



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**UBUSETAS 1.0**



Presentado por Adrián Antón García  
en Universidad de Burgos — 9 de enero  
de 2018

Tutores:

Dr. José F. Díez Pastor

Dr. Raúl Marticorena Sánchez







UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



Dr. José F. Díez Pastor, profesor del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Adrián Antón García, con DNI 71300381J, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 9 de enero de 2018

Vº. Bº. del Tutor:

Vº. Bº. del Tutor:

Dr. José F. Díez Pastor

Dr. Raúl Marticorena Sánchez





## Resumen

En este proyecto se va a desarrollar una aplicación Android que permita, mediante la fotografía de una seta, clasificar a qué especie pertenece. Además se proporcionará información de las especies más probables, así como imágenes de estas especies para comparar con nuestra fotografía.

Para realizar la parte del reconocimiento de imágenes se aplicarán técnicas de minería de datos para usar los clasificadores de imágenes.

A la hora de recopilar la información necesaria, se usarán técnicas de Web semántica para conseguir una pequeña descripción e información interesante sobre cada especie de seta. Esta parte se realizará realizando consultas a la DBpedia.

Para completar el trabajo del clasificador de imágenes, se integrarán diferentes claves dicotómicas para el reconocimiento de géneros y especies de setas. Estas claves se conseguirán realizando técnicas de Web Scraping.

El objetivo del proyecto es el de crear una herramienta que sirva de apoyo en las tareas de micología relacionadas con el reconocimiento de setas.

## Descriptores

**Palabras clave.** *Minería de datos, Clasificador, Web Semántica, Clave dicotómica, Android, Seta, Micología, Reconocimiento*

## Abstract

In this project an Android application will be developed that allows, through the photograph of a mushroom, to classify which species it belongs to. In addition information about the most likely species will be provided, as well as images of these species to compare with our photograph.

To perform the image recognition part, data mining techniques will be applied to use the image classifiers.

When gathering the necessary information, semantic Web techniques will be used to obtain a short description and interesting information about each species of mushroom. This part will be done by consulting the DBpedia.

To complete the work of the image classifier, different dichotomous keys will be integrated for the recognition of genera and species of mushrooms. These keys will be achieved by performing Web Scraping techniques.

The objective of the project is to create a tool to support the mycology tasks related to the recognition of mushrooms.

## Keywords

**Keywords** *Data Mining, Sorter, Semantic Web, Dichotomous Key, Android, Mushroom, Mycology, Recognition*



---

# Índice general

---

|  |     |
|--|-----|
| Índice general                         | III |
| Índice de figuras                      | V   |
| Índice de tablas                       | VI  |
| Introducción                           | 1   |
| Objetivos del proyecto                 | 3   |
| 2.1. Objetivos generales . . . . .     | 3   |
| 2.2. Objetivos técnicos . . . . .      | 4   |
| Conceptos teóricos                     | 7   |
| 3.1. Micología . . . . .               | 7   |
| 3.2. Hongos . . . . .                  | 7   |
| 3.3. Setas . . . . .                   | 8   |
| 3.4. Inteligencia Artificial . . . . . | 10  |
| 3.5. Minería de datos . . . . .        | 11  |
| 3.6. Redes neuronales . . . . .        | 14  |
| 3.7. Modelo Inception . . . . .        | 20  |
| 3.8. Modelo Mobilenet . . . . .        | 21  |
| 3.9. Data Augmentation . . . . .       | 24  |
| 3.10. Web Semántica . . . . .          | 24  |
| 3.11. Web Scraping . . . . .           | 26  |
| 3.12. Clave Dicotómica . . . . .       | 27  |
| Técnicas y herramientas                | 29  |

|  |           |
|--|-----------|
| 4.1. Java . . . . .                                    | 29        |
| 4.2. Python . . . . .                                  | 29        |
| 4.3. Eclipse . . . . .                                 | 30        |
| 4.4. Apache Maven . . . . .                            | 30        |
| 4.5. Android . . . . .                                 | 30        |
| 4.6. Android Studio . . . . .                          | 31        |
| 4.7. JavaScript . . . . .                              | 31        |
| 4.8. SLF4J . . . . .                                   | 32        |
| 4.9. RDF . . . . .                                     | 32        |
| 4.10. Apache Jena . . . . .                            | 33        |
| 4.11. JSON . . . . .                                   | 35        |
| 4.12. Jaunt . . . . .                                  | 35        |
| 4.13. SQLite . . . . .                                 | 35        |
| 4.14. SPARQL . . . . .                                 | 36        |
| 4.15. DBpedia . . . . .                                | 36        |
| 4.16. Tensorflow . . . . .                             | 36        |
| 4.17. Git . . . . .                                    | 37        |
| 4.18. GitHub . . . . .                                 | 37        |
| 4.19. Zenhub . . . . .                                 | 37        |
| 4.20. SourceTree . . . . .                             | 38        |
| 4.21. Latex . . . . .                                  | 38        |
| <b>Aspectos relevantes del desarrollo del proyecto</b> | <b>39</b> |
| 5.1. Propuesta del proyecto . . . . .                  | 39        |
| 5.2. ¿Cómo implementar el clasificador? . . . . .      | 40        |
| 5.3. ¿Qué modelo de clasificador usar? . . . . .       | 42        |
| 5.4. Web Semántica . . . . .                           | 44        |
| 5.5. Clave dicotómica . . . . .                        | 45        |
| 5.6. Aspectos de diseño . . . . .                      | 46        |
| <b>Trabajos relacionados</b>                           | <b>49</b> |
| 6.1. Otros proyectos . . . . .                         | 49        |
| 6.2. Aplicaciones relacionadas . . . . .               | 50        |
| <b>Conclusiones y Líneas de trabajo futuras</b>        | <b>51</b> |
| 7.1. Conclusiones . . . . .                            | 51        |
| 7.2. Líneas de trabajo futuras . . . . .               | 52        |
| <b>Bibliografía</b>                                    | <b>53</b> |

