MDPI

Article

Algoritmos de clusterización y detección de objetos en imágenes naturales

Florencia Boemo, José Saint Germain y Leopoldo Serpa

Abstract: En el presente trabajo, se estudió la capacidad de la red neuronal convolucional VGG16 para extraer atributos relevantes de imágenes naturales. Las features extraidas se utilizaron en tareas de clasificación, mediante el uso de los algoritmos *k-means*, *k-medoids* y DBSCAN, y también para la identificación de objetos, a través de clustering espectral y *Connected component labeling*. No se detectaron marcadas diferencias en la calidad de la clasificación con *k-means* y *k-medoids*. La clasificación con DBSCAN no arrojó en general resultados satisfactorios, atribuyéndose esta mala performance a la alta dimensionalidad del dataset. Por otra parte, se buscó identificar objetos en imágenes, a partir de los algoritmos *connected component labelling* y clustering espectral. Se obtuvieron resultados exitosos en ambos casos para la imagen de muestra. En particular, el primero tuvo una buena performance separando el objeto del fondo, mientras que con clustering espectral pudieron aislarse partes del objeto.

Keywords: *k-means*; *k-medoids*; *DBSCAN*; *connected component labelling*; *spectral clustering*

1. Introducción

La clasificación de imágenes es una de las bases del campo de la visión artificial (*computer vision*), ya que sienta los fundamentos para tareas más complejas como localización, detección y segmentación. Antes del advenimiento de las redes neuronales, los algoritmos tradicionales de extracción de atributos se enfocaban en identificar características específicas de las imágenes. Por ello, estos métodos tenían una capacidad de generalización pobre [1]. En años recientes, los modelos de *deep learning*, que utilizan múltiples capas de procesamiento no lineal de la información, mostraron resultados superadores. En particular, las redes neuronales convolucionales se han convertido en la arquitectura preferida en cuestiones de reconocimiento, clasificación y detección de imágenes.

Las redes neuronales convolucionales son redes *feedforward*, en las que el flujo de información se da en una única dirección. Su arquitectura puede ser variable, pero en general consisten de capas convolucionales, de *pooling*, y capas *fully-connected*, agrupadas en módulos. Los módulos se apilan para generar una red profunda. De manera simple, las capas convolucionales extraen los atributos de las imágenes, las capas de *pooling* reducen la dimensionalidad de los atributos, y las capas *fully-connected* interpretan los features extraidos.

Existen varios modelos considerados clásicos en redes neuronales convolucionales, tales como LeNet-5, AlexNet, VGG y NetworkInNetwork [2]. En particular, en este trabajo se explorará el funcionamiento de la red neuronal VGG16. Las potenciales aplicaciones de esta red han sido ampliamente estudiadas, principalmente en el campo de la salud (por ejemplo, detección de cáncer de mama [4], retinopatías producto de diabetes [5] y nódulos en pulmones [6]) y de la biología (por ej., identificación de plagas en berenjenas [7] y hojas de tabaco [8] y reconocimiento de especies de peces [9]). En el presente trabajo, se buscará extraer atributos relevantes de un dataset de imágenes naturales, para luego utilizar dichos atributos en tareas de clasificación e identificación. Se plantea que, si bien las imágenes presentan características diferentes (esto es, no pertenecen a una misma clase de objeto ni

Citation: Title. *Journal Not Specified* , , .

Received:
Accepted:
Published:

están capturadas con composiciones similares), la red VGG16 es capaz de extraer features que permitan superar estas barreras y arribar a clasificación e identificación óptimas.

2. Materiales y métodos

En el presente trabajo, se buscará extraer atributos relevantes de un conjunto de imágenes. Para ello, se analizará un dataset de imágenes naturales, conteniendo 6899 fotografías de distintas clases de objetos [3]. En particular, el conjunto de datos incluye imágenes de aviones, autos, gatos, flores, perros, frutas, motos y personas. El dataset es de acceso público, y puede descargarse de internet.

Para el análisis de atributos, se utilizará la red neuronal convolucional VGG16. Su arquitectura permite el reconocimiento de imágenes, a través de la detección y extracción de sus atributos. Esta red es de fácil implementación, si bien su entrenamiento demora mucho y requiere gran poder de cómputo. Asimismo, su configuración permite distinguir un gran nivel de detalle en las imágenes. La implementación de VGG16 en Keras provee un modelo pre-entrenado, con pesos ya calculados. Por defecto, VGG16 trabaja con imágenes RGB de 224x224 pixels, por lo cual es necesario realizar un pre-procesamiento sobre el dataset para adaptar el tamaño de las imágenes.

Los atributos relevantes extraídos de las imágenes serán utilizados para alimentar algoritmos no-supervisados para la identificación de clusters. En particular, se trabajará con *k-means*, *k-medoids* y DBSCAN. Los clusters identificados serán evaluados y validados con diferentes métricas, dado que se cuenta con las categorías de las imágenes. Se buscará además, identificar objetos en una imagen, aplicando los algoritmos de *Connected component labeling* y clustering espectral.

3. Resultados y discusión

3.1. Análisis preliminar de imágenes

A fin de familiarizarse con el dataset, se calculó la imagen promedio para cada clase (Figura 1). A simple vista, se puede notar que las clases más homogéneas presentan una imagen promedio más nítida. Tal es el caso de los aviones, las frutas y las motos. Esto podría tener implicancias en futuros procesos de clusterización.

