Clasificación de especies de flores usando técnicas de deep learning

Cristian Espinosa Ruiz

02 de julio 2018

Resum– La predicción y clasificación de especies de flores es difícil para la mayoría de los humanos ya que se requiere de un gran conocimiento sobre la materia y muchos años de aprendizaje y observación. Este proyecto utiliza técnicas de Deep Learning para la clasificación de imágenes de flores. La red neuronal convolucional se ha entrenado con diferentes conjuntos de imágenes, en concreto con Oxford 17, Oxford 102 y PlantNet con 10.000 especies diferentes de flores con un total de 2.600.000 imágenes.
Para dar valor a esta red neuronal se ha creado una aplicación para iOS para que cualquier usuario puede identificar flores desde la cámara de su móvil y ver/compartir con otros usuarios.

Palabras clave– Flores, CNN, Classificación, Aprendizaje automático, Predicción, Redes neuronales, iOS, CoreML

Abstract–The prediction and classification of flower species is difficult for most humans that requires great knowledge about the subject and many years of learning and observation. This project uses deep learning techniques for the classification of flower images. The convolutional neuronal network has been trained with different sets of images, specifically with Oxford 17, Oxford 102 and PlantNet with 10,000 different flower species with a total of 2,600,000 images.
To give value to this neural network, an iOS application has been created that can be identified from the camera of your mobile phone and viewed / shared with other users.

Keywords– Flowers,CNN,Classification,Machine learning, Prediction, Neural Network, iOS,CoreML

1 Introducción

La clasificación de flores para un experto resulta muy difícil y es una tarea muy laboriosa. A pesar de que pensamos que la mayoría de flores de la misma especie son iguales y podría resultar fácil poder distinguirlas no es así, nos encontramos de que diferentes flores de la misma especie pueden ser diferentes o muy similares y esto dificulta la labor de clasificarlas.
Actualmente los expertos en la materia utilizan algoritmos de árboles de decisiones entrenados con datasets, que dado diferentes rasgos de la flor da como resultado una predicción algo cuestionable como podemos observar en la Figura 1. Por este motivo nos encontramos con este reto, poder clasificar un conjunto de flores con la mayor precisión posible ayudándonos de las tecnologías que nos rodean y facilitar que tanto expertos como personas no expertas puedan distinguir entre diferentes flores gracias a las redes neuronales.

• E-mail de contacte: cristianespinosaruiz@gmail.com
• Menció realitzada: Enginyeria de Computadors
• Treball tutoritzat per: Fernando Vilariño (CVC)
• Curs 2017/18

1.1. Objetivos

El objetivo principal de este Trabajo Final de Grado consiste en la investigación, análisis y entrenamiento de una Convolutional Neural Network (CNN) para la clasificación de flores añadiendo valor con el desarrollo de una aplicación móvil. Lo que se quiere llegar a realizar con este trabajo es ver en profundidad y detalle como está creada una CNN, como funciona y poder entrenarla para su correcto funcionamiento.
No se pretende crear una mejor CNN de las que ya hay si no saber cuál es su funcionamiento, poder entrenar un dataset propio y poderlo aplicar en una aplicación móvil para ayudar a más personas.

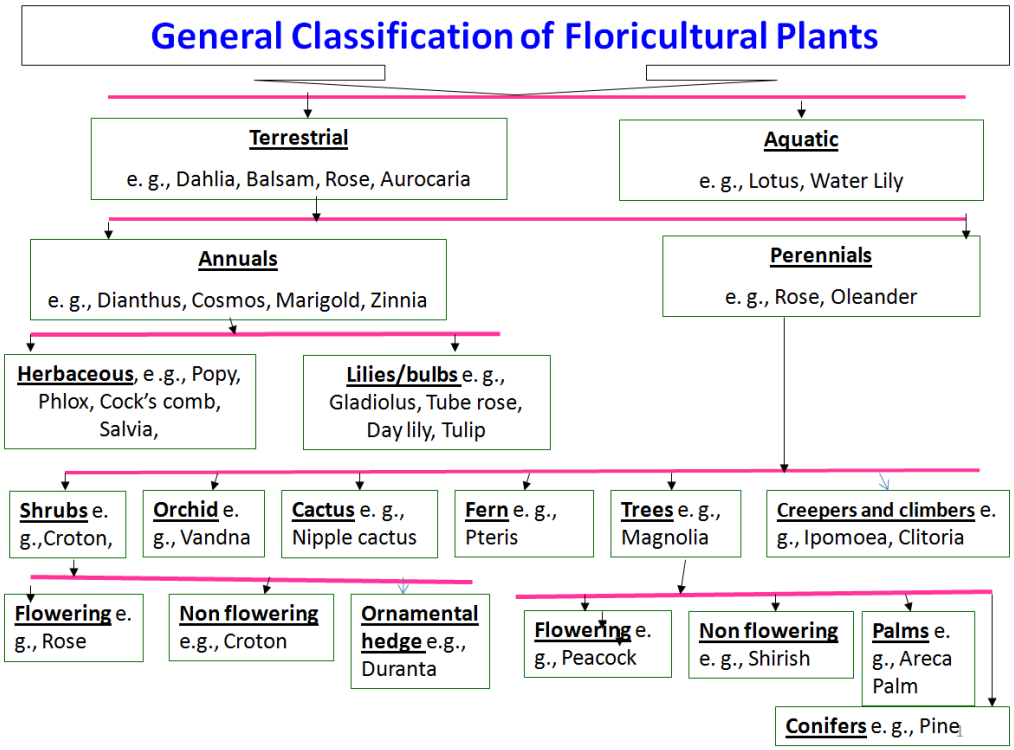

Fig. 1: Árbol de decision para la classificación de flores.