---

## Índice de figuras

---

|  |    |
|--|----|
| 3.1. Partes de una seta [15] . . . . .   | 9  |
| 3.2. Flujos de trabajo, clasificación . . . . .  | 13 |
| 3.3. Modelo de una neurona artificial . . . . .  | 14 |
| 3.4. Funciones de activación . . . . .   | 15 |
| 3.5. Grafo de una RNA . . . . .  | 17 |
| 3.6. Izquierda: grafo de una RNA. Derecha: Grafo de una Red neuronal convolucional. . . . .                | 18 |
| 3.7. Figura que muestra una convolución en dos dimensiones. . . . .  | 18 |
| 3.8. Técnica de máx pooling. . . . .   | 19 |
| 3.9. Ejemplo de un nodo Inception. . . . .   | 20 |
| 3.10. Aplicación de una convolucion 1x1 antes de ejecutar la convolucion 5x5. . . . .                      | 21 |
| 3.11. Los filtros convolucionales normales son divididos en los filtros profundos y los puntuales. . . . . | 25 |
| 4.12. Descripciones RDF. . . . .   | 33 |
| 4.13. Arquitectura de Apache Jena . . . . .  | 34 |
| 5.14. Mobilenet v1-224, 4000 steps, 118 especies. . . . .  | 42 |
| 5.15. Comparativa modelos Mobilenet . . . . .  | 43 |
| 5.16. Inception v3, 4000 steps, 118 especies. . . . .  | 43 |
| 5.17. Mobilenet v1-224, 4000 steps, 172 especies, con data augmentation. . . . .                           | 44 |

---

# Índice de tablas

---

|  |    |
|--|----|
| 3.1. Depthwise Separable vs Full Convolution MobileNet . . . . . | 24 |
|--|----|

---

# Introducción

---

La tarea de clasificar una seta es altamente compleja ya que las diferencias entre algunas especies son mínimas y se necesitan de expertos que las analicen detalladamente hasta clasificar con seguridad la especie. Además dependiendo de la fase de crecimiento en la que se encuentre la propia seta y los factores a los que se encuentre expuesta, como puede ser la humedad del ambiente, pueden provocar que la apariencia de la seta cambie dificultando aún más esta tarea.

La clasificación de una seta debe realizarse de forma segura y minuciosa ya que hay que asegurarse que la seta no sea tóxica ni provoque perjuicios a aquella persona que la consuma.

Ante estas dificultades nace la idea de crear una aplicación que nos ayude con la identificación de la especie o género a la que pertenece una seta. Debe quedar claro desde el principio que esta aplicación solo debe servir de guía y no pretende sustituir los conocimientos de un experto, simplemente servir de apoyo o de ayuda en la difícil tarea de clasificar la especie a la que pertenece una seta.

Por estos motivos los resultados obtenidos por el clasificador deben tomarse sólo como una ayuda y usarlos para encaminar nuestra búsqueda de la especie correcta en el buen camino.

Para hacer uso de la aplicación, deberemos introducir una foto de la seta a clasificar, bien directamente sacando una foto desde la cámara o desde la galería de nuestro móvil. Con esta foto la aplicación nos mostrará las especies que el clasificador ha determinado como más probables. Si pulsamos sobre cada especie mostrada se nos mostrará información de esa especie, una breve descripción, la comestibilidad, así como diferentes imágenes que podremos comparar con la seta a clasificar. Además de mostrar esta información,

la aplicación nos hará preguntas mediante claves dicotómicas según los resultados obtenidos para identificar correctamente la seta. Las preguntas consistirán en una breve descripción de características de la seta. El usuario deberá elegir la que más se adecue a la seta para poder llegar a la especie concreta.

La aplicación se ejecuta íntegramente en el móvil, es decir, no necesitamos de una conexión a Internet para ejecutar el clasificador ni obtener información de la seta. Actualmente el clasificador es capaz de ejecutarse en el propio móvil sin necesidad de un servidor externo en un tiempo casi instantáneo permitiendo diferenciar entre 173 especies de setas diferentes. La aplicación cuenta con una pequeña base de datos que contiene información de cada especie y adicionalmente, si lo deseamos, podemos acceder por Internet al link proporcionado en cada especie a la página en la Wikipedia de la especie.

---

# Objetivos del proyecto

---

En este apartado se va a proceder a explicar los objetivos marcados en el proyecto, diferenciando entre los objetivos generales que se requerían para llevar a cabo este proyecto y los objetivos técnicos que se han ido encontrando a lo largo de su ejecución.

## 2.1. Objetivos generales

A continuación se muestran los principales objetivos del proyecto de forma general:

- El objetivo principal del proyecto es el de crear una aplicación Android que fuera capaz de identificar, mediante un clasificador de imágenes, la especie a la que pertenece una seta mediante una fotografía introducida por el usuario, bien por la cámara o desde la galería del móvil.
- Otro objetivo es el de mostrar información de las diferentes especies clasificadas. En esta información se incorporaría una breve descripción de la especie, la comestibilidad y otros datos de interés. Este objetivo se realizará mediante técnicas de web semántica.
- Como complemento al clasificador se debería implementar una serie de claves dicotómicas que mediante preguntas simples acerca de la seta conduzcan a la especie correcta de la que se trata. Este objetivo se implementará mediante técnicas de *web scraping*.
- Se intentará desarrollar una interfaz atractiva y amigable en la aplicación Android.

## 2.2. Objetivos técnicos

En esta lista se van a detallar los objetivos técnicos que se han planteado para implementar los objetivos generales descritos anteriormente.

- Usar Android Studio para implementar la aplicación Android.
- Un objetivo técnico importante que se marcó fue el de ejecutar el clasificador en el propio móvil, para lo que se utilizaron las librerías de Tensorflow para Android Studio junto a los modelos de clasificación Mobilenet.
- Usar Java como lenguaje de programación principal a la hora de programar la parte de Web Semántica y la extracción de las claves dicotómicas mediante Web Scraping.
- Utilizar Sparql como lenguaje para realizar las consultas a la DBpedia.
- Usar las librerías de Apache Jena para realizar las consultas con sparql a la DBpedia en Java y extraer la información de las especies de setas.
- Utilizar las librerías de Jaunt en java para realizar la parte de Web Scraping y poder extraer las claves dicotómicas.
- Utilizar SQLite como base de datos para almacenar los datos de las especies y las claves en Android.
- Usar un sistema de control de versiones, se ha optado por utilizar el servicio GitHub.
- Utilizar la metodología Scrum para realizar un seguimiento de las tareas realizadas durante el proyecto. Se ha elegido usar la herramienta ZenHub para este propósito.
- Se intentará que la aplicación no ocupe demasiada memoria dentro del dispositivo, para ello se formatearán las imágenes para que ocupen lo necesario y se buscará un modelo de clasificador que no sea demasiado exigente, en nuestro caso se ha optado por el modelo Mobilenet-224-v1.
- Llevar a cabo un prototipo de la interfaz de usuario mediante alguna herramienta de prototipado.
- Utilizar herramientas de testing para probar el correcto funcionamiento de nuestra aplicación.