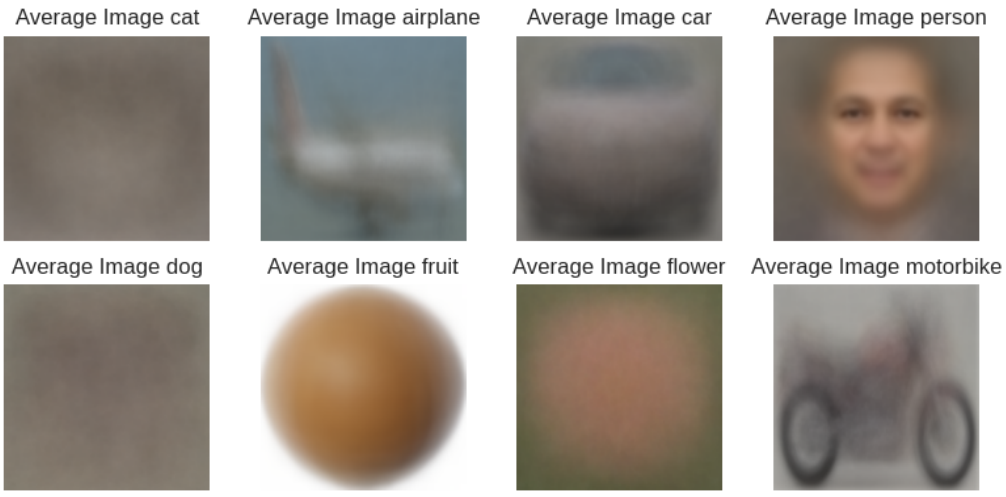


Figura 1. Promedio de imágenes para cada categoría del dataset.

Además, se estudió la distribución de brillo (*brightness*) en cada canal de color, para todos los grupos de fotografías (Figura 2). Se puede apreciar que no hay sub- o sobreexposición, a excepción de la categoría de frutas. Esto es, las distribuciones no están corridas a valores de brillo extremos. Por lo tanto, podría aventurarse que las imágenes son de buena calidad para la extracción de atributos, ya que no se pierden detalles por efecto de la luz. Para confirmar esta suposición, podría ser relevante comparar la performance de

los algoritmos en la categoría frutas frente a las restantes, ya que histogramas con colas pesadas no necesariamente implican pérdida de detalle en las fotografías.

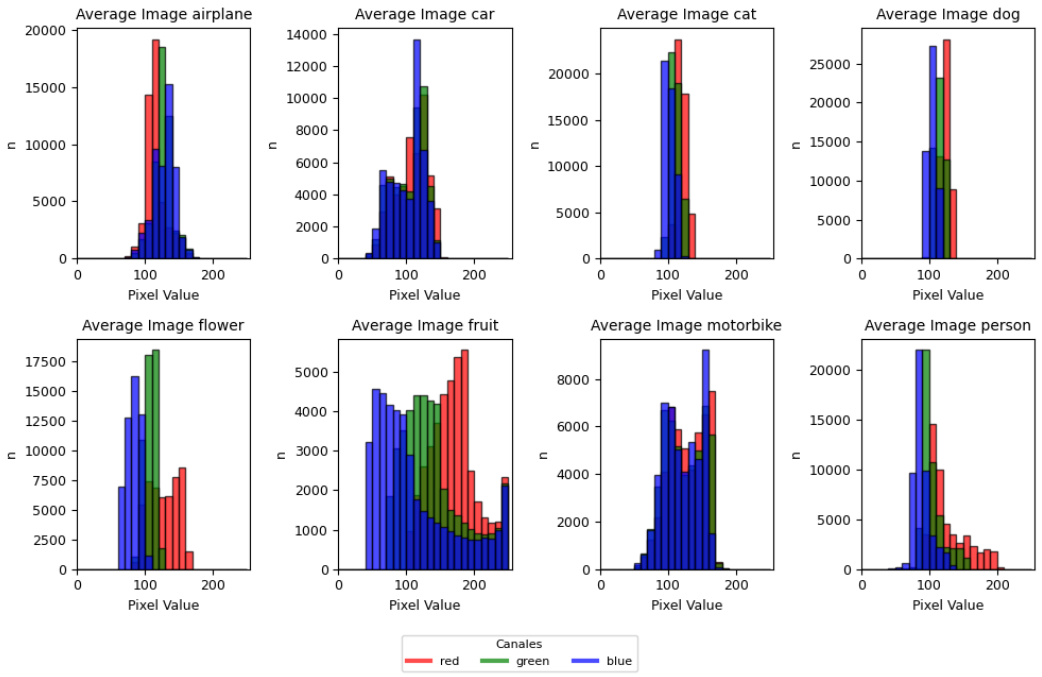


Figura 2. Histogramas RGB para cada categoría del dataset.

Finalmente, se procesó el dataset utilizando la red VGG16, obteniéndose como resultado 4096 features. Estos features son los principales atributos que pudo reconocer la red, y que implican una reducción de casi el 98 % de las dimensiones del problema.

3.2. Clusterización de imágenes con k-means

Se aplicó el algoritmo k-means a las imágenes, previo procesamiento con la red VGG16. Se realizaron pruebas con dos, tres, cuatro, cinco, seis, siete y ocho clusters, considerando los hiperparámetros listados en el Cuadro **??**. El número óptimo de grupos se definió en base al coeficiente de silhouette (Cuadro 1) y utilizando el método del codo (Figura 3). A priori, teniendo en cuenta las imágenes disponibles, la clasificación debería ser óptima con ocho clusters.

