Assessing advanced computer vision for waste image classification

Juan Manuel Camara Diaz

Resumen– El reciclaje es esencial para proteger el medio ambiente y conservar los recursos naturales al transformar materiales utilizados en nuevos productos y reducir la acumulación de desechos en vertederos. La automatización del proceso de reciclaje, tanto a nivel doméstico como profesional, puede incrementar la eficiencia y exactitud en la separación de residuos. Este artículo examina avances en visión por computadora aplicados al desarrollo de herramientas para la clasificación de desechos. Se aborda la creación de instrumentos para simplificar el etiquetado de imágenes de residuos, así como el desarrollo de una aplicación móvil destinada a promover y facilitar el reciclaje en el hogar. Por último, se evalúa la factibilidad de construir un conjunto de datos de segmentación utilizando imágenes etiquetadas.

Palabras clave– Reciclaje, automatización, visión por computación, redes neuronales, generación de imagen, lenguaje natural, clasificación, segmentación, dataset, aplicación.

Abstract– Recycling is essential for protecting the environment and conserving natural resources by transforming used materials into new products and reducing the accumulation of waste in landfills. The automation of the recycling process, both at the household and professional level, can increase efficiency and accuracy in waste separation. This article examines advances in computer vision applied to the development of tools for waste classification. The creation of instruments to simplify labeling of waste images is addressed, as well as the development of a mobile application aimed at promoting and facilitating recycling at home. Finally, the feasibility of constructing a segmentation dataset using labeled images is evaluated.

Keywords– Recycling, automation, computer vision, neural networks, image generation, natural language, classification, segmentation, dataset, application.

1 Introducción

El reciclaje es el proceso de convertir materiales usados en nuevos productos para reducir la cantidad de residuos que van a parar a los vertederos y minimizar el impacto ambiental. El reciclaje es fundamental para proteger el medio ambiente y preservar los recursos naturales.

En primer lugar, el reciclaje ayuda a reducir la cantidad de residuos que van a parar a los vertederos, lo que a su vez reduce la cantidad de espacio que se necesita para depositar-los. Los vertederos ocupan un espacio valioso y pueden ser una fuente de contaminación del aire y del agua si no se gestionan adecuadamente. Además, los residuos que se depositan en los vertederos pueden tardar décadas o incluso siglos en descomponerse, lo que aumenta aún más el problema.

• E-mail de contacto: juanma.caaz@gmail.com
• Mención realizada: Ingeniería de Computación
• Trabajo tutorizado por: Coen Antens (CVC)
• Curso 2022/2023

En segundo lugar, el reciclaje ayuda a conservar los recursos naturales. Muchos de los productos que utilizamos diariamente, como el papel, el vidrio y el plástico, están hechos de recursos naturales limitados, como los árboles, el petróleo y el gas natural. Al reciclar estos materiales, se reduce la necesidad de extraer nuevos recursos naturales, lo que a su vez reduce la huella de carbono y ayuda a preservar los recursos para las generaciones futuras.

Por otro lado, la automatización del reciclaje, tanto a nivel doméstico como profesional, puede mejorar significativamente la eficiencia y la precisión del proceso de separación de residuos. En el caso de la automatización doméstica, existen herramientas y dispositivos que pueden ayudar a clasificar los productos y saber en qué contenedor deben ir. Por ejemplo, contenedores con sensores de luz y peso que indican al usuario si un objeto debe ir al contenedor de papel, vidrio o plástico.

En cuanto a la automatización profesional, las cintas transportadoras y otras herramientas mecánicas pueden separar los residuos en diferentes categorías de manera más rápida y precisa que si se hiciera de forma manual. Además, la automatización puede reducir la cantidad de mano de obra necesaria para separar los

residuos, lo que a su vez reduce los costos y mejora la eficiencia del proceso.
Por estos motivos me he enfocado en explorar los nuevos avances que han ido saliendo en el campo de la visión por computación para aportar herramientas para la clasificación de residuos. En los últimos años, se han logrado importantes avances en el campo de la visión por computador gracias a los modelos de redes neuronales que se pueden utilizar para resolver problemas de clasificación relacionado con el reciclaje. [1]

2 Motivación

En 2020 se aprobó el Pacto Verde Europeo [2], también conocido como green deal y que tiene como objetivo lograr la neutralidad climática en Europa para el año 2050. Este plan ambicioso busca transformar la economía europea y abordar los desafíos del cambio climático y la pérdida de biodiversidad. Incluye medidas para mejorar la eficiencia energética, reducir las emisiones de gases de efecto invernadero y fomentar la transición hacia una economía circular. Además, el Pacto Verde Europeo se enfoca en la creación de empleos y en el apoyo a una transición justa para garantizar que todos los ciudadanos europeos se beneficien de una economía más sostenible. Por lo tanto creo que realizar una clasificación de estos residuos puede ayudar a cumplir con este objetivo.

La clasificación de residuos es un problema importante en la gestión de residuos y en la preservación del medio ambiente. Sin embargo, la eficiencia de los modelos de clasificación existentes todavía puede mejorarse. En este trabajo de investigación, se abordarán diversas cuestiones relacionadas con la clasificación de residuos utilizando técnicas de inteligencia artificial.

Para aportar soluciones a este problema, se han definido los siguientes objetivos:

1. Generar un gran dataset para cubrir la mayor cantidad de residuos reciclables y no reciclables utilizando utilizando una aplicación móvil para etiquetar de manera mas colaborativa.

2. Generar un dataset semántico a partir de uno de clasificación, ya que importante reducir el coste y el tiempo de etiquetado de los datos.

3. Facilitar el reciclaje de residuos utilizando una aplicación movil que permita a los usuarios clasificar sus residuos de manera automática.

Con estos objetivos se busca mejorar la clasificación de residuos y contribuir a la gestión sostenible de los mismos.

3 Estado del Arte

En los últimos años, se han logrado importantes avances en el campo del Deep Learning, esto gracias a la aparición de los Transformers. Presentados en el paper de Attention is all you need [4].

Los transformers son clave en modelos de inteligencia artificial porque han demostrado ser muy efectivos en tareas de procesamiento del lenguaje natural (NLP, por sus siglas en inglés) y otras aplicaciones relacionadas con el aprendizaje automático, incluyo extrapolando el su uso a todo tipo de tareas.

3.1. Clasificación

La clasificación ha dado un cambio radical con la salida del CLIP [8]. CLIP es un modelo de aprendizaje profundo que ha cambiado el estado del arte en la clasificación de imágenes. Utiliza un modelo de lenguaje para entender tanto las imágenes como el texto relacionado y una técnica de aprendizaje por similitud para clasificar imágenes en función de su similitud con una determinada descripción textual.

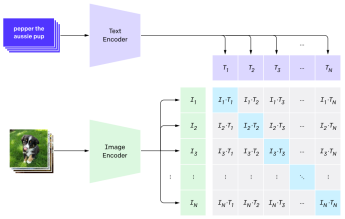


Fig. 1: Arquitectura de CLIP, por OpenAI.

