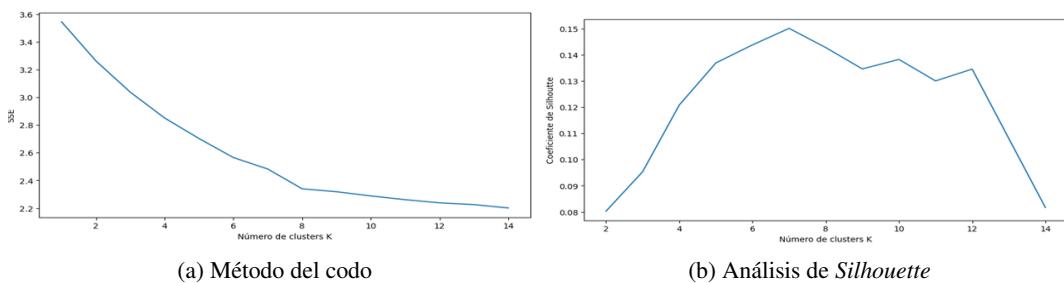


Figura 4: Análisis de la suma de errores cuadráticos (SSE) y coeficiente de *Silhouette* para *clustering* con k-means

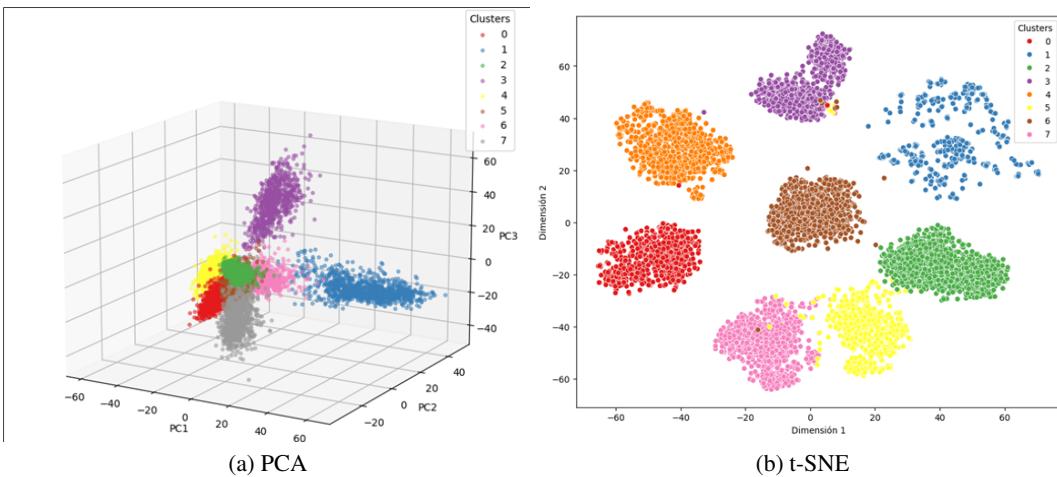


Figura 5: Visualización de reducción de dimensionalidad para los 4096 *features* obtenidos desde VGG-16. PCA (5a) y t-SNE (5b). *cluster* 0: motocicletas, *cluster* 1: frutas, *cluster* 2: personas, *cluster* 3: aviones, *cluster* 4: autos, *cluster* 5: perros, *cluster* 6: flores, *cluster* 7: gatos

A partir de los 4096 *features* extraídos por VGG-16, con un *array* de 4096x6899 se planteó un análisis de *clustering* para 8 *clusters* con diferentes algoritmos, la predicción sobre las imágenes y su correspondiente validación. La apariencia globular de las agrupaciones vistas en la Figura 5

corroboran los resultados logrados con *k-means*, con una asignación muy satisfactoria (Figura 6a) como lo evalúa su ARI (0.98) e índice de van Dongen (0.01) del Cuadro 1.