Cuadro 1. Coeficiente de silhouette para cada cluster - *k-means*.

Número de clusters	Coeficiente de silhouette
2	0.080
3	0.102
4	0.113
5	0.086
6	0.144
7	0.128
8	0.143

Observando los valores obtenidos en el Cuadro 1, es posible apreciar que, para seis y ocho clusters, los coeficientes de silhouette son similares. En la Figura 3, no es posible distinguir un cambio de pendiente marcado para k = 6, mientras que se detecta un leve codo con k = 8. Sin embargo, en la Figura 4b se pueden apreciar valores de silhouette negativos para el cluster número 4. Este fenómeno, si bien está presente, es menos marcado cuando se consideran 6 clusters (Figura 4a). Es interesante destacar que, con ocho clusters,

se obtienen grupos de tamaño similar, mientras que para seis clusters existe uno de ellos con una cantidad de elementos mucho mayor.

La eficiencia de la clasificación se puede apreciar observando las matrices de confusión para los k considerados (Figura 5), así como los índices de Van Dongen y de Rand ajustado (Cuadro 2).

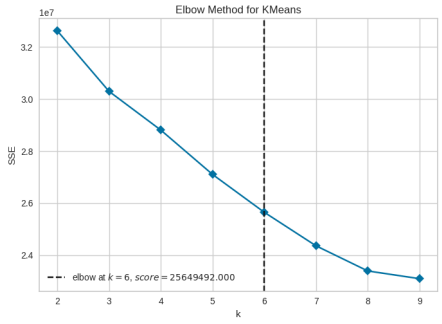
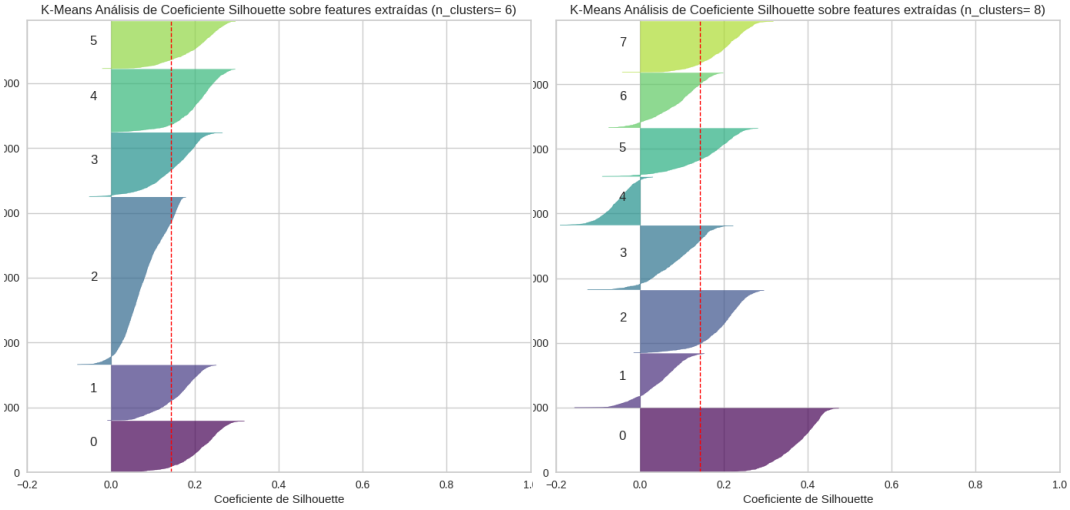
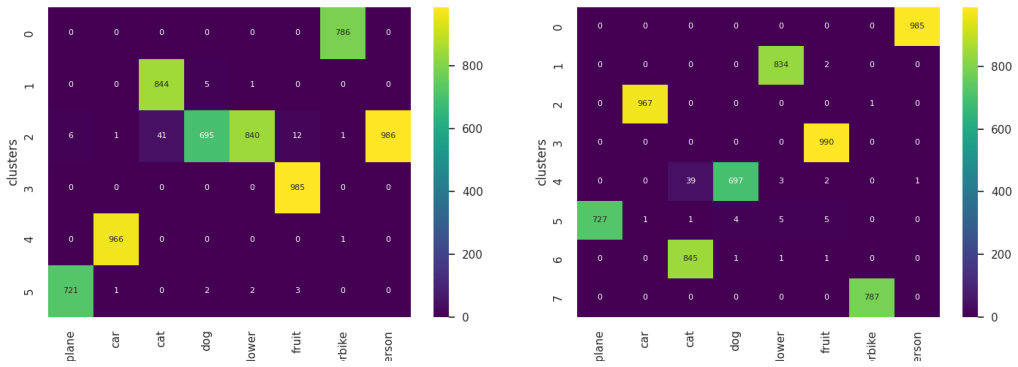