- Hacer uso de las directrices de diseño de Material design para desarrollar la interfaz gráfica.



---

# Conceptos teóricos

---

Para la correcta comprensión del proyecto se van a explicar a continuación una serie de conceptos teóricos mínimos necesarios.

## 3.1. Micología

La micología es la ciencia que se dedica al estudio de los hongos. Es una de las áreas de la ciencia más extensas y diversificadas que aporta avances significativos a la investigación científica y al desarrollo tecnológico. [31]

Tiene varias aplicaciones y objetivos, se usa para determinar que especies de hongos son comestibles o no y si podrían usarse para diferentes tratamientos médicos, ciertas sustancias de las setas son estudiadas con estos fines curativos. [3]

## 3.2. Hongos

El hongo es el nombre común de los organismos del reino Fungi. El término Fungi en biología se refiere a un grupo de organismos eucariotas<sup>1</sup> que se clasifican en un reino distinto al de las plantas, animales y protistas. Se diferencian de las plantas en que son heterótrofos<sup>2</sup> y de los animales en que tienen paredes celulares, como las plantas, pero compuestas de quitina<sup>3</sup> en vez de celulosa.

---

<sup>1</sup> Aquellos organismos formados por células con núcleo verdadero

<sup>2</sup> Seres vivos que necesitan de otros para alimentarse

<sup>3</sup> <https://es.wikipedia.org/wiki/Quitina>

Los hongos se reproducen de forma sexual o asexual mediante esporas, que se dispersan en un estado latente y solo se interrumpe cuando se dan las condiciones adecuadas para su germinación. [25]

La mayoría de los hongos están formados por estructuras microscópicas, filamentosas y ramificadas llamadas hifas. El conjunto de estas hifas forma una red a la que llamaremos micelio. Cuando la acumulación de hifas es grande se pueden observar como una red algonodosa que se pueden reconocer por ejemplo cuando los alimentos empiezan a descomponerse. [6]

### 3.3. Setas

Las setas son los cuerpos fructíferos de algunos tipos de hongos (no todos los hongos producen setas), en otras palabras, son la parte reproductiva de los hongos. La principal función de la seta es dispersar las esporas del hongo. Normalmente es la parte visible del hongo ya que este suele estar bajo tierra.

#### Partes de las setas

A continuación se van a describir las partes más características de una seta que se usan para su clasificación.

- Sombrero: Situado sobre el pie, ejerce la función de protección en la formación y desarrollo de las esporas. El sombrero es un elemento clave a la hora de diferenciar las especies ya que puede adoptar diferentes formas, aspectos y colores.
- Himenio: Es la parte situada justo debajo del sombrero y que puede adoptar diferentes formas (láminas, tubos, aguijones o pliegues), estas diferentes formas nos ayudarán a diferenciar entre las especies. La función principal de esta parte es la de crear, desarrollar, almacenar y dispersar las esporas que generan nuevos hongos.
- Lamina: Son los tabiques del himenio que contienen los basidios<sup>4</sup>.
- Pie: Elemento que no tiene por que aparecer en todas las setas y que sujeta al sombrero e himenio.

---

<sup>4</sup>Estructuras microscópicas que generan las esporas de la seta

- Volva: Es un fragmento en forma de membrana que envuelve la base del pie en algunas setas. Es un error común cortar el pie de la seta y no conservar la volva que nos podría dar indicios de a qué especie pertenece la seta.
- Micelio: El conjunto de hifas, es decir, el hongo propiamente dicho. [7]

En la siguiente figura 3.1 se muestran las partes más relevantes de una seta.