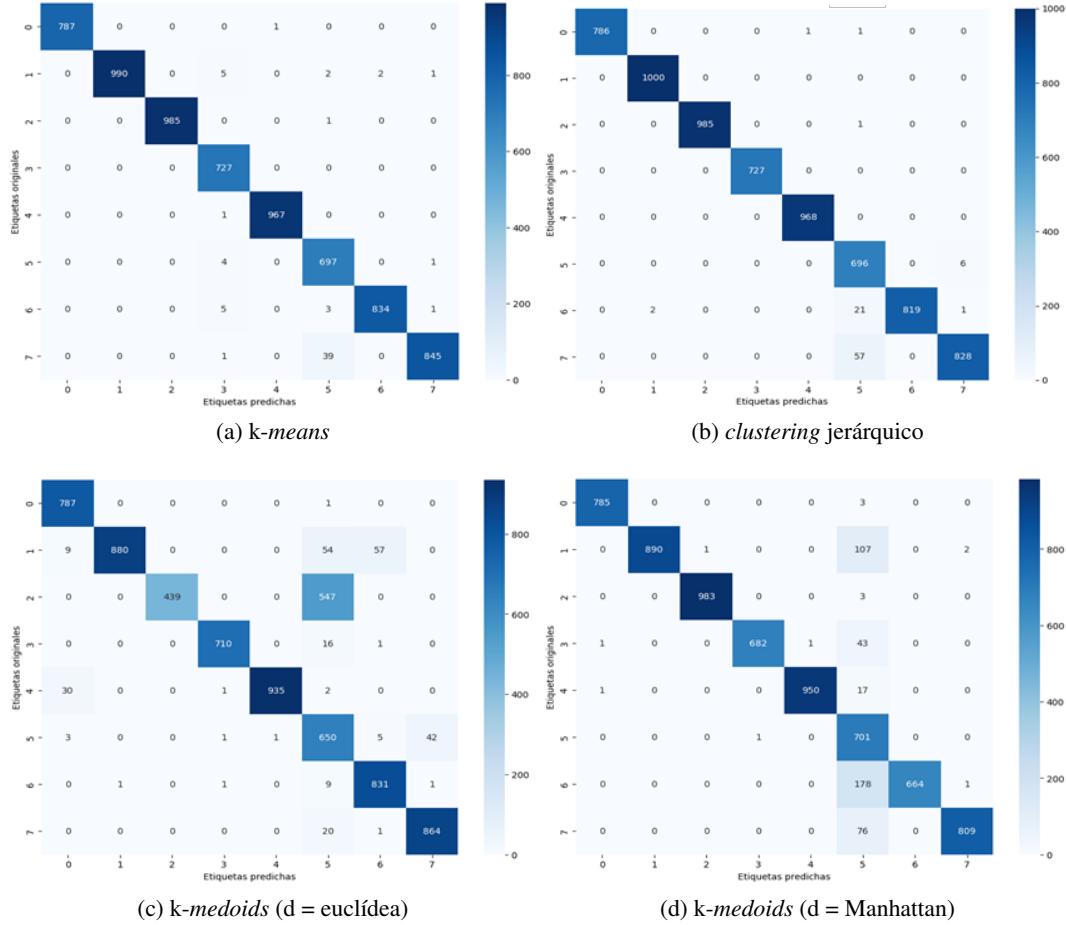


Figura 6: Matrices de confusión para evaluación de las predicciones realizadas por los algoritmos de *clustering* empleados. 6a por *k-means*, 6b por *clustering jerárquico*, 6c por *k-medoids* usando distancia euclídea, 6d *k-medoids* usando distancia Manhattan

Si bien las imágenes utilizadas no representan una correspondencia anidada o algún tipo de escalafón que las relacione, responden muy bien a la clasificación por *clustering jerárquico* (Figura 6b) con métricas comparables a *k-means* (Cuadro 1) resultando en un dendograma claro y totalmente legible cuando se usa distancia euclídea y método de enlace (o de ligamiento) “Ward” (Figura 7).

De una tercera propuesta usando *k-medoids* se obtuvieron resultados que, si bien no dejan de ser satisfactorios, no están a la par de los dos modelos anteriores y particularmente este método mejora sus predicciones cuando se usa la distancia Manhattan (Figura 6c, 6d) en especial cuando se refiere a identificar personas (*cluster 2*), donde clasifica a más de la mitad como perros, con distancia euclídea (Figura 6c), y se lleva a sólo el 10 % esa cantidad cuando se cambia a distancia Manhattan (Figura 6d).