Figura 3. Optimización del número de clusters en *k-means* - Método del codo



(a) k = 6 **(b)** k = 8

Figura 4. Coeficientes de silhouette



(a) k = 6 **(b)** k = 8

Figura 5. Matrices de confusión

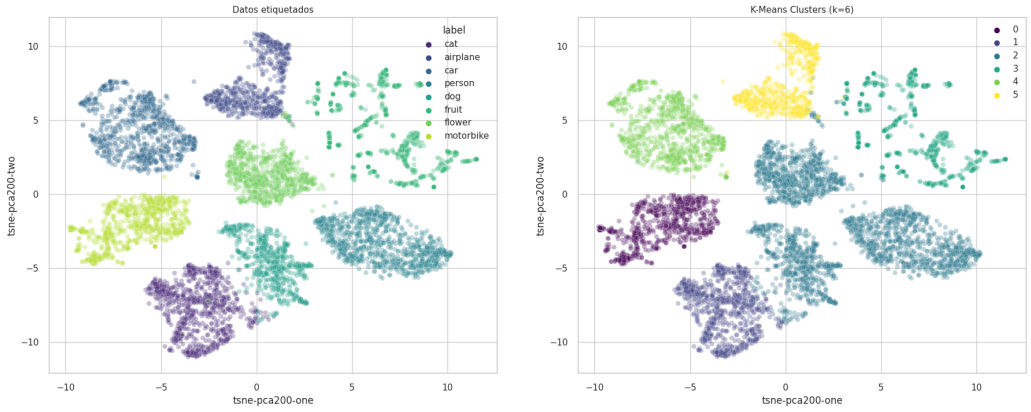
Cuadro 2. Métricas de bondad de clasificación - *k-means*

Número de clusters	Índice de Rand ajustado	Índice de Van Dongen
6	0.165	0.663
8	0.011	0.980

Considerando las métricas resumidas en el Cuadro 2, se puede afirmar que, contrariamente a lo esperado, la clasificación en seis clusters es mejor que en ocho. Los clusters generados con k = 6 son más homogéneos internamente, lo que se refleja en un mayor índice de Rand ajustado y un menor índice de Van Dongen. Si bien la matriz de confusión para k = 8 (Figura 5b) pareciera indicar menos "mezcla"de las clases en los clusters, se puede apreciar en la Figura 4b que el coeficiente de silhouette promedio para dicho k (ver Cuadro 1) en realidad está traccionado por el cluster 0, que tiene una performance superior comparada con las otras.

Es interesante destacar que, para k = 6, de todas las categorías de imágenes, la que presentó mayor dispersión entre los clusters fue la de gatos (aprox. 4 % del total de imágenes en otros clusters). Asimismo, las imágenes de personas fueron todas asignadas al mismo cluster. Retomando las observaciones del análisis preliminar (Apartado 3.1), podría pensarse que una imagen promedio más borrosa en la Figura 1 estaría correlacionada con mayor dificultad en la clasificación de dicha categoría. Por otra parte, la clasificación de imágenes de frutas resultó óptima para k = 6 y k = 8, con lo que puede afirmarse que la alta exposición de las fotos no redundó en pérdida de calidad que afectara al clasificador.

A continuación, se visualizó la clasificación en bajas dimensiones utilizando T-SNE. Previo a este análisis, se realizó una reducción de dimensionalidad de los atributos utilizando PCA. Se retuvieron las primeras 200 componentes principales, que en total explican cerca del 80 % de la varianza total. Se muestran las etiquetas reales de los datos (Figura 6a), versus la clasificación realizada con *k-means* (Figura 6b). Se pudo verificar que el cluster 2 está compuesto en realidad de tres pequeñas aglomeraciones separadas. Esto está de acuerdo con lo observado en la matriz de confusión (Figura 5a), ya que este cluster en particular alberga mayoritariamente imágenes de perros, flores y personas.



(a) Datos etiquetados **(b)** Clasificación con *k-means* (k = 6)

Figura 6. Representación bidimensional (T-SNE) de resultados

3.3. Clusterización de imágenes con k-medoids

El análisis realizado anteriormente se repitió utilizando *k-medoids o Partition Around Medoids* como método de clasificación. La diferencia entre estos algoritmos radica en la métrica de distancia considerada: para *k-means*, se utiliza la distancia euclidea, mientras que *k-medoids* hace uso de la distancia Manhattan. Teniendo en cuenta que, en general, en situaciones de alta dimensionalidad las distancias de Manhattan funcionan mejor, podría esperarse que la clasificación mejore respecto de la observada en el apartado 3.2.

Nuevamente, se probaron distintas cantidades de clusters, y se analizó el número óptimo de los mismos en base al coeficiente de silhouette (Cuadro 3) y al método del codo (Figura 7).

Cuadro 3. Coeficiente de silhouette para cada cluster - *k-medoids*.

Número de clusters	Coeficiente de silhouette
2	0.062
3	0.064
4	0.113
5	0.136
6	0.129
7	0.114
8	0.143

En un principio, puede observarse que el coeficiente de silhouette para k = 8 no difiere significativamente del obtenido en el análisis de *k-means*. Sin embargo, es relevante analizar la información como un todo, ya que observando la Figura 7, se aprecia un codo para k = 5, aunque el coeficiente de silhouette para dicha partición es bastante más bajo (Cuadro 3).