El análisis y desarrollo de este proyecto se aprenderá a utilizar herramientas como Keras [11], TensorFlow [10] y CoreML [2].
También se realizará una aplicación en iOS para integrar la red neuronal y que sea más funcional. Para ello realizaremos un sketch previo y la programación integra de la aplicación.
Para lograr todos estos objetivos se ha definido el proyecto en 4 tareas en las que incluye instalación de los IDE's y librerías, investigación y búsqueda de diferentes dataset, configuración, desarrollo de la CNN y desarrollo de la aplicación. Dado estos objetivos el proyecto queda dividido de la siguiente forma:

1. Instalación y configuración de los IDE's para el desarrollo e implementación de las CNN's como keras, tensorflow, anaconda y pycharm. (10 %)
2. Investigación y búsqueda de diferentes datasets para el buen entrenamiento de la red (20 %)
3. Configuración, desarrollo e implementacion de la CNN (40 %)
4. Desarrollo de la aplicación para añadir valor a la red implementada (30 %).

A continuación vamos a detallar como esta divido este paper, que secciones se pueden encontrar y como se ha desarrollado.

- En la sección 2 encontraremos el estado del arte actual de nuestro proyecto como de las redes neuronales.
- En la sección 3 incluye la metodología que se ha usado para el desarrollo de las CNN's como de los dataset utilizados
- En la sección 4 se realiza la configuración experimental del proyecto, la preparación del dataset y el entrenamiento de la red neuronal.
- En la sección 5 se muestra y se analiza los resultados obtenidos por cada conjunto de datasets y redes.
- En la sección 6 incluye la implementan de la aplicación.
- En la sección 7 se desarrolla las conclusiones obtenidas en el proyecto.

2 Estado del arte

En la actualidad diferentes empresas entre ellas Kaggle [1] realiza concursos diarios sobre el análisis de una gran cantidad de datos para realizar predicciones. Entre ellos hemos encontrado concursos para la clasificación de imágenes como flores, animales, coches etc. Millones de usuarios utilizan a diario esta plataforma para aprender, concursar y ayudar a otros usuarios a introducir la inteligencia artificial en diferentes aspectos de datos.
Por ello vamos a tomar como referencia datos como redes más utilizadas, diferentes tipos de preparación de dataset, tratamiento de imágenes etc.
Empresas como Apple y Google están apostando por la evolución de las redes neuronales. Apple introdujo en 2016 CoreML [2], incoporó machine learning directamente en sus dispositivos sin depender de servidores externos para el reconocmiento, clasificacion, vision y lenguaje natural. Este año 2018, Apple ha actualizado la libreria a CoreML2 [**?**] que ha definido un software propio para la creación de una red neuronal basado en imágenes con solo arrastrar una carpeta. Esto quiere decir que siguen apostando por

el aprendizaje automático y que estan proporcionando las herramientas suficientes a los programadores para que esto sea posible.
Google tambien lo ha introducido en 2018 con MLKIT [3]. Con estas dos empresas se puede ver que la inteligencia artificial cada día esta más cerca de nosotros y quieren hacernos la vida más fácil con la ayuda de esta tecnología.
Ha día de hoy nos encontramos con la rama del Machine Learning y Deep learning. Estas dos subcategorias de la Inteligencia Artificial se consideran por poder aprender según la entrenes con un conjunto de datos con la finalidad de construir una red neuronal capaz de predecir como un humano. Para la clasificación de imágenes se utilizan las CNN y podemos ver como esta diseñada en la Figura 2. En

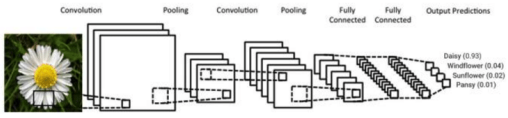

Fig. 2: Diseño de una Convolution Neural Network

la Seccion 3.1 se podrá ver en detalle como esta construida y que componentes la forman. Se puede ver en la página de ImageCLEF [9] que unos de los challanges que se realiza anualmente es la clasificación de flores. En los papers proporcionados por ImageCLEF [**?**] podemos tener una referencia de las redes con mejores resultados entre ellas ResNet50, VGGNet e Inception V3.

3 METODOLOGÍA

El proyecto se desarrolla en el lenguaje Python utilizando las librerías Keras y Tensorflow que incluye los mecanismos necesarios para poder entrenar una red neuronal de forma intuitiva y rápida. Las librerías utilizadas deben ser compatibles con GPU para el adecuado entrenamiento ya que entrenar una red neuronal con CPU puede ser imposible y muy costoso en tiempo. Para mostrar los diferentes resultados obtenidos con el entrenamiento se utiliza la librería Plot que desarrolla una gráfica con el accuracy y loss obtenido. El proyecto se va a dividir en dos grandes secciones.
La primera sera la investigación desarrollo y entrenamiento de la CNN. La segunda sera el desarrollo de la aplicación y el pleno funcionamiento con la red neuronal anteriormente entrenada.
Como metodología ágil se ha utilizado una basada en sprints 3. Cada sprint tiene 4 partes, planteamiento, diseño, desarrollo y test. Por lo tanto el proyecto se basa en diferentes sprints que tienen una duración de una semana con la siguiente estructura:

Sprint 1	Recopilación de información de las redes neuronales para la investigación
Sprint 2	Estudio de la información recopilada
Sprint 3	Aclaraciones sobre dudas
Sprint 4	Análisis de las Convolutional Neural Network - Hello World con Keras y TensorFlow - Inicio diseño de la aplicación.
Sprint 5	Análisis de las CNN con keras y TensorFlow.
Sprint 6	Desarrollo informe progreso 1 - Desarrollo de la aplicación
Sprint 7	Desarrollo de una CNN - Desarrollo de la aplicación
Sprint 8	Desarrollo de una CNN - Desarrollo de la aplicación
Sprint 9	Análisis de errores
Sprint 10	Aprendizaje
Sprint 11	Desarrollo informe progreso 2 - Desarrollo de la aplicación
Sprint 12	Conclusiones sobre la CNN desarrollada
Sprint 13	Desarrollo informe Final
Sprint 14	Desarrollo presentación.