Dentro de los modelos de mejor predicción, al margen de presentar un error aleatorio en las fotos mal clasificadas, si se observa las matrices de confusión de la Figura 6, *k-means* acentuó sus desaciertos en un grupo de fotos de gatos (*cluster 7*) clasificados como perros (4.5 % de las fotos del grupo gatos), mientras que en el *clustering jerárquico* fue de un 6.5 %. Este hecho se puede corroborar visualmente en la Figura 5 cuando se usa t-SNE, acerca de la interacción entre los clústeres 5 y 7.

Otra particularidad para destacar en cuanto a *k-medoids* es que lo disgregado de la apariencia del *cluster 1* (frutas) cuando se aplica reducción de dimensionalidad por t-SNE (Figura 5) se refleja en las predicciones de esta categoría, ya sea usando distancia euclídea o Manhattan.

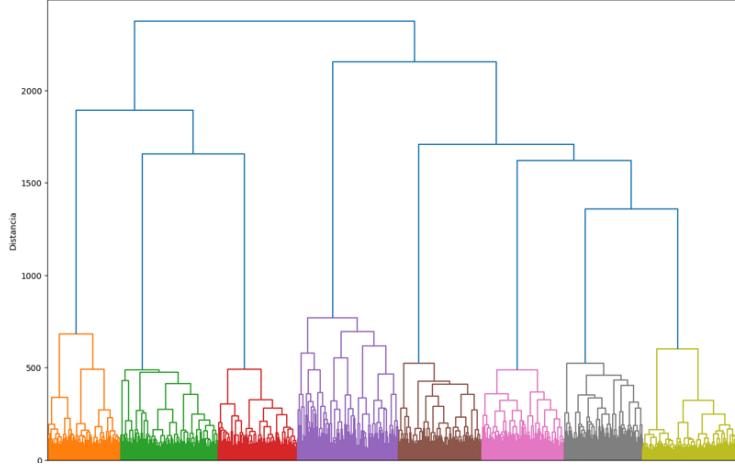


Figura 7: Dendrograma obtenido para *clustering* jerárquico de los 4096 *features* obtenidos desde VGG-16. Distancia: “euclídea”, linkage: “Ward”.

Cuadro 1: Índices de validación externa

	K-means	K-medoids d = euclídea	K-medoids d = Manhattan	C. Jerárquico
ARI	0.98	0.80	0.87	0.97
van Dongen	0.0097	0.2328	0.1261	0.0130

Un último ejercicio, dado el éxito del *clustering* realizado sobre las *features* obtenidas de la red convolucional, fue reducir esas 4096 variables sin perder poder de clasificación. Por PCA, se necesitan alrededor de 1000 atributos para lograr una explicación de la varianza por sobre el 90 % (Figura 8), y más de 200 para el 80 %, aunque al aplicar las primeras 7 componentes principales (aproximadamente el 40 % de la varianza), se obtuvo ya la segmentación adecuada de las imágenes con un Índice de Rand Ajustado de 0.9730, muy comparable al obtenido utilizando todos los *features* y totalmente superador a lo que se obtendría con un proceso simple de reducción de dimensiones y clustering únicamente (Figura 9). Esto marca la efectividad de la red VGG-16 en la extracción de información desde imágenes aún sin actualizar ninguna de sus capas previamente entrenadas.

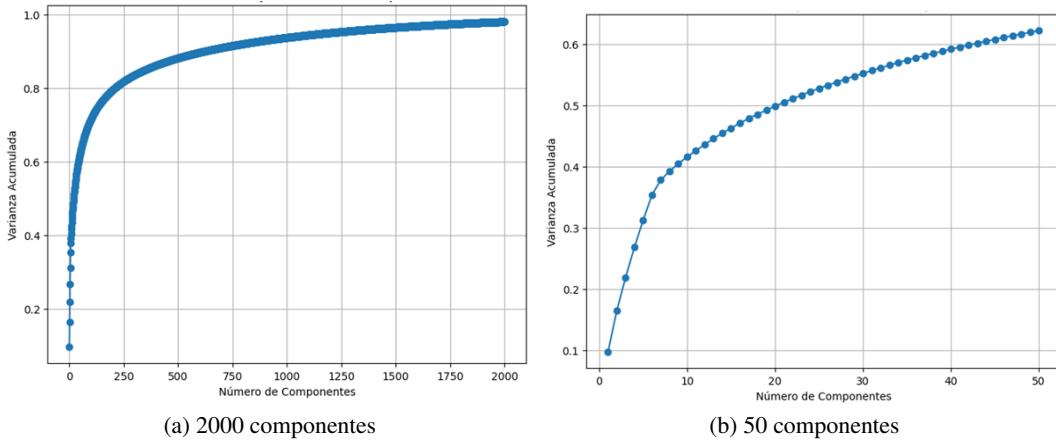


Figura 8: Varianza acumulada explicada por PCA según el número de componentes principales para una reducción de dimensionalidad de los 4096 *features* obtenidos de VGG-16.