En comparación con los modelos clásicos de redes neuronales convolucionales, CLIP puede comprender el contexto y el significado detrás de una imagen, lo que lo hace especialmente útil en situaciones en las que las imágenes pueden ser difíciles de clasificar para los modelos clásicos.

Model	Accuracy	Params	Tecnology
BASIC-L	91.1 %	2440M	Conv+Transf
CoCa	91.0 %	2100M	Transformer
Model soups	90.98 %	2440M	Conv+Transf
Model soups	90.94 %	1843M	Transformer
ViT-e	90.9 %	3900M	Transformer
CoAtNet-7	90.88 %	2440M	Conv+Transf
ViT-G/14	90.71 %	1843M	Transformer
CoCa	90.60 %	2100M	Transformer
CoAtNet-6	90.45 %	1470M	Conv+Transf
ViT-G/14	90.45 %	1843M	Transformer
DaViT-G	90.4 %	1437M	Transformer
Meta Pseudo Labels	90.2 %	480M	EfficientNet
DaViT-H	90.2 %	362M	Transformer
SwinV2-G	90.17 %	3000M	Transformer
Florence-CoSwin-H	90.05 %	893M	Transformer
Meta Pseudo Labels	90 %	390M	EfficientNet
RevCol-H	90.0 %	2158M	Pure CNN

Tabla 1: Comparativa de modelos de clasificación con mejor accuracy en imágenes del dataset Imagenet tabla extraída y adaptada de ”paperswithcode”. Modelos del 2020 al 2023-Q1.

En la tabla se puede ver una comparativa de los modelos de clasificación de imágenes más destacados del 2020 al 2023-Q1 y como los modelos basados en transformers (entrenados con el contexto) son superiores a los son puramente convolucionales. Esto se debe a que los modelos de redes neuronales convolucionales puros no pueden comprender el contexto y el significado detrás de una imagen, mientras que los modelos de redes neuronales basados en Transformer pueden comprender el contexto y el significado detrás de una imagen consiguiendo mejores resultados.

También es interesante ver que modelos con una cantidad de parámetros muy inferior a los modelos de redes neuronales convolucionales puros, como el modelo de ViT-G/14, pueden conseguir resultados mejores que los modelos de redes neuronales

convolucionales puros.

Respecto a la clasificación del lenguaje natural, también se ha visto que los transformers son las arquitecturas que mejor funcionan dejando atrás a los modelos LSTM.

3.2. Generadores

Los generadores de texto han sido una de las aplicaciones mas destacadas de los transformers. En el paper de GPT-2 [7] se presento un modelo de lenguaje que puede generar texto de manera autoregresiva.

3.3. Modelos multimodales

También ha habido una tendencia a la inteligencia general, es decir modelos que pueden realizar múltiples tareas, se ha visto que los modelos de transformers son capaces de realizar múltiples tareas, como por ejemplo el modelo de GPT-3 [6] que puede realizar tareas de clasificación de texto, generación de texto, etc. Y como otros modelos que han sido entrenados con diferentes tipos de input pueden lograr resultados sorprendentes, como por ejemplo el modelo GATO [5] por deepmind que puede realizar tareas de clasificación de texto, clasificación de imágenes, jugar a videojuegos de Atari y contestar preguntas similar a un chatbot.

En definitiva, se esta viendo que los modelos multimodales basados en transformers están logrando resultados sorprendentes en diferentes tareas. Aun así, los modelos mas tradicionales como las redes neuronales convolucionales, recurrentes y LSTM siguen siendo muy competentes en sus respectivas tareas.

4 METODOLOGÍA

Para la realización del proyecto se han determinado unas metodologías de trabajo, desde a nivel de planificación hasta a nivel de desarrollo.

4.1. Planificación

Para la planificación he decidido realizar tareas que sean ejecutadas secuencialmente para así poder llevar un mayor control del avance del proyecto. Cada fase esta compuesta de diferentes tareas a realizar. Se puede ver en el Apèndix la planificación detallada.

La metodología a utilizar en el proyecto sera Kanban, cada tarea se representara en una tarjeta Kanban y se moverá a la columna correspondiente en función de su estado. Los estados que se han definido son:

TODO: tarea pendiente de realizar.
IN PROGRESS: tarea en proceso.
DONE: tarea realizada.

Para la implementación de la metodología Kanban se ha utilizado la herramienta Trello, que permite crear tableros y gestionar las tareas de manera sencilla. En el Apèndix se puede ver el tablero Kanban.

Para todo el código desarrollado se utilizara el sistema de control de versiones Git para llevar un control de las versiones del proyecto.

4.2. Tecnologías

Para llevar a cabo el proyecto, se han seleccionado una serie de tecnologías que permitirán realizar de manera eficiente el procesamiento de imágenes y el análisis de datos, así como el despliegue de modelos de aprendizaje automático.

- OpenCV: una biblioteca de visión por computadora de código abierto que proporciona herramientas para el procesamiento de imágenes y el análisis de vídeo en tiempo real. Esta tecnología se eligió debido a su capacidad para proporcionar una amplia gama de herramientas para el procesamiento de imágenes y su eficiente despliegue de modelos de aprendizaje profundo (DNN).

- Diversas bibliotecas de Python, incluyendo SKlearn, numpy, matplotlib y pandas, para el tratamiento de datos y visualizaciones. Estas bibliotecas ofrecen un conjunto de herramientas y funcionalidades para la manipulación de datos y su representación gráfica.

- Firebase: para el despliegue de la aplicación movil. Firebase es una plataforma de desarrollo de aplicaciones móviles y web que ofrece una amplia gama de herramientas para el desarrollo de aplicaciones.

- Meta's Segment Anything Model [22]: para el procesamiento de imágenes. Este modelo de aprendizaje profundo permite segmentar objetos en imágenes dando como entrada una imagen y puntos de referencias.

- Tensorflow2 [10]: para el entrenamiento y creación de modelos multimodales. Tensorflow2 es una biblioteca de aprendizaje automático de código abierto que se utiliza ampliamente en la creación de modelos de aprendizaje profundo y su despliegue en producción.

- Stable Diffusion: para realizar la data augmentation en el entrenamiento de los modelos. Esta técnica permite la creación de nuevas imágenes a partir de las imágenes existentes para mejorar el rendimiento del modelo.

- Colab [11]: como entorno de ejecución durante la mayor parte del proyecto. La versión pro de Colab permite un mayor tiempo de ejecución y recursos computacionales, lo que es necesario para el procesamiento de imágenes y la creación de modelos de aprendizaje profundo.

- Android Studio: para el desarrollo de la aplicación móvil. Esta plataforma de desarrollo proporciona herramientas para la creación de aplicaciones para dispositivos móviles y su despliegue en dispositivos Android.

4.3. Documentación

La documentación se realizara en el lenguaje de marcado LaTeX, ya que es un lenguaje de marcado que permite crear documentos de gran calidad y que es muy utilizado en la comunidad científica.

