Capítulo 5

Aplicación de identificación de objetos

Resumen: En este capítulo se trata el desarrollo de la aplicación móvil. Esta aplicación tiene como objetivo identificar y diferenciar diversos objetos y residuos que una vez identificado se explica al usuario cuál es la forma adecuada de desecharlos y reciclarlos. La aplicación es para dispositivos con sistema operativo Android y está llevada a cabo en Android Studio con Java y la librería de TensorFlow Lite para introducir la identificación.

5.1. Introducción

Como se ha comentado, la aplicación de identificación de objetos para reciclaje es el objetivo principal de este trabajo de fin de grado. Se trata de una aplicación para dispositivos móviles la cual, utilizando la cámara del dispositivo en el que está instalada, identifica el material del objeto al que está enfocando el usuario. En la interfaz aparecen las tres opciones con más probabilidad, acompañadas del porcentaje de confianza respecto a esa opción. Independientemente de cuántas etiquetas de materiales distintas hay, siempre se muestran las tres con mayor confianza. La aplicación está constantemente interpretando lo que recibe desde la cámara, así que las etiquetas y sus respectivos porcentajes se actualizan ininterrumpidamente sin necesidad de estar tomando fotografías para cada elemento.

El desarrollo se ha centrado en dispositivos con sistema operativo Android puesto que el acceso a las otras opciones posibles, iOS y Raspberry, resultaba inviable.

5.2. Tensorflow Lite

Como se presentó en la sección 4.2, para el desarrollo del proyecto se han empleado las herramientas que brinda TensorFlow Lite, framework desarrollado por Google para aprendizaje profundo en dispositivos móviles. Además de facilitar el entrenamiento de redes neuronales como se explicó en el capítulo anterior, también simplifica la importación de los modelos generados y su uso en distintas plataformas. Los modelos de TensorFlow Lite pueden manejarse desde dispositivos Android, iOS y Raspberry, y para el desarrollo de una aplicación para cada una de estas opciones pueden utilizarse Android Studio, Swift y Objective-C; y Python, respectivamente.

Para el correcto funcionamiento, en Andorid, de la aplicación con el modelo entrenado son necesarias dos librerías. La primera es la librería de tareas de TensorFlow Lite, que cuenta con un conjunto de bibliotecas específicas de tareas potentes y fáciles de usar por los desarrolladores para crear experiencias de aprendizaje automático. Esta funciona varias plataformas y es compatible con Java y C++. La segunda librería necesaria es la de compatibilidad, esta facilita la integración de modelos en la aplicación. Proporciona API de alto nivel que ayuda a transformar los datos de entrada en el formato requerido por el modelo, además de interpretar su salida.

5.3. Identificación de objetos

La aplicación desarrollada está basada en el ejemplo que proporciona TensorFlow Lite sobre reconocimiento de flores[1]. El funcionamiento es similar pero se ha adaptado de identificar flores a diferenciar distintos materiales. Como se comentó en la sección 4.3, cuando se explicaba el entrenamiento de la red neuronal, como prototipado se ha trabajado simplemente con tres materiales (metal, vidrio y plástico). Para el correcto funcionamiento de la aplicación en Android es importante que haya al menos tres objetivos a identificar; esto es debido a que la librería de TensorFlow Lite está planteada de tal forma que muestra las tres opciones sobre las que tiene más confianza de qué se trata. Si se utilizan menos de tres no funciona.

Para utilizar TensorFlow Lite en primer lugar se han de incluir las librerías para poder utilizar sus funcionalidades. Una vez hecho esto se importa el intérprete, este carga y ejecuta el modelo dándole una serie de datos de entrada. Tras ejecutarlo escribe los resultados por pantalla.