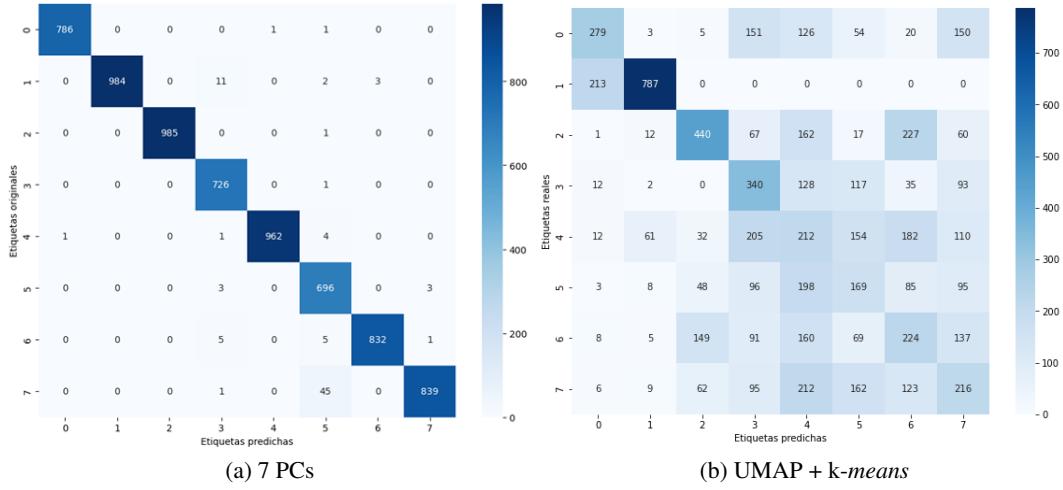
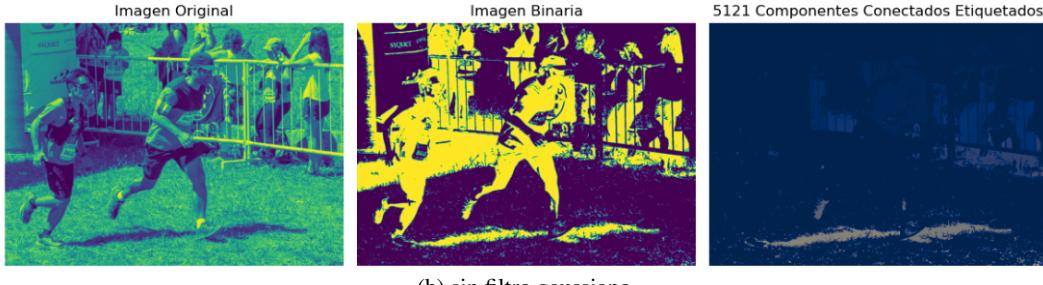


Figura 9: Matriz de confusión para *clustering* por *k-means* con las 7 PCs obtenidas desde PCA (a partir de los 4096 *features* obtenidos por VGG-16) (9a) vs. matriz de confusión tras haber aplicado UMAP + *k-means* directamente sobre las imágenes vectorizadas (sin extracción de *features* con VGG-16) (9b).



(a) con filtro gaussiano



(b) sin filtro gaussiano

Figura 10: Resultados de la imagen binaria con (10a) y sin filtro gaussiano (10b).

Detección de objetos

Connected-component labelling

Al momento de binarizar y detectar los componentes de la imagen, se encontraron diferencias (Figura 10). En el caso de la imagen binaria con *smooth gaussian*, y un *threshold* de 100 (Figura 10a) observamos mejor detección de los componentes, están más delimitados. Se observa que las sombras de las personas corriendo las pinta del mismo color. A la persona de la izquierda la distingue íntegra, en caso de la persona derecha, se pierde los límites con el fondo.