Parte de la documentación se encontrara en el repositorio del proyecto en GitHub y todo el dataset generado sera publicado y de libre acceso en el repositorio.

5 DESARROLLO

5.1. Dataset

Unos de los principales problemas para la generación del dataset es que en cada región tiene un sistema de reciclaje diferente y por lo tanto las clases de residuos que se pueden reciclar también son diferentes por lo que ha sido necesario crear un dataset propio adaptado a Cataluña. El motivo de utilizar el sistema de Cataluña es porque llevan tiempo fomentando la separación de residuos y hay mucha información disponible sobre el tema.

Residuonvas.cat es una página web que tiene como objetivo informar a la población sobre la gestión de residuos en Cataluña. En su sitio web, promueven un sistema de clasificación y separación de residuos en diferentes categorías para su posterior tratamiento y reciclaje. A continuación, se presenta una lista de las diferentes clases de residuos para reciclar que se promueven en este sistema:

- Envase de vidrio
- Medicamentos
- Aparatos eléctricos
- Punto verde
- Envase ligero
- Pilas y baterías
- Ropa y calzado
- Orgánica
- Resta

Esta lista serán las clases que se utilizaran para la creación del dataset.

Para la recolección del dataset se ha utilizado diferentes dataset de residuos para tener una base solida para poder entrenar el modelo. También se ha desarrollado una aplicación móvil para la recolección de imágenes de residuos para poder tener un dataset mas completo y concreto de residuos domésticos.

A continuación, se presenta una tabla con el número de imágenes que se han obtenido para cada clase:

Class	Train	%	Val	%	Total
Amarillo	7371	15.15	500	10.64	7871
Azul	4230	8.7	500	10.64	4730
Marron	4980	10.24	500	10.64	5480
Medicamento	3156	6.49	400	8.51	3556
Pilas	3214	6.61	300	6.38	3514
Punto	4576	9.41	500	10.64	5076
RAEE	6258	12.87	500	10.64	6758
Resta	4443	9.13	500	10.64	4943
Ropa	5332	10.96	500	10.64	5832
Verde	5081	10.45	500	10.64	5581
Total	**48641**	**100**	**4700**	**100**	**53341**

Tabla 2: Distribución de las clases en los conjuntos de entrenamiento y validación

El motivo de no tener aun el conjunto de test es porque se esta esperando a que los usuarios de la aplicación móvil contribuyan con mas imágenes para poder tener un dataset mas completo y concreto de residuos domésticos.

Para la búsqueda del mejor modelo de momento se utilizara el conjunto de validación.

Fig. 2: Imagenes de cada clase extraidas del conjunto de entrenamiento

5.2. Aplicación móvil

Para poder recolectar imágenes domesticas reales de residuos se ha desarrollado una aplicación móvil para Android llamada EcoMate.

Ha sido diseñada para permitir a los usuarios contribuir y mantener un dataset de imágenes en tiempo real de manera sencilla. Esta aplicación fomenta la colaboración y el aprendizaje en torno a la clasificación y manejo de residuos, y está disponible para su descarga en la Play Store.

Las principales funcionalidades de la aplicación son:

1. Listado en tiempo real: EcoMate muestra la cantidad de imágenes que los usuarios han contribuido al dataset, permitiendo a los usuarios mantenerse actualizados sobre el progreso de la comunidad.

2. Ranking de contribuyentes: La aplicación incluye un ranking que muestra la cantidad total de imágenes que ha contribuido el usuario en el primer puesto, fomentando la competencia amistosa y el compromiso con la plataforma.

3. Visualización de la última imagen: Los usuarios pueden ver la última imagen con la que han contribuido, lo que les permite revisar su progreso y mantenerse al tanto de sus aportaciones al dataset.

4. Carga de imágenes desde la cámara: EcoMate permite a los usuarios agregar imágenes nuevas directamente desde la cámara de su dispositivo móvil. Una vez tomada la foto, se asigna una clase del dataset y se carga en el servidor, actualizando automáticamente los datos en tiempo real.

5. Autenticación y seguridad: La aplicación utiliza Firebase como backend, con módulos de autenticación, Firestore Database, Storage y Realtime Database activos. Además, EcoMate ofrece inicio de sesión y registro con correo electrónico, verificación de correo electrónico y funcionalidad de “olvidé mi contraseña“ para garantizar la seguridad y privacidad de los usuarios.

En el futuro, se planea agregar características adicionales, como la visualización de imágenes similares de residuos y la clasificación automática utilizando un modelo de clasificación avanzado. Estas mejoras enriquecerán aún más la experiencia del usuario y permitirán a la comunidad aprender y colaborar de manera más efectiva en torno al tema de la clasificación y gestión de residuos.

Desde que estuvo disponible en la Play Store, se ha recolectado un total de 276 imágenes de residuos, distribuidas en las siguientes clases:

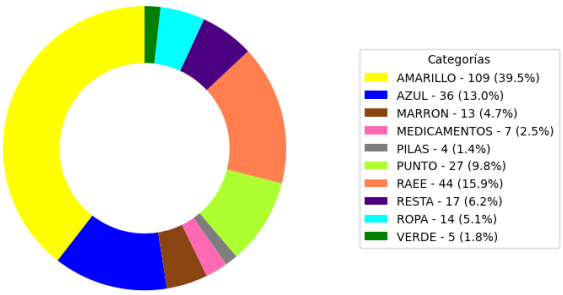


Fig. 3: Distribución de las clases en el dataset de la aplicación

La clase con más imágenes es la de AMARILLO, con 109 imágenes, mientras que la clase con menos imágenes es la de PILAS. Esta información puede proporcionar una idea de cuáles son las clases más fáciles de obtener imágenes y cuáles son las más difíciles.

5.3. Modelo de clasificación

El principal objetivo de esta sección es identificar y seleccionar el modelo óptimo para llevar a cabo la clasificación de residuos de manera eficaz y precisa. Para lograrlo, se ha empleado la plataforma TensorFlow V2, en conjunto con la API de Keras, la cual facilita el diseño, entrenamiento y evaluación de modelos de aprendizaje profundo. Asimismo, se ha utilizado una tarjeta gráfica NVIDIA RTX 3060 con 12 GB de memoria de vídeo, permitiendo agilizar el proceso de entrenamiento y mejorar el rendimiento de los modelos.

En todas las pruebas realizadas se han usado los siguientes hiperparámetros:

Hiperparámetro	**Valor**
Optimizador	Adam
Learning rate	0.001
Amsgrad	True
Loss	Categorical crossentropy
Early stopping	Monitor: val_loss, Patience: 4
ModelCheckpoint	Monitor: val_loss, Save best only: True
Número de épocas	Hasta converger

Tabla 3: Hiperparámetros utilizados en las pruebas

Se realizaron las primeras pruebas utilizando modelos simples, como una red neuronal multicapa (MLP) y una red neuronal convolucional (CNN). Estos modelos se entrenaron utilizando el conjunto de datos de entrenamiento sin realizar ninguna modificación en las imágenes.