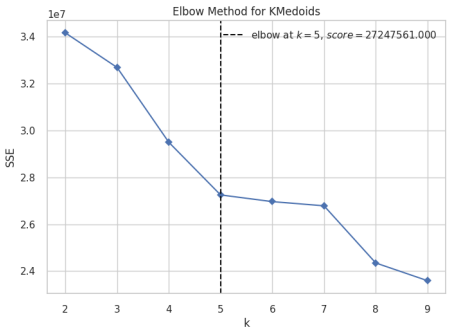
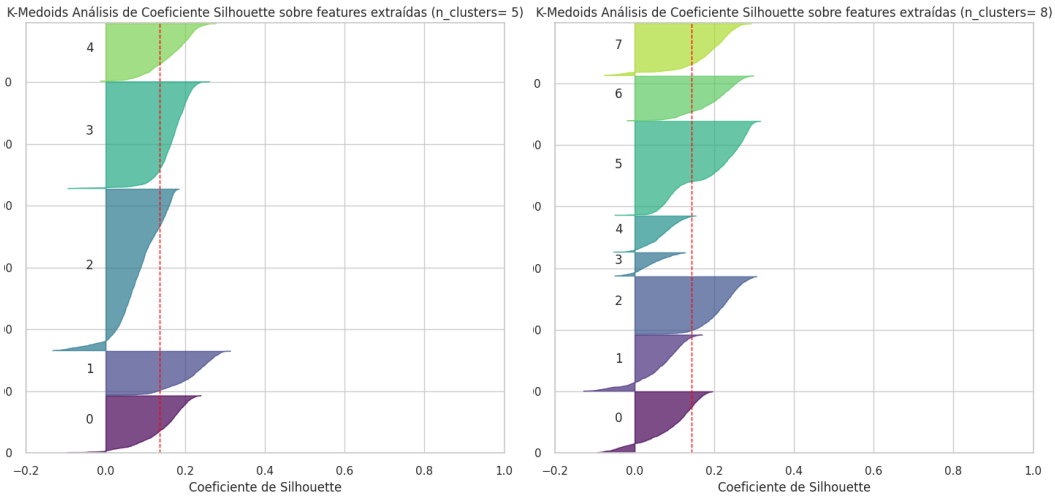


Figura 7. Optimización del número de clusters en *k-medoids* - Método del codo

Los coeficientes de silhouette para cada cluster, con k = 5 y k = 8, se muestran en la Figura 8. Se detectan coeficientes de silhouette negativos en ambas particiones, sin ser este fenómeno más marcado en ninguna de ellas. Se aprecia para k = 5 la presencia de un cluster (número 2) con mayor número de elementos. Por el contrario, cuando k = 8 existen agrupamientos con pocos elementos (clusters 3 y 4), lo que sugeriría un posible *overfitting* de los datos.



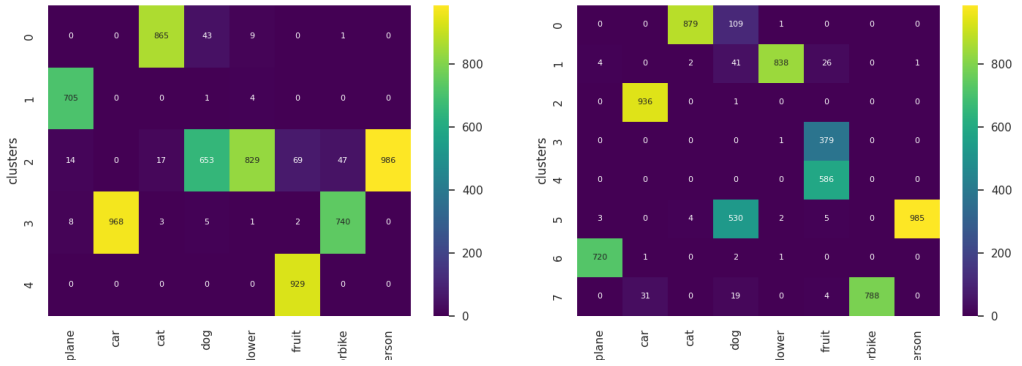
(a) k = 5 **(b)** k = 8

Figura 8. Coeficientes de silhouette

La calidad del clustering se verificó observando los índices de Rand ajustado y de Van Dongen (Cuadro 4), así como las matrices de confusión para ambos k (Figura 9).

Cuadro 4. Métricas de bondad de clasificación - *k-medoids*

Número de clusters	Índice de Rand ajustado	Índice de Van Dongen
5	0.231	0.554
8	0.011	0.792

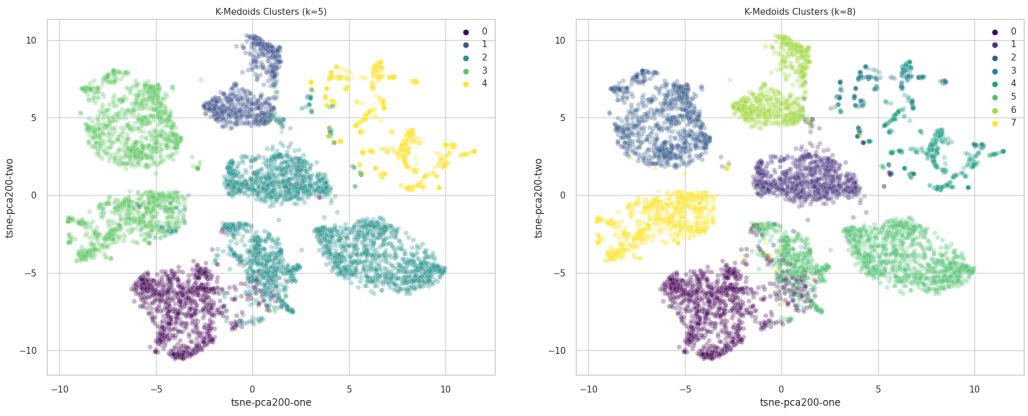


(a) k = 5 **(b)** k = 8

Figura 9. Matrices de confusión

De acuerdo a la matriz de confusión para k = 5 (Figura 9a), el cluster número 2 alberga imágenes de categorías distintas. Más aún, es el cluster que concentra la proporción mayoritaria de fotografías de perros, flores y personas. Es un fenómeno similar al observado en *k-means* para k = 6 (Figura 5a). Tampoco existe buena separación en las imágenes de autos y motocicletas. Por el contrario, al considerar la partición con k = 8, se detecta que tanto las imágenes de frutas como las de perros están divididas en dos clusters, en una fracción no despreciable. Además, el algoritmo no separa bien las fotografías de perros y de personas. Por lo tanto, si bien el coeficiente de silhouette en este caso es mayor que para k = 5 (Cuadro 8), es válido cuestionarse si la clasificación es necesariamente mejor. Este razonamiento se ve apoyado por las métricas de bondad de clasificación resumidas en el Cuadro 4.