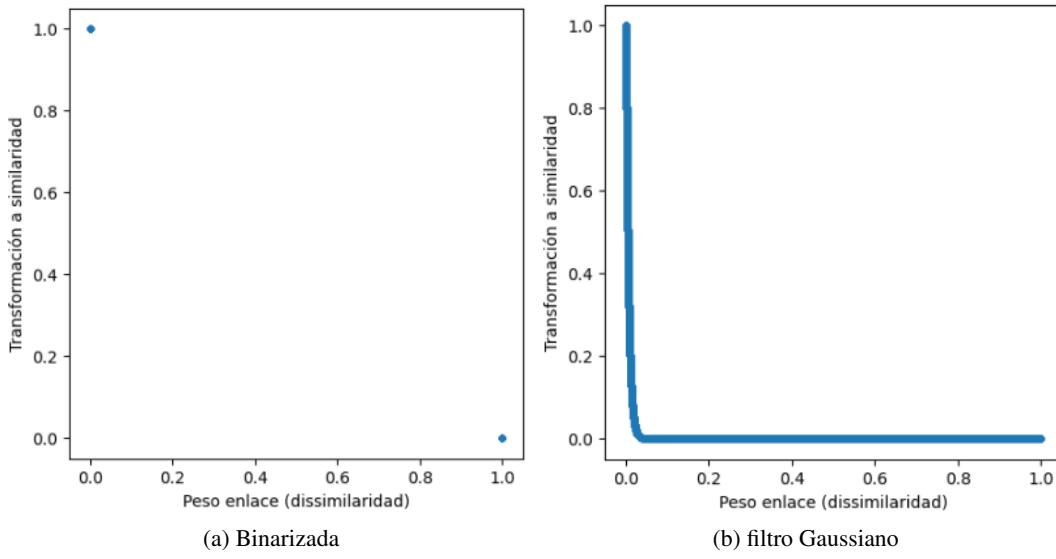


Figura 11: Disimilaridad/similaridad para la imagen binarizada (11a) y con filtro Gaussiano (11).

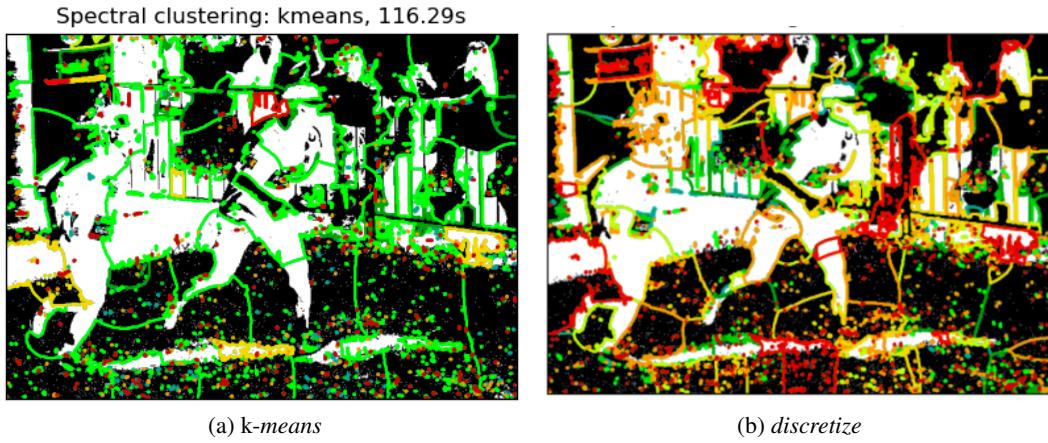


Figura 12: Clustering espectral a partir de la imagen binarizada, empleando los métodos *k-means* (12a) y *discretize* (12b).

Cuando se binariza la imagen sin el filtro gaussiano (Figura 10b) y un *threshold* de 90, el resultado que se observó es menos exitoso, a diferencia del primero. Cabe destacar que se encuentran más componentes conectados y no se llega a entender la imagen.

Spectral clustering

En el gráfico de similaridad/disimilaridad (Figura 11) se observaron diferencias con respecto a los grafos correspondientes en escalas de grises. Esto se debe a que en el caso de una imagen binaria, donde los píxeles solo tienen dos valores (por ejemplo, 0 y 1), la disimilitud entre píxeles también tendrá solo dos valores posibles. La disimilitud depende de la diferencia de intensidad entre píxeles adyacentes, y en una imagen binaria, las intensidades son discretas y limitadas.