TABLA 1: PLANIFICACIÓN DEL PROYECTO EN SPRINTS.

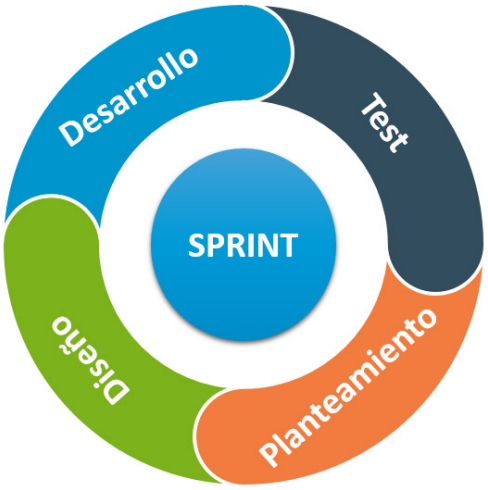

Fig. 3: Metodología ágil basada en sprints.

3.1. Convolitional Neural Network

Las redes neuronales convolucionales son una categoría de las redes neuronales y estas han demostrado ser muy eficaces en áreas como el reconocimiento y la clasificación de imágenes.
Las CNN se utilizan en áreas de investigación para el aprendizaje automático para la clasificación de cualquier tipo de imagen como TACs de paciente con tuberculosis como la clasificación de objetos en una cámara de seguridad.Como podemos observar en la figura 2 la CNN recibe una flor como entrada y se le realiza diferentes operaciones como convolución, capa de activación, Pooling y por ultimo se le añade una etiqueta donde se realiza la clasificación.
La arquitectura básica de una CNN se divide en cuatro operaciones principales:

- Convolución.
- Capa de activación.
- Pooling.
- Clasificación.

Estas 4 operaciones son los componentes básicos de una CNN.
La operación principal de las CNN como su nombre indica es la convolucion, que extrae las características principales dada una imagen de entrada.

3.1.1. Convolución

Las CNN no utilizan un filtro de convolucion codificado a mano como kernel si no que una CNN los parámetros de cada núcleo de convolución son entrenados por el algoritmo backpropagatión [15]. Cada capa puede tener diferentes núcleos que estos se utilizan a lo largo de la imagen con los mismos parámetros.
La función de esta consiste en extraer diferentes características de la imagen de entrada. Las primeras capas de la convolución suelen obtener características y detalles de bajo de nivel de una imagen como lineas y bordes en cambio cuantas más capas tenga nuestra red mas características de alto nivel recibirá.

3.1.2. Capa de activación

Para la función de activación en las CNN se utiliza principalmente la signoide ReLU que selecciona una tasa de aprendizaje adecuada y la fracción de neuronas muertas. También se puede probar con Leaky Relu y Maxout. La función que nunca se utiliza en las CNN son las sigmoide logística.

3.1.3. Pooling

El pooling en la reducción disminuye la cantidad de información en las imágenes y se queda con las características más importantes. Pooling tiene diferentes tipos : Max, Average,Sum... Como podemos observar en la figura 4 de una imagen 4x4 que contiene información importe destacada con un numero elevado e información menos importante con un numero mas cerca del 0 cuando se le aplica max pooling guarda de cada cuadrante de la imagen el máximo de información de la misma ayudando al entrenamiento de la red y reducir el ruido de las imágenes para una mejor clasificación.

3.1.4. Clasificación

La última capa de la red es una capa clasificadora que tendrá el mismo numero de neuronas como el de clases que hayamos utilizado para entrenar la red neuronal. Por ello debemos entrenar tantas redes como dataset utilicemos ya que esta depende para desarrollar la última capa. A continuación vamos a realizar el análisis de como es la arquitectura de la CNN Inception V3. Analizando las diferentes redes que nos

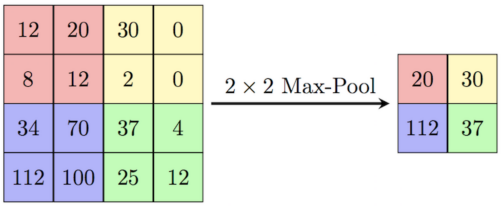

Fig. 4: Resultado de aplicar Max pooling en una imágen.

proporciona Keras, se ha optado por utilizar las siguientes redes:

- ResNet50[4].
- Inception V3[5].
- VGG16 [7].
- VGG19 [7].
- NasNet [7].

Se ha realizado un análisis de las redes utilizadas en la clasificación de imágenes y analizando los resultados de estas, las 5 anteriores redes mencionadas son las que dan mejores resultados. Esto se debe porque estas redes proporcionan mayor numero de capas que hacen que la clasificación de cada pixel sea mayor.

3.2. Metricas de evaluación

En la clasificación de imágenes la salida de la red etiqueta a esta imagen con una etiqueta donde predice con un porcentaje el acierto de la misma. Para las métricas de loss se ha utilizado la entropía cruzada. Esta métrica es la más utilizada para la clasificación de imágenes con más de dos categorías ya que por su defecto se debería utilizar una binaria. La entropía cruzada mide los bits necesarios para identificar un imagen en todo el conjunto de posibilidades es decir en el total de las clases entrenadas. La entropía cruzada se define:

$$H(p,q) = E[-logq] = H(p) + Dkl(p||q)$$

Otra métrica utilizada es la de accuracy donde evalúa el porcentaje de aciertos sobre el total de muestras. Se define con la siguiente formula:

$$Accuracy = \frac{VerdaderoPositivo + VerdaderoNegativo))}{Total}$$

3.3. DataSets

Para el entrenamiento de cualquier CNN el conjunto de datos es una de las partes más importantes para un buen entrenamiento de esta y que posteriormente de buenos resultados. Se ha echo una búsqueda exhaustiva de todos los dataset proporcionados en la red.
Los dataset seleccionados para el entrenamiento de la CNN han sido los siguientes :

- Oxford 17 flowers[4].
- Oxford 102 flowers[5].