Se visualizó también el clustering en bajas dimensiones, utilizando T-SNE (figura 10), previa reducción de la dimensionalidad. Nuevamente, para k = 5 (Figura 10a), se observan tres aglomeraciones diferentes para el cluster 2, que se corresponde con la composición heterogénea de esa clase (perros, flores y personas principalmente, ver Figura 6a), y dos aglomeraciones para el cluster 3. Puede afirmarse entonces que, con k = 5, el algoritmo no posee buen poder de discriminación. Por otra parte, para k = 8 (Figura 10b) se aprecia como particularmente problemático el agrupamiento de perros y personas en un mismo cluster, la frontera no definida entre el cluster 0 y el 5 (pues ambos contienen imágenes de perros) y la clasificación de frutas en dos grupos separados (clusters 3 y 4, superpuestos). Por lo tanto, no es posible concluir cuál de las dos particiones resulta tener mejor performance para separar los datos, ya que ambas muestran una performance pobre en distintas categorías.



(a) k = 5 **(b)** k = 8

Figura 10. Visualización en bajas dimensiones de *k-medoids*

3.4. Clusterización de imágenes con DBSCAN

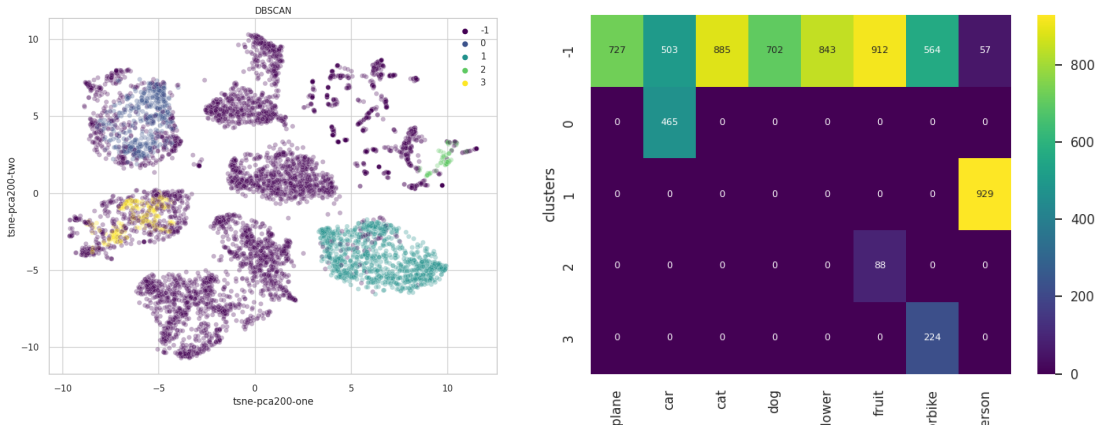
Una última clasificación de imágenes se realizó utilizando el algoritmo DBSCAN, que se basa en la idea de que, para cada punto de un cluster, debe existir una vecindad con un número mínimo de puntos. De lo contrario, se considera un punto de ruido. La diferencia con los algoritmos explorados anteriormente es que no requiere que el usuario defina el número de clusters a priori. En este experimento, se exploraron dos hiperparámetros: el *eps* que corresponde a la máxima distancia entre dos puntos para que sean considerados vecinos, y *min_samples* que es el el número mínimo de puntos para formar una región densa.

Las medidas de bondad de clasificación para los cinco mejores experimentos, así como la cantidad de clusters encontrados en cada uno, se resumen en el Cuadro 5.

Cuadro 5. Métricas de bondad de clasificación - DBSCAN

Experimento	Número de clusters	Coeficiente de silhouette	Índice de Rand ajustado	Índice de Van Dongen
1	4	-0.049	0.110	0.673
2	5	-0.054	0.00	0.992
3	5	-0.062	0.030	0.855
4	6	-0.064	0.032	0.843
5	8	-0.072	0.130	0.632

En general, observando el coeficiente de silhouette, se puede apreciar que la clasificación no es buena. Los valores negativos y cercanos a cero indican superposición de los clusters, así como asignaciones a grupos erróneas. Una imagen representativa de los resultados obtenidos se muestra en la Figura 11. Como puede apreciarse, la mayoría de los elementos pertenece al cluster -1. Esto es, no han podido ser clasificados por el algoritmo. Esto es un resultado esperable, ya que DBSCAN es un algoritmo cuya performance es pobre con alta dimensionalidad.