La aplicación hace uso de la cámara del dispositivo, lee *frames* desde esta y los convierte a imágenes, estos son los datos que se utilizan como entrada para el modelo. Dichos datos de entrada se reciben desde la cámara en formato *bitmap*, matrices donde cada casilla tiene asignado un color y que en

[1] `https://codelabs.developers.google.com/codelabs/recognize-flowers-with-tensorflow-on-android/?hl=es-419#0`

conjunto forman una imagen. Ese *bitmap* es convertido a *byte buffer*, denominado *IMG data*, lo que lo hace legible para el modelo entrenado. Además utilizar Byte buffers como entrada permite que la API de Java sea más rápida[2]. Estos datos se cargan en el intérprete de TensorFlow Lite y finalmente se obtienen los valores del resultados. Estos valores que se obtienen son índices respecto a las etiquetas y la probabilidad de que esa imagen corresponda a esa etiqueta, y se devuelven en un array correspondiendo cada posición de este a cada una de las etiquetas disponibles.[3] Finalmente, se seleccionan las tres etiquetas con mayor probabilidad que son mostradas por la interfaz; por esto se comentaba previamente que es importante tener al menos tres objetivos de identificación para el correcto funcionamiento de la aplicación.

Respecto a la interfaz de usuario de la aplicación, se ha mantenido semejante a la proporcionada en el ejemplo. En la parte superior aparece el logo de Tensorflow, en la parte central se muestra lo que se está apuntando con la cámara y en la parte inferior se tiene un panel. En este panel es donde se encuentran las etiquetas sobre el objeto identificado y las probabilidades correspondientes a cada una. Este, además, se despliega hacia arriba y mostrando varias características del proceso. Estas son la resolución con la que se recogen las imágenes, la rotación del dispositivo y el tiempo de inferencia, que es el tiempo que tarda en detectar de qué objeto se trata. Además de estas características, cuenta con dos opciones configurables por el usuario que permiten la mejora de la aplicación. La primera es la elección de cuántos hilos utilice, estos permiten acelerar la ejecución de los operadores. Sin embargo, el aumento de hilos utilizados hará que se utilicen más recursos y potencia. La otra opción es elegir entre utilizar la CPU o la GPU. La GPU es uno de los aceleradores que TensorFlow Lite puede aprovechar. Si la GPU del dispositivo es compatible con las especificaciones de TensorFlow Lite se utilizará esta, si no, será la CPU con cuatro subprocesos. Según el dispositivo y la opción que se utilice la velocidad variará[4].

5.4. Pruebas

Con todo lo necesario desarrollado, se llevan a cabo varias pruebas para comprobar el funcionamiento de la aplicación de identificación. Para tener suficientes datos con los que comparar el funcionamiento se escogen cuatro de los modelos entrenados con los distintos *datasets*. Estos son los dos modelos extremos, correspondiendo a los compuestos al completo por imágenes sintéticas y por imágenes reales, respectivamente; el modelo óptimo, con el 70 % de imágenes generadas; y, por último, el simétrico del anterior, con el

[2] https://www.tensorflow.org/lite/performance/best_practices?authuser=1

[3] https://www.youtube.com/watch?v=JnhW5tQ_7Vo

[4] https://codelabs.developers.google.com/codelabs/recognize-flowers-with-tensorflow-on-android/?hl=es-419#7

30 % de imágenes sintéticas.

Para las comparaciones se han cogido diversos objetos y se ha estudiado qué etiqueta cuenta con el mayor valor de confianza para cada uno. También se tiene en cuenta aquellos casos en que las dos primeras etiquetas tienen porcentajes muy similares y por lo tanto van cambiando entre primera y segunda posición.

El objeto de estudio principal han sido los objetos metálicos, ya que son aquellos que han sido entrenados con imágenes generadas. En la figura 5.1 se pueden observar los resultados en los distintos casos con una lata de atún. A excepción del modelo entrenado por completo con imágenes sintéticas, los demás detectan que se trata de una lata con conveniente confianza. Algo similar ocurre al poner como objetivo la tapa metálica de un bote.