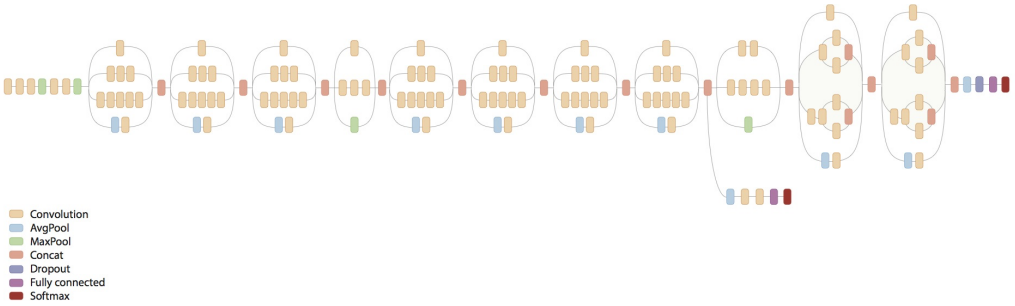

Fig. 5: Arquitectura de la Inception V3

- Plantnet con 10.000 especies de flores y 2.600.000 imagenes [7].

Como podemos observar tenemos dos conjuntos pequeños de flores Oxford 17 con un total de 17 categorías y 80 imágenes por clase y Oxford 102 con un total de 102 categorías y entre 40 y 258 imágenes por clase.
Se han utilizado estos conjuntos de datos de la Universidad de Oxford porque son los más utilizados para el desarrollo de una CNN ya que proporcionan imágenes sin ruido de cada categoria con un conjunto elevado de imagenes proporcionales. Se puede ver en [12] que diferentes investigadores realizan los entrenamientos de la CNN con los dataset Oxford anteriormente mencionados.
Por otra parte tenemos un big data de PlantNet con un total de 10.000 clases y 2.600.000 imágenes. Se ha solicitado una licencia a PlantNet y nos ha proporcionado una licencia CC-BY-SA para el desarrollo de este proyecto.
Este conjunto de datos se ha adquirido ya que es el conjunto utilizado para las diferentes competiciones a nivel mundial sobre las CNN. Se quiere ver que resultados obtendremos con nuestra CNN desarrollada y poder compararlos con los resultados obtenidos.

4 Configuración Experimental

Para el desarrollo de la red neuronal se ha utilizado el siguiente hardware. Un ordenador compuesto de una CPU E5-1620 v4 3.50GHz , 64gb de memoria ram DDR4 y una tarjeta gráfica Titan X. Para la implementación de la aplicación se ha utilizado un MacBook PRO .

4.1. Preparación del DataSet

Antes de empezar con el entrenamiento de la CNN, tenemos que realizar un pasos preliminares.
Las carpetas donde están agrupadas cada clase de flores deberemos renombrarlas con el nombre de la clase. Este paso se realiza porque hay dos maneras diferentes de entrenar una red. Se puede crear un lista que contenga de forma ordenada los nombres de las flores o nombran cada carpeta por su clase. Se ha optado por esta forma ya que si se quiere suprimir una clase basta con eliminarla la carpeta.
Hemos observado que las imágenes de los dataset no están a proporción y cada una de ella tienen una dimension diferente. Para el entrenamiento de la CNN todo el conjunto de datos tiene que tener las mismas dimensiones ya que se tiene que proporcionar por parámetro.
Se ha dimensiado las imagenes a una escala de 299x299 que es el tamaño estandard [16] utilizado para el entrenamiento de las CNN en clasificación de imágenes que no sea texto .
Una vez redimensionado todas las imágenes, debemos dividir este dataset en un conjunto de entrenamiento, validación y test con el mismo tamaño de clases. Para realizar este paso hemos diseñado otro script que dividirá el dataset según el porcentaje que deseamos. Para este primer experimento vamos a dividir el conjunto en 0.7 para el conjunto de entrenamiento y 0.2 para el conjunto de validación y y 0.1 para el test.
Con estos pasos damos como concluido la preparación del dataset y estará listo para poder entrenar de manera correcta la CNN.

4.2. Preparación de la CNN

Para el desarrollo de la CNN se ha utilizado el IDE PyCharm [8] utilizando el lenguaje Python v2.7 para su implementación.
Se ha optado por la técnica de Transfer Learning [13] ya que realizar un entrenamiento desde 0 de una CNN conlleva mucho tiempo y dedicación.
Transfer Learning es un método que nos permite usar modelos y redes ya pre-entrenados con un conjunto de datos como por ejemplo imagenet. Podemos usar los pesos y configuración ya desarrollada por la red para entrenar la nuestra.
La red no se ha conectado totalmente ya que es propenso a overfitting por ello se ha utilizado un dropout de 0.2 que como se puede ver en la Figura 6 que se optiene mejores resultados como en accuracy y en loss. Toda esta informacion se a corraborarado en [14]
Como parámetro de la capa de perdida se ha añadido la Softmax que es la más utilizada para poder predecir una solo clase de N clases.
Con todo esto hemos podido entrenar las diferentes redes anteriormente mencionas y a continuación se hará una exposición y detalle de los resultados obtenidos. Se debe tener en cuenta de que se ha optado por este método por el corto plazo de tiempo que se tiene para realizar el entrenamiento de las redes con las diferentes pruebas. En todo caso, cabe decir que normalmente se tiende a realizar un Transfer Lear-

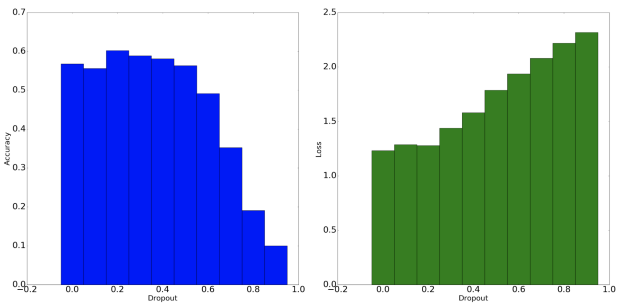

Fig. 6: Resultados obtenidos con diferentes dropout.

ning o inicializar los pesos obtenidos por redes que han sido probadas anteriormente por grandes empresas.