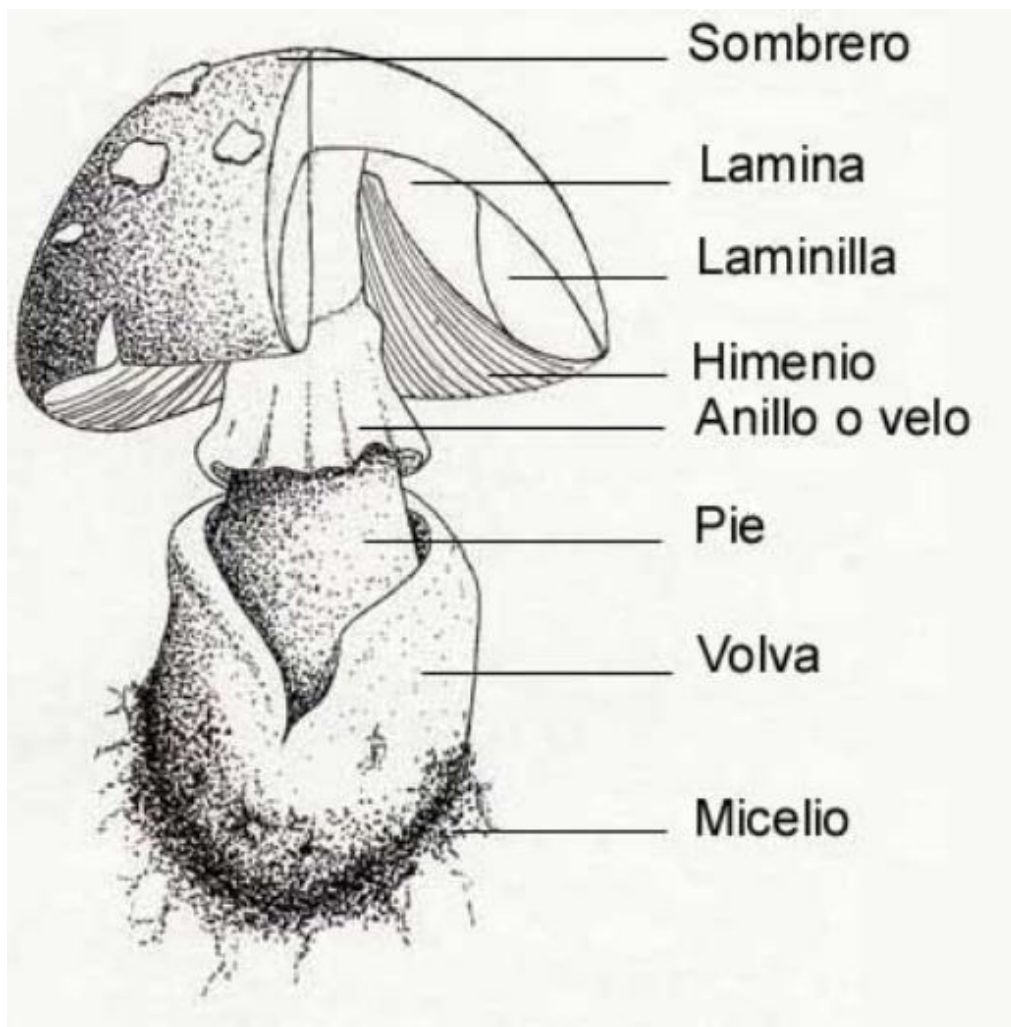


Figura 3.1: Partes de una seta [15]

### 3.4. Inteligencia Artificial

Hay muchas definiciones para definir lo que es la inteligencia artificial, normalmente aparece bajo las siglas IA o AI (del inglés Artificial Intelligence), pero de forma general la podemos definir como el desarrollo de algoritmos y métodos que otorgan a las computadoras la capacidad de realizar acciones o tareas como las realizaría un ser humano. La inteligencia artificial se aplica en multitud de campos como puede ser la robótica, automovilismo, reconocimiento de imágenes, medicina, sistemas de apoyo...

Stuart Russell y Peter Norving [26] diferencian estos cuatro tipos de inteligencia artificial:

- Sistemas que piensan como humanos: Son sistemas que intentan emular el pensamiento humano automatizando actividades que relacionamos con los procesos de pensamiento humano (toma de decisiones, resolución de problemas...), por ejemplo las redes neuronales artificiales.
- Sistemas que actúan como humanos: Este tipo de sistemas intentan imitar el comportamiento humano, por ejemplo la robótica.
- Sistemas que piensan racionalmente: Tratan de imitar con lógica el pensamiento lógico racional del ser humano, por ejemplo los sistemas expertos.
- Sistemas que actúan racionalmente: Intentan emular de forma racional el comportamiento humano.

Algunos de estos sistemas necesitan desarrollar una capacidad de aprendizaje para poder imitar de forma correcta el comportamiento humano.

#### Aprendizaje automático

El aprendizaje automático (del inglés Machine Learning) es una rama de la inteligencia artificial que se encarga de desarrollar los algoritmos que permitan a las computadoras aprender <sup>5</sup>. Para ello, se entrenan los sistemas con una serie de ejemplos, o información de entrenamiento, para generalizar comportamientos que deben realizar ante diferentes casos de entrada.

Hay una gran variedad de algoritmos de aprendizaje automático, a continuación se presentan algunos de los tipos más extendidos:

---

<sup>5</sup>Adquirir conocimiento por medio de la experiencia

- **Aprendizaje supervisado:** en estos algoritmos se intenta crear una función que nos dé la correspondencia entre los datos de entrada y una de las salidas deseadas. En este tipo de algoritmos se tiene información tanto de los valores de entrada como de los valores deseados. Para entrenar se suelen usar pares en los que un componente son los datos de entrada y otro los valores deseados. Los valores deseados suelen ser denominados como clases. [22]

El valor de salida de la función puede ser un número (Regresión) o una etiqueta de clase (Clasificación).

Los algoritmos de clasificación usados en este proyecto se corresponderían con este tipo de aprendizaje, siendo las etiquetas de clase las especies de setas que se quieren clasificar.

- **Aprendizaje no supervisado:** el aprendizaje no supervisado se diferencia del supervisado en que no se conoce a priori los resultados deseados. En este tipo de aprendizaje, normalmente, se tratan los datos de entrada como una serie de datos aleatorios y se intentan buscar relaciones ocultas entre ellos. Una forma de aprendizaje no supervisado es el clustering, el cuál se encarga de organizar los datos de entrada en grupos en los que comparten propiedades y los diferencian del resto. [21]
- **Aprendizaje semisupervisado:** Este tipo de aprendizaje mezcla las dos técnicas descritas anteriormente. Se usa cuando tenemos tanto datos conocidos como datos de los que no tenemos conocimiento.
- **Aprendizaje por refuerzo:** Este tipo de algoritmos realiza un aprendizaje mediante el modelo de prueba-error, es decir, el algoritmo realiza acciones y a partir de los resultados de estas el algoritmo evalúa cuáles han sido beneficiosas y cuáles no.

En nuestro proyecto se hace uso del aprendizaje supervisado para entrenar el clasificador de imágenes que reconocerá las especies de las setas. [20]

## 3.5. Minería de datos

La minería de datos consiste en la aplicación de técnicas de inteligencia artificial sobre grandes cantidades de datos con el objetivo de descubrir patrones o relaciones ocultas entre los datos. La parte de la minería de datos más relevante para nuestro proyecto son los clasificadores previamente introducidos [16]. Básicamente los clasificadores se basan en la hipótesis del aprendizaje inductivo:

Cualquier hipótesis que aproxime bien una función objetivo sobre un conjunto de ejemplos de entrenamiento suficientemente grande también aproximará bien la función objetivo en ejemplos no observados,

La minería de datos, típicamente, consta de los siguientes pasos:

- Selección del conjunto de datos: En esta parte del proceso se seleccionarán las variables objetivo, las variables independientes y se realizará un muestreo con los registros disponibles.
- Análisis de las propiedades de los datos: Se comprobarán los histogramas, diagramas de dispersión, si se encuentran valores atípicos entre el conjunto de datos y si faltan ciertos datos.
- Transformación de los datos: En esta etapa del proceso se procesarán los datos de entrada para adecuarlos a la estructura requerida por las técnicas de inteligencia artificial que se aplicarán sobre ellos.
- Selección de la técnica de minería de datos: Se construirá el modelo predictivo a aplicar sobre los datos. Clasificación o segmentación.
- Extracción del conocimiento: Esta es la parte donde se aplicarán las técnicas de inteligencia artificial sobre los datos para extraer el conocimiento buscado. La técnica que hayamos aplicado nos devolverá un modelo de conocimiento sobre los datos.
- Interpretación y evaluación de los datos: Esta última etapa consiste en validar si los resultados que nos ofrece el modelo entrenado son satisfactorios y se adecuan a los requisitos pedidos.