Los valores analizados para evaluar el rendimiento de los modelos fueron la precisión con los datos de validación (Acc), la pérdida con los datos de validación (Loss), el número de parámetros (Params), el número de épocas (Epo) y el tiempo de ejecución por época en segundos (T/Epo).

Model	**Acc**	**Loss**	**Params**	**Epo**	**T/Epo**
simple_cnn	0.54	3.78	44.4M	7	184.61
simple_mlp	0.33	1.95	77.24M	22	71.35

Tabla 4: Resultados de las pruebas con modelos simples

El modelo de red neuronal convolucional (CNN) ha demostrado un rendimiento superior, ya que, al trabajar con imágenes, las CNN superan a las redes neuronales multicapa (MLP). Además, se observa que el modelo CNN requiere un tiempo de entrenamiento por época significativamente mayor que el modelo MLP, a pesar de tener menos parámetros. Esto se debe a que el entrenamiento de capas convolucionales es mucho más costoso que el de capas densas. Sin embargo, el modelo CNN ha necesitado menos épocas para converger. Los resultados de esta primera prueba nos indican que es posible desarrollar un modelo de clasificación con una precisión aceptable si optamos por utilizar una red neuronal convolucional.

En la segunda prueba, se han utilizado modelos preentrenados con el conjunto de datos ImageNet. Para esta prueba inicial, se han congelado las capas de la red preentrenada y se han añadido una capa densa de 1024 neuronas y una capa de salida con 10 neuronas. De esta manera, podemos aprovechar el conocimiento previo de la red preentrenada y entrenar únicamente las capas añadidas.

Model	**Acc**	**Loss**	**Params**	**Epo**	**T/Epo**
xception[12]	0.83	0.69	123.63M	5	280.99
inceptionresnetv2[18]	0.83	0.78	93.67M	6	510.01
nasnetmobile[20]	0.81	0.73	57.27M	4	263.30
resnet50v2[13]	0.81	1.10	126.34M	4	291.10
mobilenetv2[16]	0.80	1.06	66.49M	4	114.87
vgg16[21]	0.74	1.36	40.42M	8	221.76
convnexttiny[19]	0.53	1.67	66.37M	6	727.44

Tabla 5: Resultados de las pruebas con modelos pre-entrenados y las capas convolucionales congeladas

La mayoría de los modelos preentrenados han logrado una precisión superior al 80

Cabe destacar que los modelos preentrenados requirieron menos épocas para converger en comparación con modelos anteriores. Esto se debe a que las capas convolucionales de los modelos preentrenados ya han aprendido a extraer características de las imágenes, por lo que solo necesitan aprender a clasificar las características extraídas.

Es importante mencionar que, a pesar de ser un modelo preentrenado, el modelo convnexttiny obtuvo una precisión muy baja en comparación con los demás.

En la tercera prueba, se llevó a cabo un entrenamiento completo de los modelos preentrenados, lo que implica descongelar las capas convolucionales y entrenar todo el modelo.

Para llevar a cabo este entrenamiento, se congelaron todas las capas convolucionales y se entrenaron únicamente la capa

densa y la capa de salida durante cinco épocas. Posteriormente, se descongelaron todas las capas convolucionales y se entrenó todo el modelo hasta alcanzar la convergencia.

Además, se implementó el aumento de datos (data augmentation) debido a que, al descongelar todas las capas convolucionales, se incrementó el número de parámetros entrenables. Esto hace necesario aumentar el número de datos de entrenamiento para evitar el sobreajuste.

Las transformaciones aplicadas incluyen: rotación, traslación, volteo horizontal y ajuste aleatorio de brillo y contraste.

Se optó por utilizar un aumento de datos en tiempo real y diferente para cada época, aplicando transformaciones aleatorias a las imágenes de entrenamiento. Las probabilidades de aplicar cada transformación son:

Rotación: 0.5
Traslación: 0.5
Flip horizontal: 0.5
Random brightness and contrast: 0.5

En la tabla se muestran los resultados sin tener en cuenta el tiempo de entrenamiento de las 5 primeras épocas.

Model	Acc	Loss	Params	Epo	T/Epo
convnexttiny	0.91	0.36	66.37M	12	1799.02
xception	0.90	0.36	123.63M	10	670.20
inceptionresnetv2	0.90	0.38	93.67M	10	1149.01
nasnetmobile	0.87	0.54	57.27M	13	789.63
resnet50v2	0.86	0.54	126.34M	17	649.03
mobilenetv2	0.88	0.44	66.49M	15	358.09
vgg16	0.84	0.56	40.42M	22	689.81

Tabla 6: Resultados de las pruebas con modelos pre-entrenados y las capas convolucionales descongeladas

Ahora podemos observar que tres modelos han obtenido un accuracy superior al 90 % y lo mas importante es que el accuracy de convnexttiny ha mejorado considerablemente.

Finalmente se ha realizado una prueba con los modelos pre-entrenados sin data augmentation para comparar los resultados y demostrar que el data augmentation es necesario para obtener buenos resultados.

Model	Acc	Loss	Params	Epo	T/Epo
xception con aug	0.90	0.36	123.63M	10	670.20
xception sin aug	0.90	0.53	123.63M	6	808.19
mobilenetv2 con aug	0.88	0.44	66.49M	15	358.09
mobilenetv2 sin aug	0.88	0.70	66.49M	10	335.72

Tabla 7: Comparativa de modelos usando data augmentation y sin usar

Como se puede observar, en los dos casos, el data augmentation hace que el loss sea menor, esto quiere decir que el modelo es mas robusto y sera menos propenso al sobre-ajuste y generalizara mejor.

Sobre el tiempo de inferencia de los modelos, se ha realizado una prueba con una imagen y se ha medido el tiempo que tarda cada modelo en realizar la predicción.

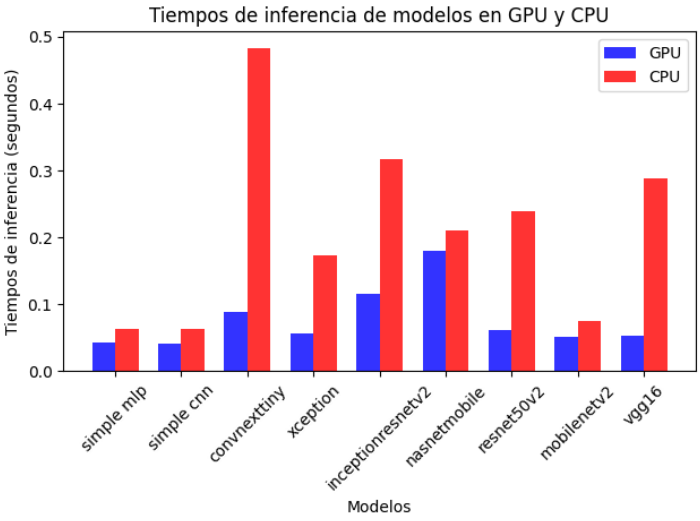


Fig. 4: Tiempos de inferencia de los modelos

Es interesante analizar el tiempo de inferencia al desplegar un modelo en un dispositivo de bajo rendimiento, como un dispositivo móvil o un dispositivo IoT.