5 Resultados

5.1. Modelo Keras

A continuación se va a mostrar los resultados obtenidos con la red InceptionV3 utilizando el dataset Oxford 17.
Como podemos ver en la gráfica 7 los resultados del train son de 0.97 de accuracy y del 0.15 de loss. En cambio obtenemos una validación de un 0.7 en accuracy y un 1.5 en loss. Podemos observar que la validación oscila más que el propio train. Esto puede ser debido a que el conjunto de validación no sea correcto o contenga imágenes con mucho ruido. Los siguientes resultados han sido obtenidos con la

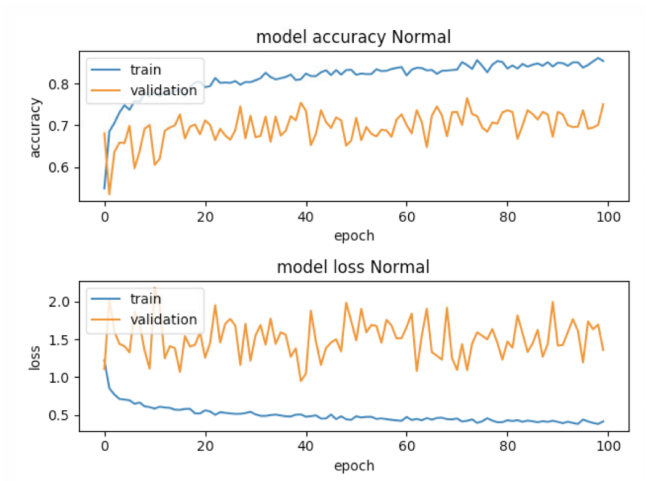

Fig. 7: Resultados CNN InceptionV3 con 17 clases y 100 epochs

misma red partiendo de los pesos modificados por la red anterior. Como podemos ver en la gráfica 8 el accuracy en train se mantiene constante en 0.98 y el validation en 0.8 después de 500 epochs.
Podemos dejar como finalizado el entrenamiento de esta red Inception V3 con un total de 0.98 de accuracy y 0.8 para la validación. Si hacemos un lectura de la gráfica los resultados son bastantes correctos y tendría un buen funcionamiento.
La siguiente gráfica 9 a analizar se trata de la CNN Nasnet. Como se puede observar la distancia entre el entrenamiento y la validación se van separando a medida van pasando los epochs. La red esta sobreentrenando y cuando intenta

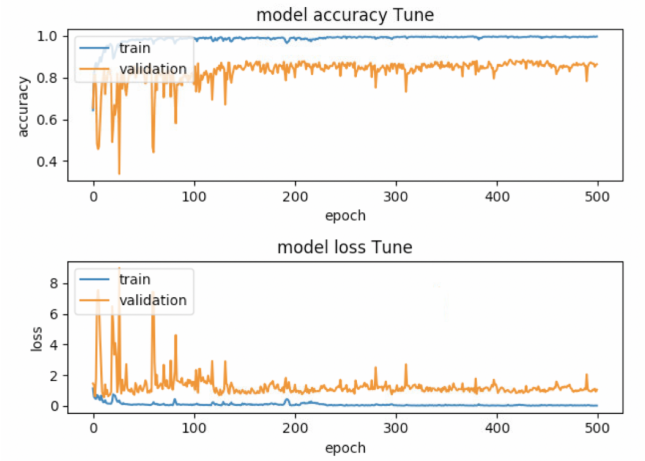

Fig. 8: Resultados CNN InceptionV3 Tune con 17 clases y 100 epochs

validar esta se equivoca y va aumentando la distancia. Esto puede ser debido a que la red Nasnet esta diseñada para distinguir conjunto de clases muy diferentes como por ejemplo un coche y un caballo. Cuando se trata de imágenes muy similares como es nuestro caso la red no llega a funcionar por la su propia arquitectura.
Después de realizar el entrenamiento con el conjunto de

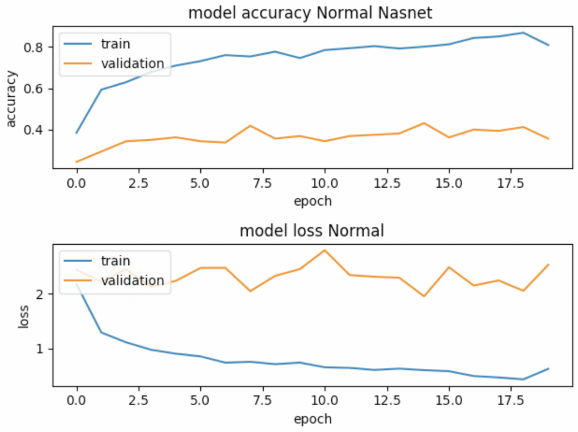

Fig. 9: Resultados CNN Nasnet

datos Oxford 17 se va a realizar el entrenamiento con el dataset Oxford 102. Se espera que los resultados sean muy similares ya que el dataset esta muy bien diseñado para el entrenamiento de la CNN.
El primer resultado obtenido ha sido utilizando la CNN Inception V3 10. Podemos observar un train muy elevado llegando incluso al 0.95 de accuracy en cambio el train no supera el 0.55. La función de perdida se puede apreciar como en la validación va aumentando y esto es debido a que se esta produciendo overfiting a partir del epoch 10. En el siguiente entrenamiento la vamos a realizar con una CNN con arquitectura nueva VGG16 11. Se puede analizar que los resultados obtenidos han sido los mejores por el momento. En la gráfica podemos observar que la validación esta por encima del train. Esto se debe a que la CNN esta entrenando de manera tan optima que la validación esta por encima. Se debe a que la información utilizada para cada conjunto de datos es tan dispersa que no hay interferencias