## Evolución del flujo de trabajo

Al principio, el flujo de trabajo para entrenar un clasificador, era el de extraer unas características de los datos de entrenamiento que se proporcionaban al clasificador. En este modo de trabajo se usan métodos como el Bag-of-words<sup>6</sup> para determinar las características de los datos de entrada.

Posteriormente, este flujo de trabajo se ha modificado, introduciendo redes neuronales que se encargan de extraer estas características y dividiéndolas en diferentes jerarquías para que no todas tengan el mismo peso a la hora de clasificar.

---

<sup>6</sup>Por ejemplo, tratar las características de una imagen como palabras y guardar las veces que se repiten estas características en cada imagen. [https://es.wikipedia.org/wiki/Modelo\\_bolsa\\_de\\_palabras](https://es.wikipedia.org/wiki/Modelo_bolsa_de_palabras)



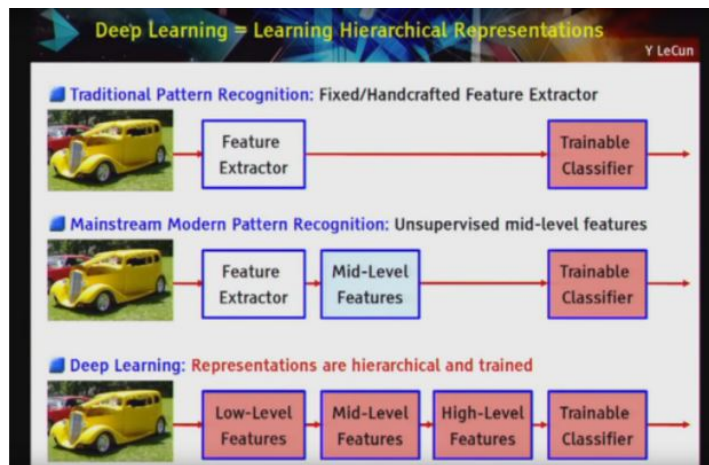


Figura 3.2: Flujos de trabajo, clasificación

En la figura 3.2 se muestra como se han ido incorporando diferentes capas, todas de ellas entrenables, para extraer las características de una forma jerárquica mediante redes neuronales.

### 3.6. Redes neuronales

En esta sección se explicarán algunos conceptos teóricos sobre redes neuronales necesarios para entender el funcionamiento de los modelos Mobilenet e Inception usados para clasificar imágenes en nuestro proyecto. Actualmente hay gran variedad de clasificadores que se pueden usar para esta tarea además de los dos mencionados.

#### Modelo de neurona artificial

Una neurona artificial de forma general se puede describir según el modelo de Rumelhart y McClelland (1986), el cual define la neurona o elemento de proceso (EP) como un dispositivo el cuál a partir de un conjunto de entradas, vector  $x$ , genera una única salida  $y$ . [5]

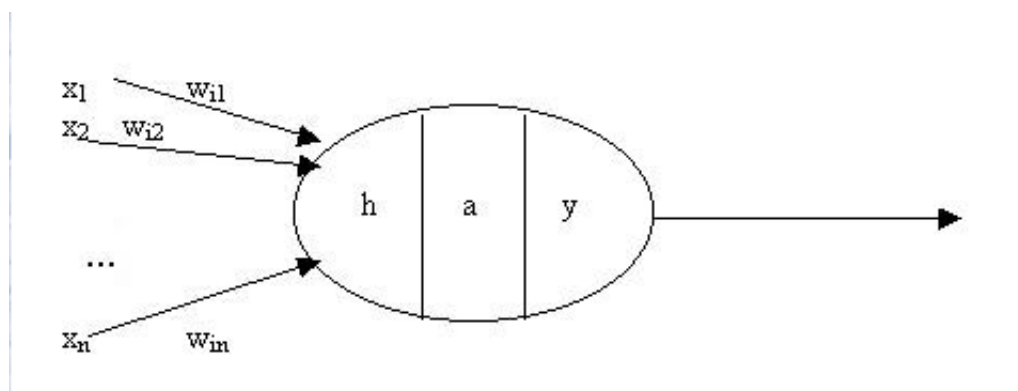


Figura 3.3: Modelo de una neurona artificial

La neurona artificial consta de los siguientes elementos:

- Conjunto de entradas  $x$ : El vector  $x$ .
- Conjunto de pesos sinápticos  $w_{i,j}$  : Representan la relación entre la neurona  $i$  y  $j$ .
- Regla de propagación: proporciona el potencial postsináptico,  $h_i(t)$ . Se suele representar como una suma ponderada de la siguiente forma 3.1

$$h_1(t) = \sum (w_{1j} * x_j) \quad (3.1)$$

- Función de activación  $a$ : proporciona el estado de activación en función del estado anterior y el potencial postsináptico.
- Función de salida  $y$ : proporciona la salida  $y$  en función del valor de activación. En la figura 3.4 podemos ver diferentes funciones de activación.



Figura 3.4: Funciones de activación

## Red Neuronal Artificial RNA

Una red neuronal Artificial RNA es un grafo dirigido en el que las neuronas artificiales están conectadas con otras y los enlaces pueden aumentar o disminuir el estado de activación de las neuronas conectadas. Cada neurona opera de forma individual mediante operaciones de suma.<sup>[33]</sup>

Estas neuronas se organizan en capas o layers formando diferentes filtros para generar algoritmos que puedan resolver diferentes problemas. Dentro de una misma capa las neuronas suelen ser del mismo tipo y podemos diferenciar de forma general 3 tipos distintos de capas:

- De entrada: Reciben datos o señales externos del entorno.
- De salida: Proporcionan las respuestas de la red ante los estímulos recibidos por las capas de entrada.
- Ocultas: No interaccionan con el entorno, son nodos intermedios que ejecutan las operaciones internas de la red.