En mi caso, dada la eficiencia de MobileNetV2, decidí utilizar este modelo para el despliegue en un dispositivo móvil, mientras que elegí el modelo Xception para realizar pruebas en el ordenador.

Una vez seleccionados los dos modelos con los que trabajar, realicé una reducción de dimensionalidad en los datos de validación para poder visualizar los resultados de manera más clara.

Para ello, utilicé la capa convolucional de Xception para la extracción de características y el algoritmo t-SNE para la reducción de dimensionalidad. De esta manera, se pasó de 1024 dimensiones a 2 dimensiones.

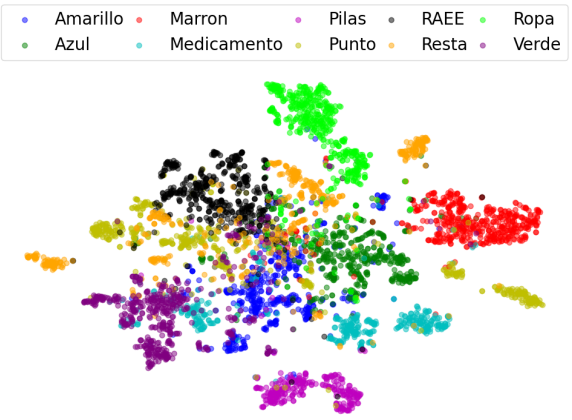


Fig. 5: Visualización de los datos de validación en un espacio de 2 dimensiones

Al observar el gráfico, notamos que las categorías Ropa y Marrón están claramente definidas y separadas. Sin embargo, las categorías Medicamento y Restos parecen menos distintas. Esta representación visual nos sugiere que el clasificador podría confundirse al asignar una observación a una de estas dos últimas clases.

Para confirmar esta sospecha, es recomendable examinar la matriz de confusión del clasificador, que muestra la cantidad de observaciones clasificadas correctamente y las que no lo fueron para cada par de clases. También es útil considerar la tasa de error entre las diferentes clases para identificar posibles problemas en la clasificación.

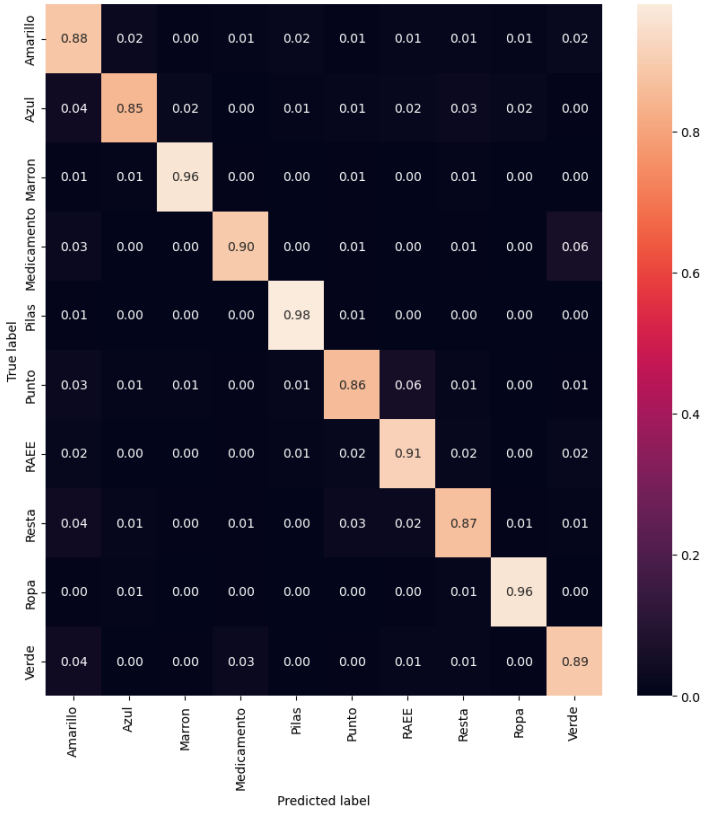


Fig. 6: Visualización de los datos de validación en un espacio de 2 dimensiones

La matriz de confusión nos permite confirmar que las clases que están cercanas en el espacio de dos dimensiones son las que tienen mayor probabilidad de confundirse, mientras que las clases más separadas tienen menos posibilidades de ser confundidas. Esto se debe a que las clases que están más cercanas en el gráfico pueden tener características similares, lo que hace que el clasificador tenga dificultades para distinguirlas.

Referencias

[1] Por que es importante reciclar https://ecoembesdudasreciclaje.es/por-que-es-importante-reciclar/

[2] Green Deal https://commission.europa.eu/strategyand-policy/priorities20192024/europeangreendeal_es

[3] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2021). High-Resolution Image Synthesis with Latent Diffusion Models. doi:10.48550/ARXIV.2112.10752

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention Is All You Need. doi:10.48550/ARXIV.1706.03762

[5] GATO (A genelist Agent) https://openreview.net/pdf?id=1ikK0kHjvj

[6] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language Models are Few-Shot Learners. doi:10.48550/ARXIV.2005.14165

[7] Language Models are Unsupervised Multitask Learners https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

[8] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. doi:10.48550/ARXIV.2103.00020

[9] PaddleOCR https://github.com/PaddlePaddle/PaddleOCR

[10] Tensorflow https://www.tensorflow.org/

[11] Google Colab https://colab.research.google.com/

[12] Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1800-1807). https://doi.org/10.1109/CVPR.2017.195

[13] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity Mappings in Deep Residual Networks. arXiv preprint arXiv:1603.05027.

[14] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2018). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[15] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... Adam, H. (2019). Searching for MobileNetV3. arXiv preprint arXiv:1905.02244.

[16] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2019). MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv preprint arXiv:1801.04381.

[17] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. arXiv preprint arXiv:1512.00567.

[18] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Thirty-First AAAI Conference on Artificial Intelligence.

[19] Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A ConvNet for the 2020s. arXiv preprint arXiv:2201.03545.

[20] Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning Transferable Architectures for Scalable Image Recognition. arXiv preprint arXiv:1707.07012.

[21] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

[22] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., ... Girshick, R. (2023). Segment Anything. arXiv preprint arXiv:2304.02643.