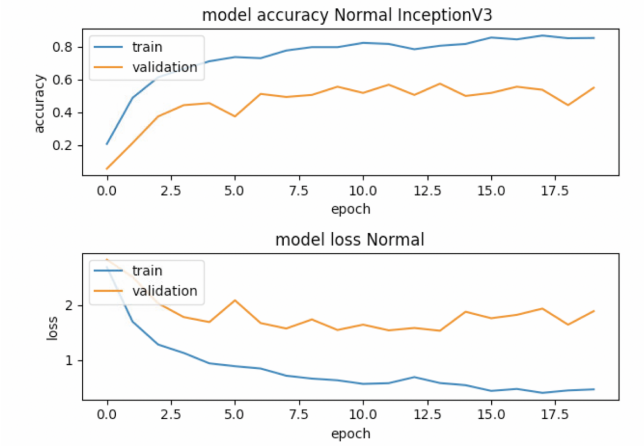

Fig. 10: Resultados CNN InceptionV3

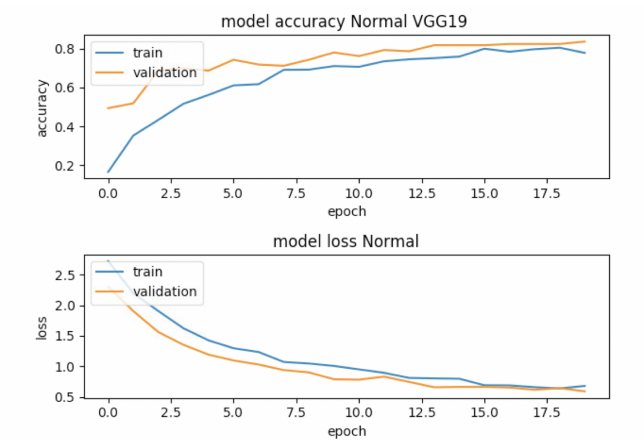

Fig. 12: Resultados CNN VGG19

entre las neuronas y así se puede obtener hasta un 0.82 de acuraccy como en train y validación.
La función de loss sucede a la inversa ya que la perdida sucede de forma gradual manteniendo el validación por debajo del train.

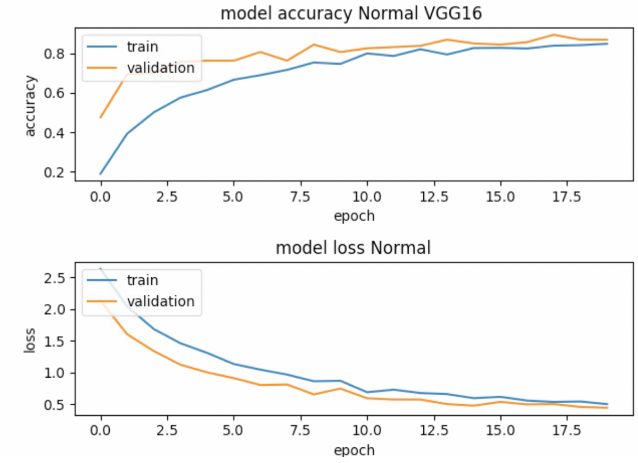

Fig. 11: Resultados CNN VGG16

La siguiente arquitectura utilizada ha sido la VGG19. Como se puedo observar en la gráfica 12 los resultados obtenidos son muy similares a la VGG16. Podemos extraer las mismas conclusiones que la anterior red ya que utilizando una arquitectura muy similar.
Seguidamente se ha realizado el entrenamiento con el conjunto de datos de 10.000 clases. Los resultados obtenidos con las anteriores arquitecturas han sido inferiores al 0.01 de accuracy. Podemos cuantificar que las redes anteriores no están preparadas para realizar aprender con ese tamaño de datos y con clases tan parecidas.
Se ha encontrado una configuración realizando un dropout de 0.5 en la red ResNet50 que hemos obtenido un total de 0.64 de accuracy y un loss que inició de 40 y llego a bajar hasta 5. Cabe decir que si se analiza el dataset con atención podemos encontrar clases con solo 3 imágenes y otras con 1.000. Esta desproporción entre clases es uno de los factores ha tener en cuenta ya que no se podrá realizar un buen entrenamiento. También ha sido limitado por hardware ya que cada epoch con este conjunto de datos tarda entre 8 y 10 horas y 4 para su validación. Con esto solo podemos realizar pocas epochs y no podemos analizar que hubiera pasado si esta se hubiera dejado entrenar con 200 o 300 epochs.
Por lo tanto al estar limitado en hardware y tiempo no se ha podido realizar más pruebas con diferentes parámetros o tratamientos de las imágenes.

5.2. Desarrollo del modelo con CoreML2

En Mayo del 2018 Apple presento su nueva tecnología con CoreML2. Han desarrollado un nuevo software totalmente de escritorio en el que cualquier usuario puede crear su propio modelo de predicción con tan solo arrastras la carpeta de imágenes ya clasificada. Como se puede ver en la figura 13 con tan solo dos lineas de código podemos crear cualquier modelo basado en imágenes. Se han realizado di-

Fig. 13: Coreml2 de Apple.

ferentes pruebas con este software y los resultados han sido los siguientes. Para el dataset Oxford 102 se han obtenido los mismos resultados que nuestra red anteriormente entrenada, con un total de 95 de accuracy en el test. Ahora bien, si esta entrenamos un conjunto grande como es el caso del dataset de 10.000 clases nos encontramos que no es posible entrenarlo. El problema viene dado en que la validación lo hace mediante CPU y esta tarda entre 8 y 10 horas en hacer una sola validación de 10.000 imágenes. Por lo tanto podemos llegar a la conclusión de que por ahora esta herramienta creada por Apple funciona para usuarios o aficionados que quieran entrenar una red clasificadora de imágenes en pocos segundos. Ahora bien, si esta se tiene que diseñar para más de 100 clases o se quieren cambiar diferentes parámetros de

momento no podremos realizarlas con CoreML.