A modo de conclusión, se afirma que en donde hubo peor desempeño es con la imagen binarizada (Figura 12). Se observó gran cantidad de puntos asociadas al ruido en la imagen. El ruido en la imagen binarizada puede ser causado por componentes desconectados, que son pequeños grupos de píxeles o píxeles individuales que no forman parte de los objetos principales que se desea segmentar. Estos componentes desconectados pueden ser interpretados por el algoritmo de *clustering* espectral

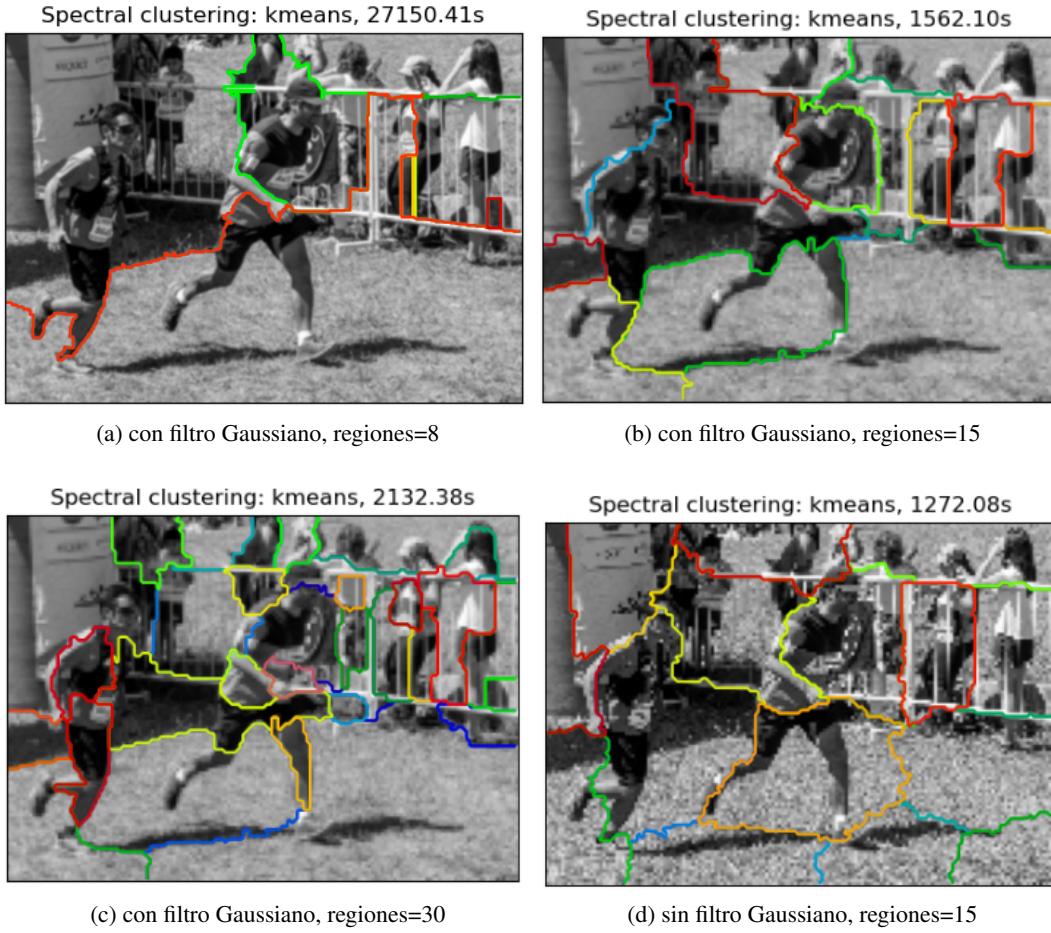


Figura 13: *Clustering* espectral con filtro Gaussiano y método *k-means*.

como objetos separados, lo que lleva a una detección excesiva de componentes conectados. Esto sucede tanto con el método *k-means*, como en *discretize*.

En el caso de las imágenes en escala de grises (Figura 13), se observó que no hay presencia de ruido identificada por puntos, si bien en algunas de las pruebas que se realizaron se pudo lograr la detección de alguna de las personas sin que las particione, puede detectar una pierna. En el caso de la cantidad de regiones=15 y con filtro gaussiano (Figura 13b) detecta la zona inferior de la persona a la izquierda.