(a) Visualización en bajas dimensiones **(b)** Matriz de confusión

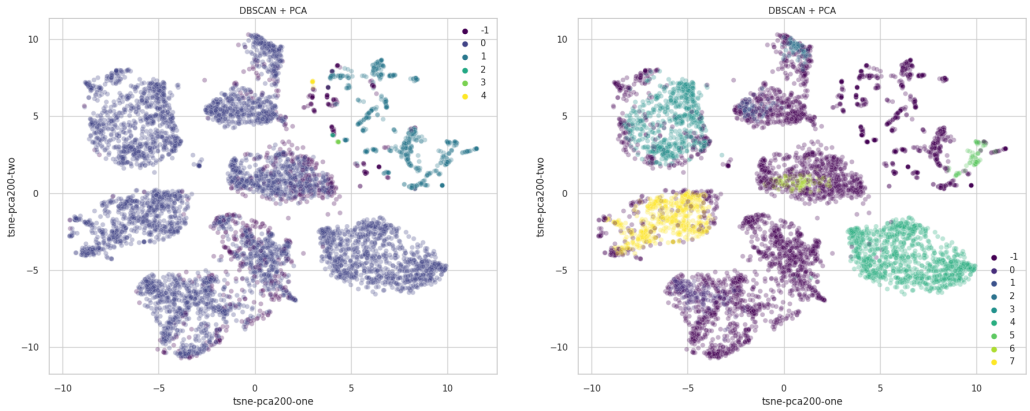
Figura 11. Resultados de Experimento 1 con DBSCAN (ver Cuadro 5)

Para comprobar si la performance mejora en dimensiones menores, se volvió a aplicar DBSCAN al dataset, previa reducción de la dimensionalidad con PCA. Se conservaron las primeras 200 componentes, que explican un 80 % de la varianza. Los resultados de los cinco mejores experimentos se muestran en el Cuadro 6.

De acuerdo a los resultados, la técnica mejora levemente al considerar menos dimensiones. Para los experimentos 1 y 2, que podrían considerarse los más exitosos, se obtienen valores de clusters similares a los evaluados en los experimentos de *k-means* y *k-medoids*. Los resultados se pueden visualizar en la Figura 12.

Cuadro 6. Métricas de bondad de clasificación - DBSCAN + PCA

Experimento	Número de clusters	Coeficiente de silhouette	Índice de Rand ajustado	Índice de Van Dongen
1	6	0.107	0.084	0.759
2	8	-0.031	0.206	0.515
3	5	-0.061	0.000	0.992
4	6	-0.105	0.055	0.780
5	3	-0.106	0.000	0.999



(a) Experimento 1 **(b)** Experimento 2

Figura 12. Visualización en bajas dimensiones de DBSCAN + PCA (ver Cuadro 6)

De todas maneras, si bien la clusterización es levemente mejor que para el experimento con DBSCAN (Figura 11a), los resultados siguen siendo desalentadores, considerando la cantidad de puntos que no pudieron ser asignados a ninguna aglomeración.

3.5. Detección de objetos dentro de una imagen

3.5.1. *Connected-component labelling*

El algoritmo *connected-component labelling* permite identificar aquellos pixels que están conectados entre sí. De esa forma, tiene la posibilidad de detectar patrones que correspondan a objetos dentro de una imagen. Para su utilización, el algoritmo requiere previamente la binarización de la imagen; es decir, transformar cada uno de sus píxeles en blanco o negro, dependiendo de si su valor se encuentra por encima o por debajo de un umbral establecido. En el presente trabajo, para la imagen elegida, se invirtió la binarización (es decir, se asignó blanco a pixel<=threshold) y se iteró sobre distintos valores de threshold (50, 60, 65, 70, 75, 80, 85, 90, 100, 127) hasta encontrar el que visualmente presentara la mejor segmentación.

En la Figura 13 se muestra el mejor resultado encontrado, el cual corresponde a un threshold = 75. Sin embargo, para umbrales entre 60 y 80 se logra también una correcta identificación de la silueta de la moto. Mientras que con valores menores a 60, no se logran identificar los espejos y con intensidad mayores a 80 comienza a mezclarse con el fondo.

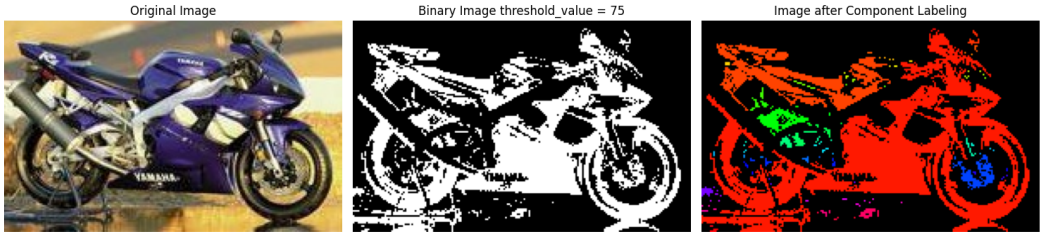


Figura 13. De izquiera a derecha: Imagen original, binarización con umbral seleccionado e imagen con los componentes reconocidos.

En definitiva, el algoritmo permitió identificar satisfactoriamente la motocicleta, diferenciándola del resto de la fotografía.

3.5.2. Clustering espectral

Para la implementación de este algoritmo se utilizaron las librerías *spectral_clustering* de *sklearn.cluster* y *image* de *sklearn.feature_extraction*. Con la primera de ellas se implementa la clusterización propiamente dicha, mientras que con la segunda se convierte la imagen a analizar en un grafo. A continuación se describe la metodología utilizada:

1. Se carga la imagen RGB (posee tres canales).
2. Se convierte a escala de grises (un solo canal)
3. Se re-escala la imagen (usando *rescale* de *skimage.transform)*. Este paso puede ser o no necesario, dependiendo de lo costoso computacionalmente que sean los procesos posteriores.
4. Se convierte la imagen en un grafo.
5. Se corre el algoritmo *spectral_clustering* sobre el grafo.