5.3. Keras vs CoreML

Después de hacer pruebas con las dos herramientas se ha llegado a la conclusión de que CoreML esta mas orientado a pequeños dataset y para usuarios con un nivel bajo en Machine Learning. Ahora, si se quiere desarrollar una red, modificar parámetros, afinar y sin preocuparnos por el volumen del dataset se deberá desarrollar en Keras. Por todo esto hemos decidido crear el modelo en Keras.

6 Implementación de la app con modelo Keras

Para darle un valor añadido a la red neuronal y que esta la pueda utilizar cualquier usuario se ha optado a realizar una app.
El desarrollo de la aplicación se ha realizado para la plataforma iOS usando xCode y iOS 11. Se ha optado por esta opción porque era el único que ofrecía poder utilizar Machine Learning internamente en el móvil y no depender de un servidor externo.
Como se puede observar en la Figura 19 el flujo de la aplicación es el siguiente.
Primero de todo partiendo de un conjunto de imágenes tenemos que crear el modelo de la CNN con keras como hemos realizado anteriormente. Una vez obtenido este modelo, tendremos que convertir con python el modelo anterior a CoreML. Una vez echo la conversión se introducirá dentro de la aplicación donde a esta se le pasara una imagen y dará una predicción como datos de salida.
La salida contiene todos las clases con sus probabilidades correspondientes. Se ha realizado un filtrado para que de como resultado solo la clase con mayor predicción y que esta sea mayor a 0.7 ya que en el caso de no ser así no se clasificara la flor.

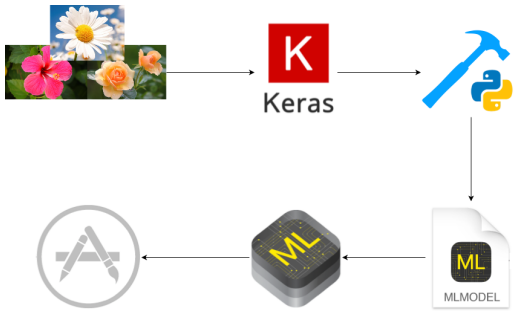

Fig. 14: Flujo de la App

El sketch de la aplicación la podemos ver en la Figura 15 donde se ha realizado las siguientes vistas.

La aplicación consta de una pequeña red social donde cada usuario tiene un perfil propio donde se ira guardando las fotografías de las flores que realice en una linea de tiempo. Este perfil también incluye una fotografía personal y un fondo que el propio usuario podrá editar.
Al clicar en la fotografía se observar la información de esta en detalle.

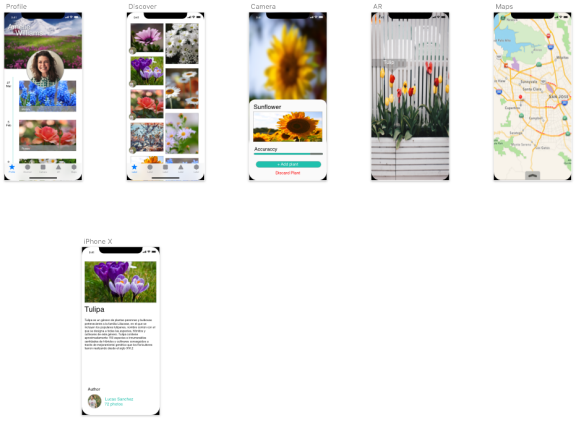

Fig. 15: Sketch de la App

La siguiente sección de la aplicación se llama Discover. En esta podemos encontrar todas las fotografías ordenadas por fecha de inclusión de todos los usuarios que utilizan la aplicación. Así podemos aprender, observar o reportar las fotografías tomadas por otros usuarios. Para tomar una fotografía y predecir que genero es pulsaremos el botón de la cámara y se procede a tomar una foto de la flor. A continuación mostrara el resultado con el nombre, una fotografía de la flor, el accuracy y dos botones uno para añadir la flor a nuestro perfil o descartarla.
Solo se obtendrá un resultado si el valor del accuracy es mayor a 0.7. Con este dato nos aseguramos de que la imagen clasificada es correcta y no se trata de otra imagen.
La última sección incluye un mapa donde podremos encontrar diferentes flores posicionadas geologicamente posicionas y podremos observar el detalle de cada una de ellas. Al tomar una foto esta se añadirá también al mapa.
Para el almacenamiento de los datos de usuario e imágenes se ha utilizado FireBase. Se ha optado por este servidor ya que facilita la integración con la aplicación. El almacenamiento en la API se realiza según el resultado. Con este dato podemos crear un dataset más grande para un posterior entrenamiento de la red y que sea más eficaz en el futuro.
En la sección apéndice podremos ver con más detalle cada una de las partes de la aplicación.

7 Conclusiones

Con el desarrollo de este proyecto podemos llegar a la conclusión de que entrenar una red neuronal para la clasificación de pocas clases puede ser rápida y menos laboriosa, en cambio, si queremos entrenar una red para la clasificación de mas de 100 clases y que estas clases tienen muchos detalles en común como colores, formas y textura puede ser mas costosa y obtendremos menor porcentaje de acierto.
Por otra parte, la importancia del conjunto de datos a utilizar puede ser el elemento decisivo para que una red entrene de forma menos o más correcta. Un buen filtrado del data set puede ahorrar mucho tiempo y obtener mejores resultados. Con las herramientas de Keras y realizando un Transfer Learning de la red se puede obtener resultados aceptables para un usuario sin conocimientos pero si se quiere diseñar una red neuronal optima esta se tendra que desarrollar de manera más técnica y usando las técnicas anteriormente

mencionadas.