Con un tercer objeto metálico diferente los resultados cambian en el modelo con el 70 % de imágenes sintéticas. El resto se mantienen prácticamente igual, variando levemente la confianza, pero manteniendo la misma etiqueta que en los otros casos. En cambio, con el modelo mencionado se observa que el porcentaje queda equilibrado entre las tres etiquetas como se ejemplifica en la figura 5.3.

Una observación importante que se descubre, es la dificultad, en todas las opciones, para distinguir los objetos de vidrio respecto a los de plástico. Para tres objetos de vidrio diferentes todas los identifican como plástico con más de un 60 % de confianza. Esto no genera mucha preocupación puesto que como seres humanos a veces también es costoso diferenciar plástico translúcido de vidrio a simple vista.

Un problema importante que se ha observado es la variabilidad del resultado dependiendo de la iluminación. Como se puede observar en la figura 5.2 para el mismo objeto y con el mismo modelo, en este caso el generado a partir de solamente imágenes reales, se obtienen resultados diferentes al cambiarlo de posición.

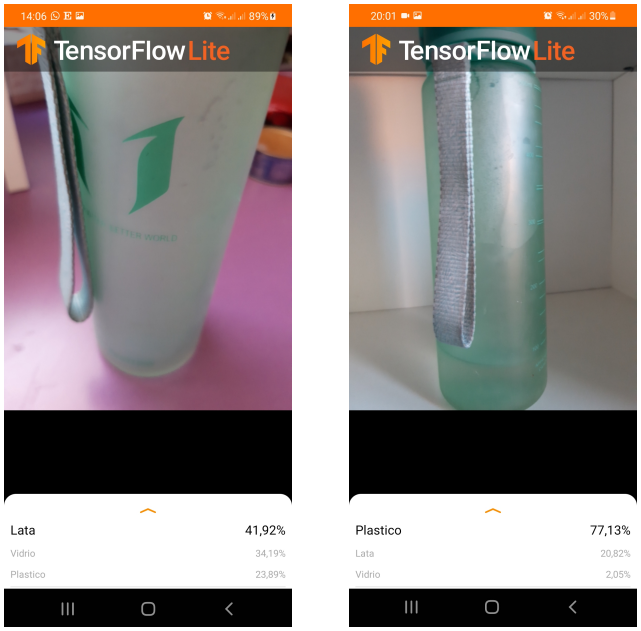
(a) 0 % imágenes sintéticas

(b) 30 % imágenes sintéticas

(c) 70 % imágenes sintéticas

(d) 100 % imágenes sintéticas

Figura 5.1: Comparación con una lata de atún



(a) 100 % imágenes reales (b) 100 % imágenes reales

Figura 5.2: Comparación con una botella de agua.

Figura 5.3: Resultados del modelo 70-30 con una lata de bálsamo labial.

Bibliografía

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. y Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015. Software available from tensorflow.org.

Alfonso, C., Estévez Estévez, R., Lobo, J. M., Lozano Diéguez, B., Prieto, F., Santamarta, J. y Gaerter, A. Emergencia climática en España. 2016.

Alsing, O. *Mobile Object Detection using TensorFlow Lite and Transfer Learning*. Proyecto Fin de Carrera, KTH, School of Electrical Engineering and Computer Science (EECS), 2018.

Fonfría, R., Sans, R. y de Pablo Ribas, J. *Ingeniería ambiental: contaminación y tratamientos*. Colección productiva. Marcombo, 1989. ISBN 9788426707420.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. y Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2017.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J. y Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. 2016.

Karim, R. *TensorFlow: Powerful Predictive Analytics with TensorFlow*. Packt Publishing, Limited, 2018.

Sánchez, M. y Castro, J. *Gestión y Minimización de Residuos*. Fundación Confemetal, 2007. ISBN 9788496743342.

SANDLER, M., HOWARD, A., ZHU, M., ZHMOGINOV, A. y CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

WANG, S.-C. *Artificial Neural Network*, páginas 81–100. Springer US, Boston, MA, 2003. ISBN 978-1-4615-0377-4.

WARDEN, P. y SITUNAYAKE, D. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, 2019. ISBN 9781492051992.