Al realizar *clustering* espectral en la imagen sin filtro gaussiano (Figura 13d), no mejora en relación a la que sí tiene este filtro. Se puede afirmar que la mejora de resultados no está asociada a la cantidad de regiones, a pesar de que se observe n cantidad de componentes en la misma.

Red neuronal

Dado el alcance de la red VGG-16 que fue entrenada para 1000 *features* en un *dataset* de una mayor magnitud, entendimos que un enfoque ingenioso para implementar la VGG16 a nuestro *dataset* sería utilizar un red neuronal sobre los *features* generados por VGG-16.

A continuación (Figura 14), se presentan los resultados de la experimentación donde se entrena los tres *pipelines* sobre los *datasets* aumentados con imágenes distorsionadas. Se observa que los tres *pipelines* mantienen un *accuracy* cercano a 1 tanto en *train* como en *test* sobre el *dataset* sin distorsiones. El *accuracy* logrado en *test* y *train* resultan comparables en cada uno de los *pipelines*.

Resulta notable la persistencia del *pipeline* con la red neuronal manteniendo un *accuracy* elevado en *test* y *train* por sobre los *pipelines* no supervisados.

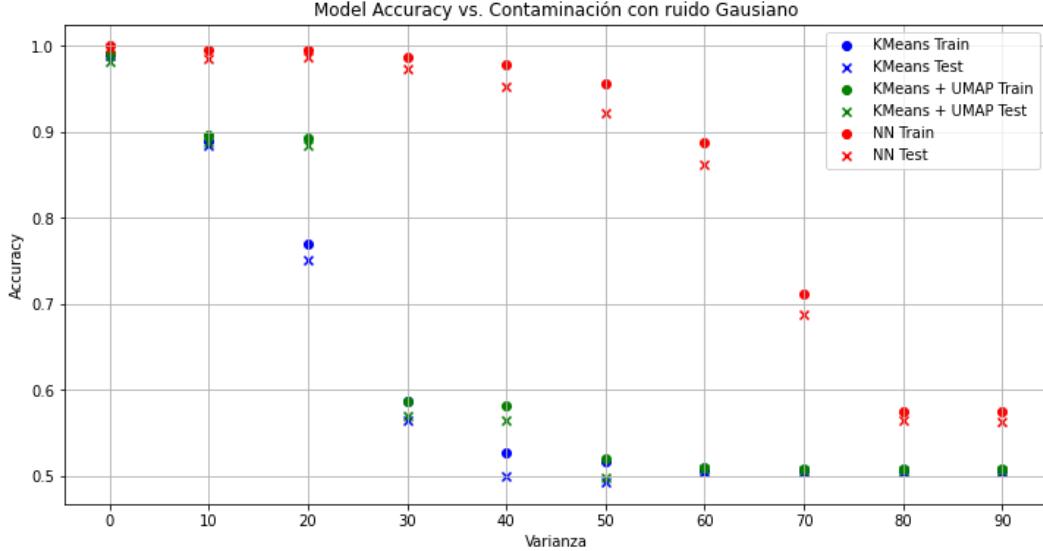


Figura 14: Resultados de *accuracy* en *train* y *test* de tres *pipelines*, para los *datasets* aumentados con ruido gaussiano creciente.

4. Conclusiones

De la aplicación de *transfer learning* con el modelo de redes neuronales convolucionales VGG-16 sobre un *set* de 6899 imágenes de objetos (aviones, autos, motos, gatos, perros, flores, frutas y personas) se encontraron resultados muy satisfactorios en la identificación de cada una de ellas. Después de la extracción de los 4096 *features* se reemplazó la capa *softmax* por diferentes algoritmos de *clustering* para la clasificación. Se utilizó *k-means*, *clustering* jerárquico y *k-medoids* para tal fin, siendo los dos primeros los de mejor desempeño. Si bien *a priori*, el *set* de imágenes no presenta una estructura de capas, el *clustering* jerárquico aportó resultados significativamente mejores utilizando el enlace Ward por sobre los otros (*single* y promedio), dando la idea de *clusters* más compactos y homogéneos, y que al margen de no distar uno del otro, la similaridad a nivel interno de cada *cluster* pesó más. Estas condiciones son las que lo hace propicio para aplicar *k-means*, con *clusters* esféricos, regulares, sin o con muy pocos *outliers*, con la mayoría de los datos alrededor de un centro, pero no alrededor de uno de los datos en particular, quizás por esto último la merma en la predicción por parte de *k-medoids*. Esta estructura a modo de casquete por parte de cada *cluster* sin ningún nivel de importancia de uno por sobre otro hace que la robustez de *k-medoids* no sea necesaria y hasta prediga un poco peor por su condición de considerar uno de los datos como referencia de la toma de distancias.