APÉNDICE

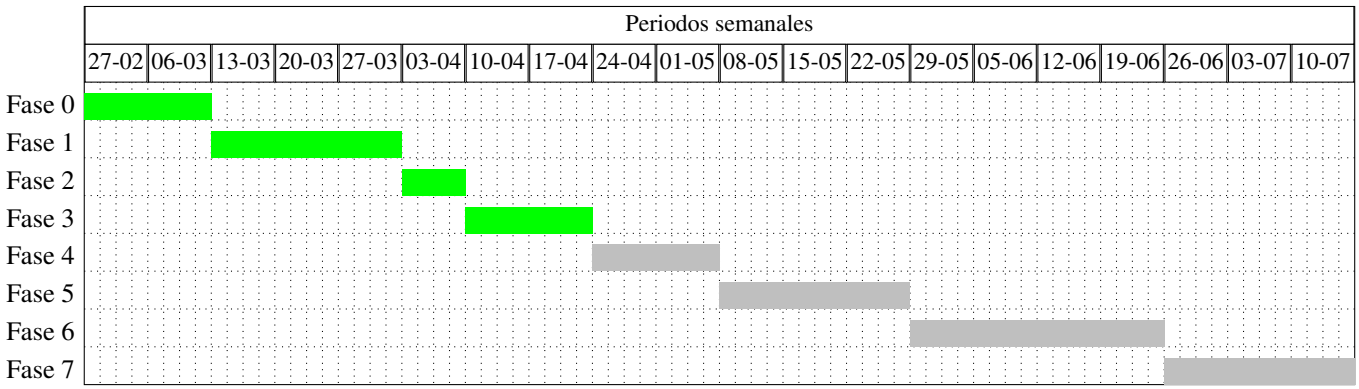
A.1. Planificación

Para la planificación he decidido realizar tareas que sean ejecutadas secuencialmente para así poder llevar un mayor control del avance del proyecto. Casa fase esta compuesta de diferentes tareas a realizar.

- **Fase 0 – Planificación del proyecto (2 semanas)**
  - Pensar el enfoque del proyecto.
  - Creación de la petición de TFG.
  - Validación y adaptación del TFG.
- **Fase 1 – Creación del dataset (3 semana)**
  - Pensar qué clases se usarán para clasificar.
  - Buscar/Crear imágenes de residuos.
  - Clasificar las imágenes en las diferentes clases.
- **Fase 2 – Creación de la aplicación (1 semana)**
  - Pensar en las funcionalidades de la aplicación.
  - Configurar firebase.
  - Diseñar el logo de la aplicación.
  - Crear la aplicación cliente.
  - Subir la aplicación a la Play Store.
- **Fase 3 – Buscar el mejor modelo de clasificación (2 semanas)**
  - Entrenar modelos de clasificación con las capas convolucionales congeladas.
  - Entrenar modelos de clasificación con las capas convolucionales y con data augmentation.
  - Comparativa y evaluación de los modelos entrenados.
  - Prueba de velocidad de inferencia en CPU y GPU.
- **Fase 4 – Buscar imagenes simulares (2 semanas)**
  - Extraer características de las imágenes del dataset.
  - Entrenar Nearest Neighbors para buscar imágenes similares.
  - Buscar alternativas de búsqueda de imágenes similares.
- **Fase 5 – Crear dataset de segmentación no supervisado (3 semanas)**
  - Instalar y configurar el entorno de trabajo de SAM.
  - Utilizar grad-CAM para obtener las regiones de interés de las imágenes.
  - Utlizar Weakly Supervised Semantic Segmentation (WSSS) para segmentar las imágenes.
  - Adjuntar WSSS con SAM para obtener el dataset de segmentación no supervisado.
- **Fase 6 – Agregar funcionalidades a la aplicacion (4 semanas)**
  - Agregar funcionalidad de búsqueda de imágenes similares.
  - Agregar la funcionalidad de clasificación de residuos.
  - Agregar la funcionalidad de segmentación de residuos.
- **Fase 7 – Fiscalización del proyecto (3 semanas)**
  - Finalizar informe.
  - Creación del póster.
  - Creación de la presentación.

A.2. Distribución temporal

Como se ha explicado comentado anteriormente, las fases se realizarán de manera secuencial. Hasta no terminar una fase no se podrá empezar la siguiente.



A.3. Trello

Para llevar un control de las tareas a realizar he decidido utilizar Trello.

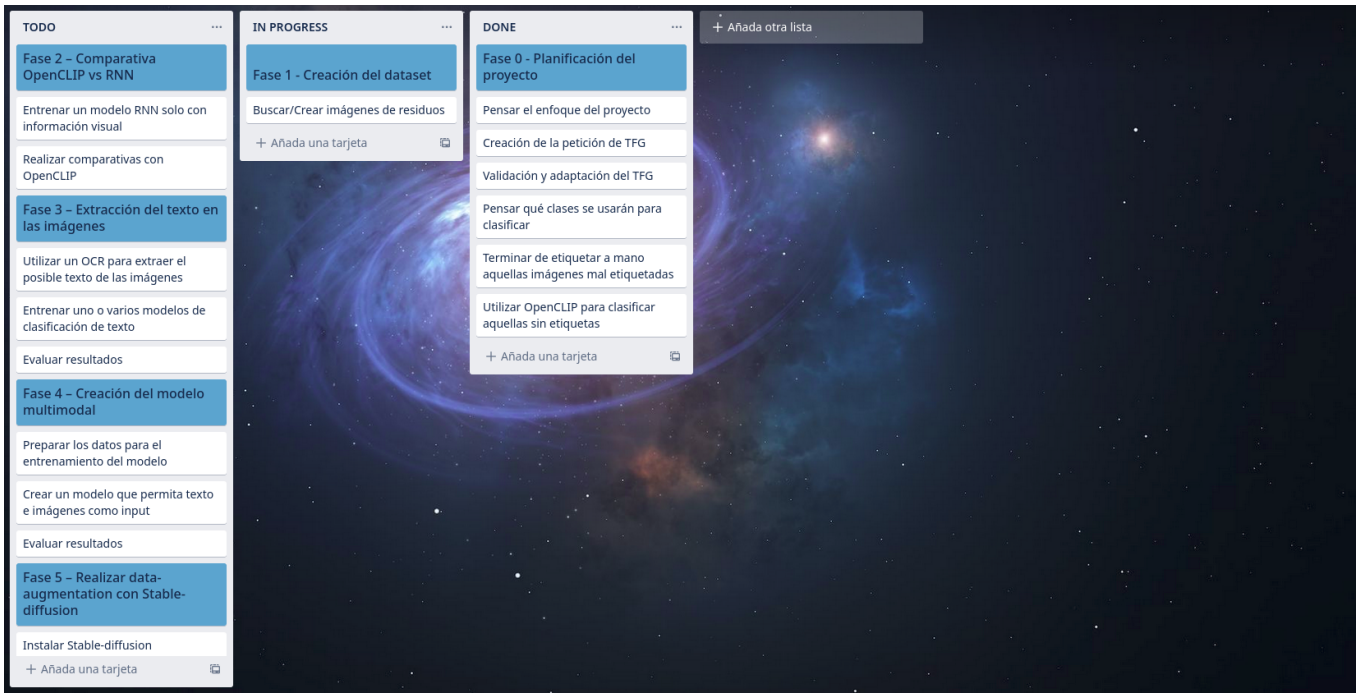


Fig. 7: Tablero de Trello antiguo

Por cambios en el plan de trabajo, se ha modificado el tablero de Trello.