Como se puede observar en la figura 3.5, las neuronas representadas como nodos se organizan en las diferentes capas pudiéndose interconectar con varias neuronas a la vez. Estas redes pueden tener un número de capas y neuronas ilimitados. Pueden tener un número ilimitado de enlaces o solo estar conectadas con unas pocas.

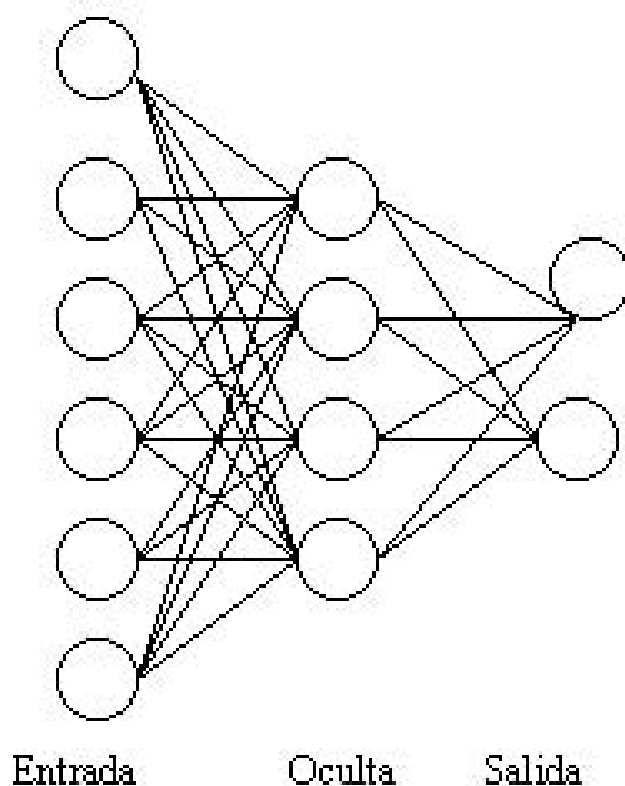


Figura 3.5: Grafo de una RNA

## Redes neuronales convolucionales

Este tipo de redes neuronales son muy parecidas a las RNA normales pero tienen dos cambios significativos. El primero es que parten de la idea de que las entradas son imágenes lo que permite programar optimizaciones a los algoritmos para que se adecuen a este tipo de entradas, el segundo cambio es que intentan reducir en la mayor medida posible el número de parámetros utilizados en cada capa.<sup>[4]</sup>

Las redes neuronales convolucionales organizan sus neuronas en capas de 3 dimensiones normalmente coincidiendo con las características de la imagen, siendo height la altura de la foto, width el ancho y depth los colores de la imagen. Por ejemplo se podría aplicar a imágenes de 32x32x3.

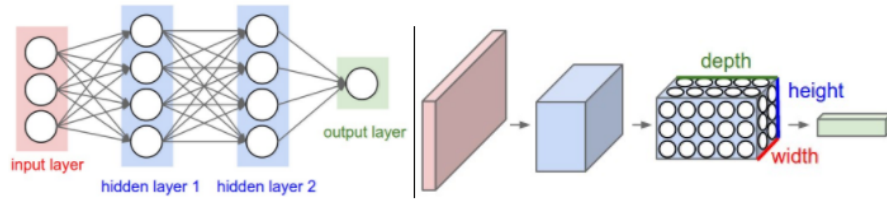


Figura 3.6: Izquierda: grafo de una RNA. Derecha: Grafo de una Red neuronal convolutiva.

## Tipos de capas

En este apartado se van a explicar los tres tipos de capas más importantes a la hora de elaborar las redes neuronales convolucionales:

### Capas convolucionales

En estas capas los píxeles de salida son combinaciones lineales de diferentes píxeles de entrada lo que generan nuevos mapas de píxeles más sencillos y que no son tan sensibles a cambios en las entradas. [12] Las convoluciones se pueden producir teniendo en cuenta solo 2 dimensiones (el alto y ancho) o teniendo en cuenta las 3 dimensiones (alto, ancho y profundidad). En algunas arquitecturas son llamadas capas RELU, ejecutan la función de activación de las neuronas.

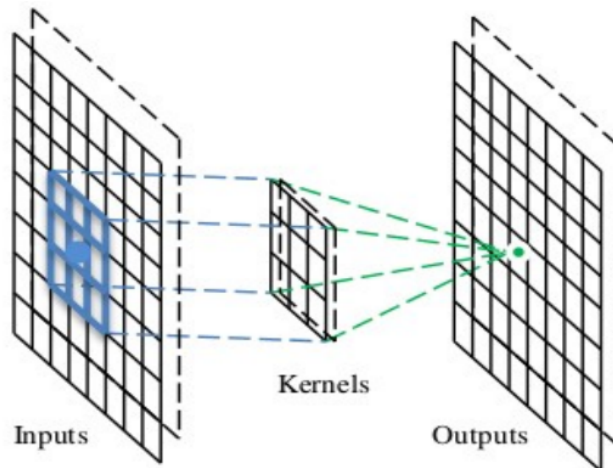


Figura 3.7: Figura que muestra una convolución en dos dimensiones.

Como vemos en la figura 3.7 Los kernels definen el área sobre el que se va a aplicar la convolucion, en el ejemplo 3x3.

### Capas de pooling

Las capas de pooling o reescalado tiene el objetivo de disminuir el número de parámetros o píxeles de los mapas, para ello se aplican técnicas como la explicada en la siguiente figura.

Estas capas van a tener un gran peso en los modelos de Mobilenet e Inception, ya que nos van a permitir ajustar el número de operaciones realizadas por los modelos para que se puedan ajustar correctamente a las limitaciones hardware de los teléfonos móviles.