Se ejecutó varias veces la conversión de la imagen a grafo (probando distintas transformaciones) y se probaron diferentes hiperparámetros de *spectral_clustering*, particularmente se recorrieron distintos valores de *n_clusters*, *assign_labels* y *eigen_solver*.

Uno de los mejores resultados obtenidos (Figura 14) se obtuvo con *n_clusters*=15, *eigen_tol*=1e-7 y *eigen_solver*='arpack'. Con respecto al hiperparámetro *assign_labels* no se aprecian grandes diferencias, sin embargo con *discretize* la rueda delantera esta perfectamente identificada.

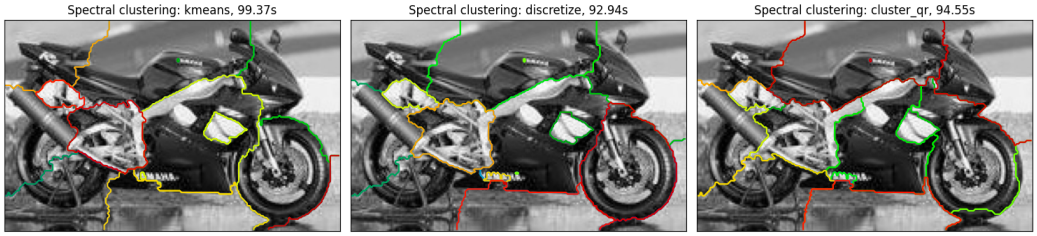


Figura 14. *Spectral_clustering* con diferentes *assign_labels*

3.5.3. Comparación *spectral_clustering* vs *connected component labelling*.

A la luz de la aplicación de ambos algoritmos en la imagen de ejemplo, podemos identificar que ambos son útiles para la detección de objetos, aunque para aplicaciones diferentes: mientras que *component-labelling* es útil para diferenciar al objeto principal (la motocicleta) del resto de la imagen, spectral clustering puede detectar subconjuntos del objeto principal (en el caso de discretize, la rueda delantera).

4. Conclusiones

Como primer idea, se destaca la potencia del algoritmo VGG16. Con distintas performance, la mayoría de los algoritmos no supervisados tomaron los atributos generados por la red neuronal y lograron diferenciarlos.

Profundizando en los tres algoritmos de clustering, DBSCAN se destacó por su muy baja performance en comparación con los otros dos. Adicionalmente, el algoritmo PAM se

destacó con respecto a Kmeans, puesto que logró diferenciar las imágenes de perros, flores y personas (agrupados en un solo cluster por Kmeans).

Finalmente, ambos algoritmos de detección de objetos resultaron set útiles: connected component labelling para la detección de un objeto con respecto a su fondo, y spectral clustering para la detección de partes específicas de un objeto. En ambos casos, se pudo constatar la importancia de un adecuado pretratamiento de la imagen para obtener resultados fiables.

References

1. Rawat, W., & Wang, Z. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, 29(9), 2352–2449. doi:10.1162/neco_a_00990
2. Chen L, Li S, Bai Q, Yang J, Jiang S, Miao Y. Review of Image Classification Algorithms Based on Convolutional Neural Networks. *Remote Sensing*. 2021; 13(22):4712. https://doi.org/10.3390/rs13224712
3. Roy, Prasun and Ghosh, Subhankar and Bhattacharya, Saumik and Pal, Umapada. Effects of Degradations on Deep Neural Network Architectures. arXiv preprint arXiv:1807.10108, 2018. Disponible en https://www.kaggle.com/datasets/prasunroy/natural-images
4. D. Albashish, R. Al-Sayyed, A. Abdullah, M. H. Ryalat and N. Ahmad Almansour, "Deep CNN Model based on VGG16 for Breast Cancer Classification,"2021 *International Conference on Information Technology (ICIT)*, Amman, Jordan, 2021, pp. 805-810, doi: 10.1109/ICIT52682.2021.9491631.
5. da Rocha, D.A., Ferreira, F.M.F. & Peixoto, Z.M.A. Diabetic retinopathy classification using VGG16 neural network. *Res. Biomed. Eng*. 38, 761–772 (2022). https://doi.org/10.1007/s42600-022-00200-8
6. Zhao D, Zhu D, Lu J, Luo Y, Zhang G. Synthetic Medical Images Using F&BGAN for Improved Lung Nodules Classification by Multi-Scale VGG16. *Symmetry*. 2018; 10(10):519. https://doi.org/10.3390/sym10100519
7. Krishnaswamy Rangarajan, A., Purushothaman, R. Disease Classification in Eggplant Using Pre-trained VGG16 and MSVM. *Sci Rep* 10, 2322 (2020). https://doi.org/10.1038/s41598-020-59108-x
8. D. I. Swasono, H. Tjandrasa and C. Fathicah, Çlassification of Tobacco Leaf Pests Using VGG16 Transfer Learning,"2019 *12th International Conference on Information Communication Technology and System (ICTS)*, Surabaya, Indonesia, 2019, pp. 176-181, doi: 10.1109/ICTS.2019.8850946.
9. Hridayami, P., et al. Fish Species Recognition Using VGG16 Deep Convolutional Neural Network. *Journal of Computing Science and Engineering*, 2019, 13(3), 124-130. http://dx.doi.org/10.5626/JCSE.2019.13.3.124