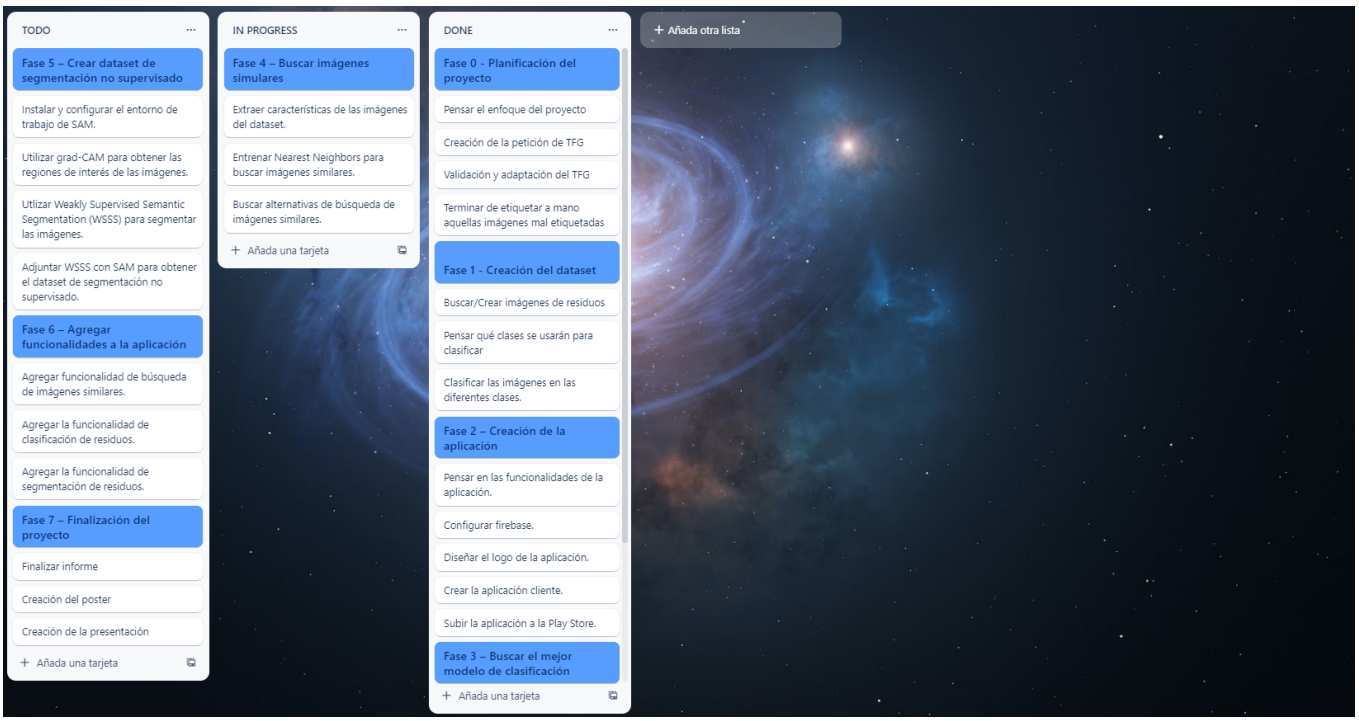


Fig. 8: Tablero de Trello actual

A.4. Aplicacion movil

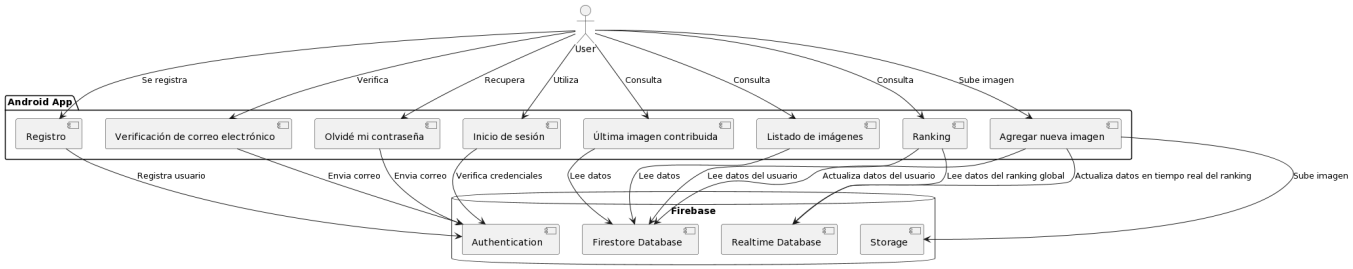
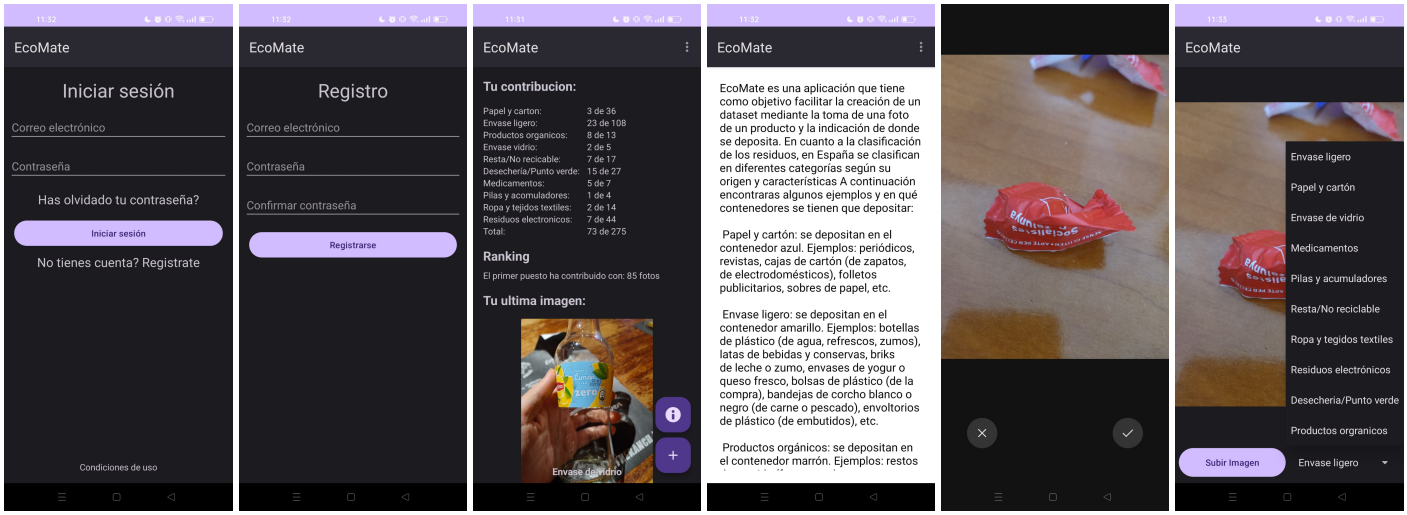


Fig. 9: Diagrama de componentes de la aplicación Android para contribuir y visualizar imágenes en tiempo real utilizando Firebase como backend.



(a) Inicio de sesión (b) Registro (c) Menú principal (d) Manual de uso (e) Tomar foto (f) Clasificación

Fig. 10: Interfaz de la aplicación Android.