Si bien la diferenciación de estos objetos presentes en las imágenes pueda parecer sencillo, lo cierto es que habla de la bondad del modelo en capturar las características que dan unicidad a los objetos, incluso disminuyendo la dimensionalidad de manera drástica con PCA, de 4096 *features* a 7 componentes principales (aproximadamente un 40 % de la varianza explicada), se mantuvieron los aciertos en la identificación de las imágenes con *k-means*.

En este trabajo se han explorado diversas estrategias en la creación de *pipelines* para la clasificación de un *set* de imágenes cotidianas. Relativo a los algoritmos no supervisados las estrategias con *spectral clustering* y *connected-component labelling*, a pesar de las distintas combinaciones de hiperparámetros realizadas sobre las imágenes, el resultado obtenido fue poco satisfactorio en comparación al método *k-means*. Se encontró que un primer enfoque vectorizando imágenes y aplicándose una reducción de dimensionalidad con UMAP dio resultados modestos clasificando correctamente una sola categoría. Creemos que es un enfoque válido para continuar en futuros trabajos combinado con la creación de nuevas *features*, tales como imágenes binarizadas.

Relativo a la exploración de enfoques supervisados, se entrenaron los *features* generados por la red VGG-16, por una red neuronal con una activación final *softmax* de 8 neuronas. Este enfoque tuvo una *performance* con mayor *accuracy*, con respecto a los *pipelines* con un enfoque no supervisado.

La base de datos utilizada se encuentra adecuadamente procesada y clasificada; sus elementos se encuentran centrados, con adecuada exposición y cada elemento se encuentra representado por una única categoría. Entendiendo la necesidad general de que los modelos de clasificación funcionen adecuada o previsiblemente incluso ante *inputs* denigradas, se evaluó la robustez de los *pipelines* desarrollados. En este trabajo, se presentó un enfoque novedoso de *data-augmentation* incorporando un *dataset* equivalente con imágenes distorsionadas con ruido gaussiano. Los modelos no supervisados sufrieron un impacto considerable en *accuracy* en este nuevo *dataset*, en cambio el modelo supervisado desarrollado mantiene una robustez sustancial.

En conclusión, se describe una técnica de *data-augmentation* de utilidad en la evaluación y entrenamiento de modelos de clasificación. Se halla que un enfoque supervisado de *transfer learning* partiendo de la red VGG-16 y finalizando con una nueva red neuronal dio mejores resultados de clasificación antes que otros enfoques no supervisados.

Referencias

- [1] Jia Deng et al. «Imagenet: A large-scale hierarchical image database». En: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, págs. 248-255.
- [2] Samuel Dodge y Lina Karam. «Understanding how image quality affects deep neural networks». En: *2016 eighth international conference on quality of multimedia experience (QoMEX)*. IEEE. 2016, págs. 1-6.
- [3] Anil K Jain, M Narasimha Murty y Patrick J Flynn. «Data clustering: a review». En: *ACM computing surveys (CSUR)* 31.3 (1999), págs. 264-323.
- [4] Prasun Roy et al. «Effects of degradations on deep neural network architectures». En: *arXiv preprint arXiv:1807.10108* (2018).
- [5] Karen Simonyan y Andrew Zisserman. «Very deep convolutional networks for large-scale image recognition». En: *arXiv preprint arXiv:1409.1556* (2014).
- [6] Christian Szegedy et al. «Intriguing properties of neural networks». En: *arXiv preprint arXiv:1312.6199* (2013).
- [7] Pang-Ning Tan, Michael Steinbach y Vipin Kumar. *Introduction to Data Mining*. Pearson, 2021.
- [8] Dongkuan Xu y Yingjie Tian. «A comprehensive survey of clustering algorithms». En: *Annals of data science* 2 (2015), págs. 165-193.