Agradecimientos

Mis más sinceros agradecimientos al tutor de este trabajo Fernando Vilariño, ya que me ha ayudado a realizar y organizar este proyecto y me ha dado todas las herramientas que tenía en sus manos. También agradecer a mi pareja Marta Jordi Cazalla y a dos compañeros de la mención Manel Santiago de Toro y Alejandro Garcia Miñano por apoyar y buscar soluciones de este proyecto cuando las necesitaba. Sin ellos no se podía haber realizado este proyecto.

Referencias

[1] Kaggle empresa especializada en realizar retos de inteligencia artifical : https://www.kaggle.com

[2] Pagina oficial de CoreML desarrollado por Apple : https://developer.apple.com/machine-learning/

[3] Desarrollo de Machine Learning por Google : https://developers.google.com/ml-kit/

[4] Dataset de Oxford con 17 clases : http://www.robots.ox.ac.uk/ vgg/data/flowers/17/index.html

[5] Dataset de Oxford con 102 clases : http://www.robots.ox.ac.uk/ vgg/data/flowers/102/index.html

[6] Reto realizado por kaggle sobre la clasificación de 1000 flores : https://www.kaggle.com/c/fgvc2018-flower

[7] Corporación sobre la investigación de las flores : https://identify.plantnet-project.org

[8] IDE para el desarrollo en Python. http://www.imageclef.org/2011/plants

[9] Empresa especializada en realizar retos de inteligencia artificial : http://www.imageclef.org/2011/plants

[10] Libreria para el desarrollo de las redes neuronales : https://www.tensorflow.org

[11] Libreria para el desarrollo de las redes neuronales : https://keras.io

[12] Flower Classification with Few Training Examples via Recalling Visual Patterns from Deep CNN https://www.iis.sinica.edu.tw/papers/song/19840-F.pdf

[13] Transfer Learning https://www.learnopencv.com/keras-tutorial-transfer-learning-using-pre-trained-models/

[14] Dropout in Machine learning https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

[15] Backpropagation https://en.wikipedia.org/wiki/Backpropagation

[16] Dimension standar CNN https://www.quora.com/How-can-l-rescale-images-to-a-standard-size-to-train-a-CNN

Apéndice

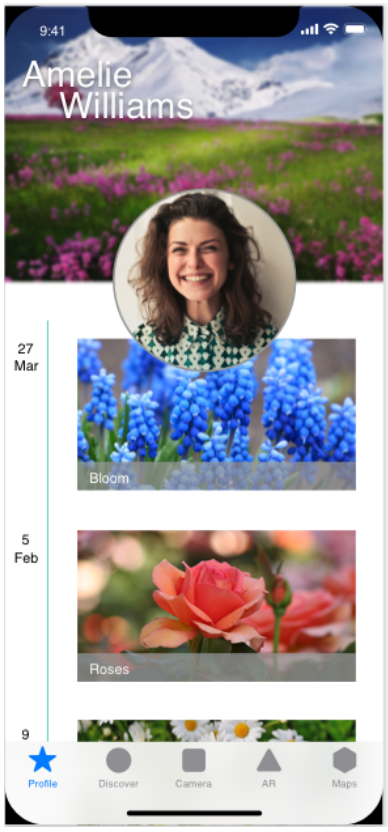

Fig. 16: Vista del perfil de la aplicación

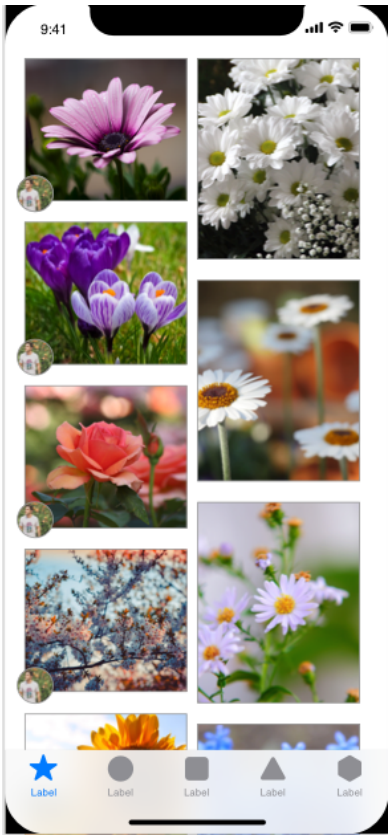

Fig. 17: Vista para ver las fotos realizadas por otros usuarios

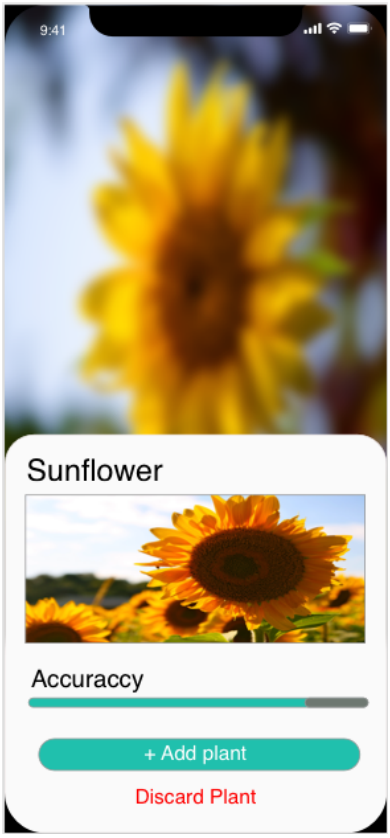

Fig. 18: Vista de la prediccion dada una foto.

Fig. 19: Detalle de la flor.