A.5. Modelos de clasificación

Model	Train acc	Train loss	Val acc	Val loss	Params	Epochs	Time (s)/Epoch	Time (s)
convnexttiny_no_freeze	0.57	1.17	0.91	0.36	66.37M	12	1799.02	21588.27
xception_fine_no_freeze	0.62	1.07	0.90	0.36	123.63M	10	670.20	6702.02
xception_fine_no_freeze_no_aug	0.99	0.02	0.90	0.53	123.63M	6	808.19	4849.13
inceptionresnetv2_no_freeze	0.65	0.98	0.90	0.39	93.67M	12	1306.87	15682.44
inceptionv3_no_freeze	0.56	1.20	0.89	0.42	74.24M	19	684.87	13012.54
densenet121_no_freeze	0.55	1.22	0.89	0.42	58.43M	13	1050.03	13650.33
mobilenetv2_no_freeze_no_aug	1.00	0.02	0.88	0.70	66.49M	10	335.72	3357.20
mobilenetv2_no_freeze	0.58	1.15	0.88	0.44	66.49M	15	358.09	5371.32
nasnetmobile_no_freeze	0.60	1.10	0.87	0.54	57.27M	13	789.63	10265.25
resnet50v2_f_no_freeze	0.55	1.22	0.86	0.54	126.34M	17	649.03	11033.57
mobilenetv3large_no_freeze	0.56	1.20	0.86	0.52	51.18M	18	352.28	6341.13
vgg16_no_freeze	0.54	1.25	0.84	0.56	40.42M	22	689.81	15175.75
resnet50_no_freeze	0.55	1.22	0.84	0.57	126.36M	14	698.19	9774.64
xception	0.93	0.22	0.83	0.69	123.63M	5	280.99	1404.95
inceptionresnetv2	0.92	0.23	0.83	0.78	93.67M	6	510.01	3060.03
nasnetmobile	0.92	0.26	0.81	0.73	57.27M	4	263.30	1053.21
densenet121	0.91	0.28	0.81	0.76	58.43M	4	296.39	1185.57
resnet50v2	0.93	0.24	0.81	1.10	126.34M	4	291.10	1164.40
mobilenetv2	0.95	0.18	0.80	1.06	66.49M	4	114.87	459.49
mobilenetv3small_no_freeze	0.54	1.25	0.80	0.74	29.85M	24	201.78	4842.67
inceptionv3	0.93	0.21	0.80	0.92	74.24M	6	276.80	1660.82
nasnetlarge_aug	0.92	0.24	0.79	1.02	287.24M	5	1025.08	5125.40
xception_aug	0.65	1.02	0.75	0.84	123.63M	14	280.99	3933.85
vgg16_pre	0.96	0.12	0.74	1.36	40.42M	8	221.76	1774.11
vgg19	0.91	0.27	0.71	1.34	45.73M	6	228.42	1370.51
vgg16_pre_no_freeze	0.65	0.96	0.59	1.48	40.42M	9	727.05	6543.48
simple_cnn	0.97	0.11	0.54	3.78	44.4M	7	184.61	1292.26
convnexttiny	0.73	0.77	0.53	1.67	66.37M	6	727.44	4364.65
convnextsmall	0.66	1.01	0.52	1.61	88.0M	10	1164.44	11644.40
mobilenetv3large	0.51	1.43	0.42	1.76	51.18M	13	126.94	1650.24
resnet50	0.51	1.44	0.35	1.93	126.36M	15	309.29	4639.32
mobilenetv3small	0.37	1.80	0.34	1.89	29.85M	14	84.99	1189.87
simple_mlp	0.38	1.76	0.33	1.95	77.24M	22	71.35	1569.77
efficientnetv2b0	0.15	2.27	0.11	2.31	70.16M	4	194.60	778.41
vgg19_no_freeze	0.15	2.27	0.11	2.31	45.73M	8	774.15	6193.17

Tabla 8: Resultados de la evaluación de los modelos de redes neuronales.

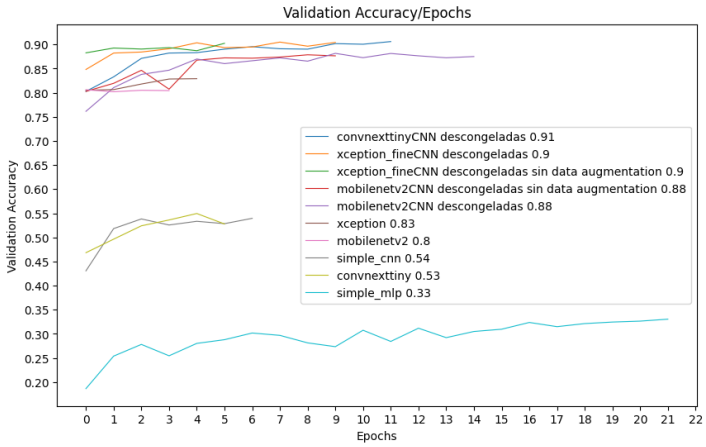


Fig. 11: Gráfica del accuracy por épocas de los modelos de redes neuronales.

Para la red neuronal simple_mlp se utilizó la siguiente arquitectura:

Capa	Tipo	Parámetros
Entrada	Aplanar (Flatten)	(224, 224, 3)
Capa Densa 1	Densa (Dense)	512, activación='relu'
Capa Densa 2	Densa (Dense)	256, activación='relu'
Capa Densa 3	Densa (Dense)	128, activación='relu'
Capa de Salida	Densa (Dense)	10, activación='softmax'

Tabla 9: Arquitectura de la red neuronal simple_mlp

Para la red neuronal simple_cnn se utilizó la siguiente arquitectura:

Capa	Tipo	Parámetros
Entrada	Convolución 2D	32, (3, 3), activación='relu', input_shape=(224, 224, 3)
Capa Max-Pooling 1	MaxPooling2D	2, 2
Capa Convolución 2	Convolución 2D	64, (3, 3), activación='relu'
Capa Max-Pooling 2	MaxPooling2D	2, 2
Capa Convolución 3	Convolución 2D	128, (3, 3), activación='relu'
Capa Max-Pooling 3	MaxPooling2D	2, 2
Capa Aplanar	Aplanar (Flatten)	-
Capa Densa 1	Densa (Dense)	512, activación='relu'
Capa de Salida	Densa (Dense)	10, activación='softmax'

Tabla 10: Arquitectura de la red neuronal simple_cnn

Para las redes convolucionales pre-entrenadas se ha usado la misma arquitectura, solo ha cambiado el modelo convolucional base. La arquitectura de las redes convolucionales es la siguiente:

Capa	Tipo	Parámetros
Base ConvNetwork	ConvNetwork	ConvNetwork
Capa Aplanar	Aplanar (Flatten)	-
Capa Densa 1	Densa (Dense)	1024, activación='relu'
Capa de Salida	Densa (Dense)	10, activación='softmax'

Tabla 11: Arquitectura usada para las distintas redes convolucionales