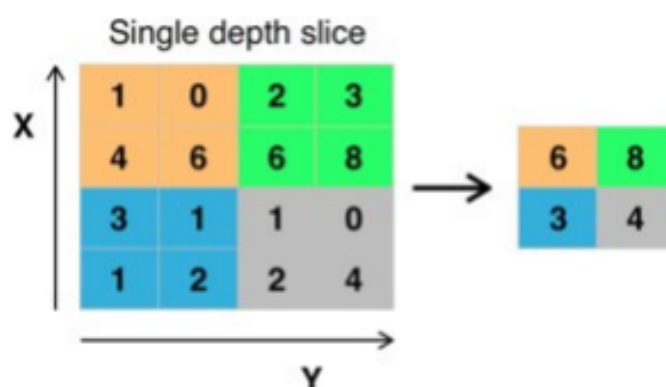


Figura 3.8: Técnica de máx pooling.

En la figura 3.8 se puede observar que los diferentes píxeles se agrupan eligiendo sólo aquellos de mayor valor para el nuevo mapa.

### Capas totalmente conectadas

En este tipo de capas las neuronas están totalmente conectadas con las neuronas de la capa anterior y tienen una profundidad igual al número de clases que se quieran clasificar. Por ejemplo si tenemos 10 clases diferentes, el último layer podría ser de tamaño 32x32x10. Estas capas son las encargadas de producir la señal de salida y clasificar la imagen eligiendo una de las clases.

Cuando re entrenamos un modelo, normalmente lo que estamos haciendo es modificar estas últimas capas que se encargan de clasificar las clases en vez de tener que re entrenar todas las demás capas que forman el modelo.

### 3.7. Modelo Inception

El modelo Inception es una red neuronal convolucional que se usa para la clasificación de imágenes. Inception ha sido desarrollado por Google y parte de la pregunta de que tipo de convolución aplicar en cada capa o filtro del modelo. Es decir, preguntarse que tamaño de convolución sería el adecuado para cada etapa, ¿3x3?, ¿1x1?, ¿5x5?.[2]

Ante esta pregunta, la respuesta que han aplicado es la de ejecutar los diferentes tamaños de convoluciones en cada capa de manera paralela y concatenar los diferentes mapas en uno solo que es el que se pasa a la siguiente capa del modelo. La idea es ejecutar todos los tipos de convoluciones a la vez y elegir el que mejor funcione.

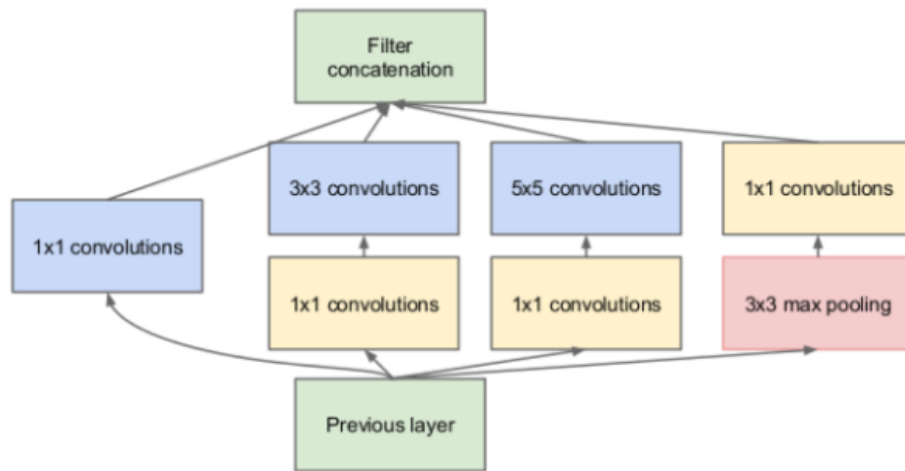


Figura 3.9: Ejemplo de un nodo Inception.

En la figura 3.9 podemos observar como se aplican las diferentes convoluciones en paralelo. Además podemos observar que se aplican unas convoluciones de 1x1, estas convoluciones tienen el objetivo de reducir el número de parámetros que se usan ya que aunque no reducen el área de los mapas sí reducen la profundidad de las capas.[13]



Los modelos inception se construyen concatenando estos bloques o nodos Inception, en teoría cuanto más profundidad se dé a estos modelos y más se concatenen más precisos deberían ser los modelos pero esto tiene un coste alto de computación (Aún reduciendo los computos con las convoluciones 1x1) y se corre el riesgo de sobreajustar los modelos.

Actualmente se pueden entrenar diferentes versiones de Inception, en el proyecto hemos trabajado reentrenando la versión 3 del modelo mediante Tensorflow.

Este modelo se sigue desarrollando y los desarrolladores lo van evolucionando por lo que no es de extrañar que en un futuro cercano aparezcan nuevas versiones mas optimizadas de este modelo.

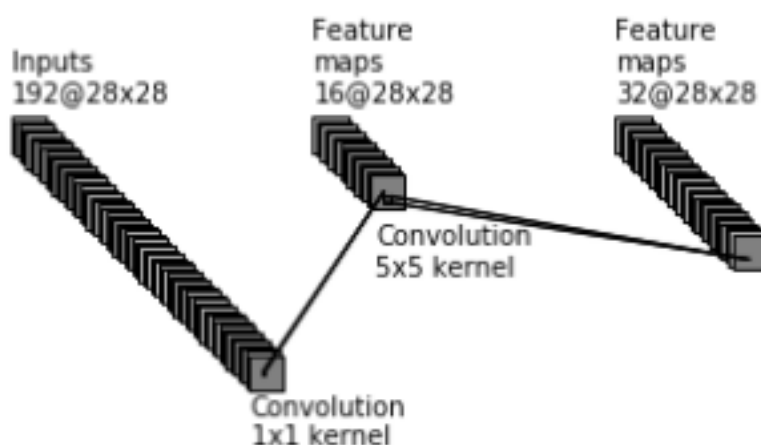


Figura 3.10: Aplicación de una convolucion 1x1 antes de ejecutar la convolucion 5x5.

En la figura 3.10 se puede observar el efecto de aplicar una convolucion 1x1 antes de efectuar la convolucion 5x5.

## 3.8. Modelo Mobilenet

Mobilenet hace referencia a una serie de modelos clasificadores de imágenes desarrollados por Google para ser embebidos en dispositivos móviles o sistemas con pocos recursos. La filosofía de este tipo de clasificadores nace

en contraposición a la tendencia que se estaba implantando en las redes neuronales de crear redes profundas y complejas con el objetivo de mejorar la precisión dejando a un lado cuestiones como el tamaño del modelo o la rapidez.[9]

Los modelos Mobilenet intentan ser precisos pero a la vez ser modelos de poco tamaño y rápidos que se adecuen a sistemas que necesiten de estas ventajas, como pueden ser los automóviles autónomos, realidad aumentada, sistemas de reconocimiento, entre otros.

La arquitectura de este tipo de modelo, al igual que el modelo Inception explicado anteriormente, se basa en una red neuronal convolucional que se explicará en el siguiente apartado.

## Arquitectura Modelo Mobilenet

La arquitectura del modelo Mobilenet se basa en dividir los filtros convolucionales típicos de una RNA totalmente conectada en dos nuevos tipos de filtros, los filtros convolucionales profundos (del inglés Depthwise Convolutional Filters) y los filtros convolucionales puntuales (del inglés Pointwise Convolution Filters).

El objetivo de esta división de filtros es el de minimizar el número de operaciones realizada. La convoluciones normales tienen el siguiente coste 3.2:

$$DK * DK * M * N * DF * DF \quad (3.2)$$

DK=Tamaño del kernel (alto x ancho), M=Número de canales de entrada, N=número de canales de salida, DF=Tamaño del mapa de salida.

Los Depthwise Convolutional Filters tienen el siguiente coste computacional 3.3:

$$DK * DK * M * DF * DF \quad (3.3)$$

Y los Pointwise Convolution Filters tienen el siguiente coste 3.4;

$$M * N * DF * DF \quad (3.4)$$

Si expresamos la convolución como un proceso de dos pasos de filtrado y combinación de los dos filtros anteriores obtenemos la siguiente reducción

de costes de computación 3.5:

$$\frac{(DK * DK * M * DF * DF + M * N * DF * DF)}{(DK * DK * M * N * DF * DF)} = (1/N) + (1/(DK * DK)) \quad (3.5)$$

Lo que supone una reducción respecto a los filtros convolucionales normales. Esta reducción la podemos observar en la siguiente tabla:

| Modelos        | Imagenet Accuracy | Million Mult-Adds | Million Parameters |
|----------------|-------------------|-------------------|--------------------|
| Conv MobileNet | 71.7 %            | 4866              | 29.3               |
| MobileNet      | 70.6 %            | 569               | 4.2                |

Tabla 3.1: Depthwise Separable vs Full Convolution MobileNet

Estos dos tipos de filtros se aplican de forma separada y paralela como sucedía en el modelo Inception. Para ver como surgen estas convoluciones vamos a fijarnos en la siguiente figura 3.11.

El uso de estos tipos de filtros de forma paralela, más el uso de filtros de normalización y pooling, es lo que usa Mobilenet para ejecutarse de forma rápida y utilizando la menor cantidad de memoria posible.

### 3.9. Data Augmentation

El Data Augmentation es una técnica que busca ampliar el conjunto de datos de entrenamiento, aplicando una serie de transformaciones sobre las imágenes iniciales. [10]

En las imágenes de entrenamiento se aplican una serie de técnicas como son la rotación, recortes, incremento del brillo, cambio de la gama de colores entre otras, para conseguir nuevas imágenes que usar para el entrenamiento del clasificador.

Esta técnica nos permite ampliar nuestro conjunto de datos sin tener almacenadas estas imágenes extras a cambio de un procesamiento extra a la hora de entrenar.

Para evitar el sobreajuste del modelo, ya que estas imágenes creadas están muy interrelacionadas entre si, se ejecutan las transformaciones con una probabilidad, disminuyendo así el número de imágenes artificiales.

### 3.10. Web Semántica

La Web semántica es una Web extendida desarrollada bajo los estándares W3C (World Wide Web Consortium). Esta Web extendida aplica metadatos a la información que recoge para que los usuarios puedan encontrar respuestas de una manera más rápida y sencilla gracias a esta información adicional.[17]

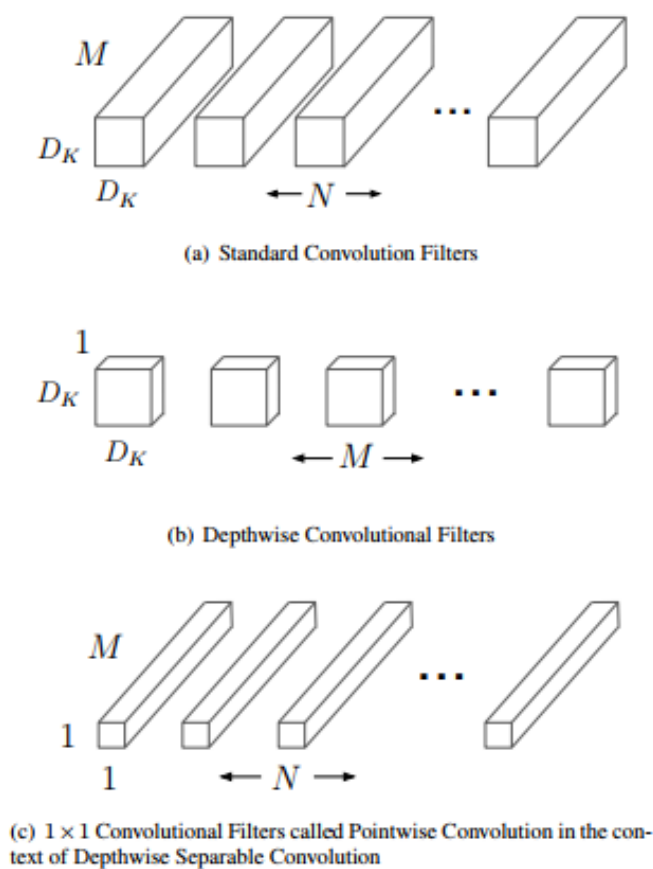


Figura 3.11: Los filtros convolucionales normales son divididos en los filtros profundos y los puntuales.

Estos metadatos son los que otorgan más semántica a la Web y permiten obtener soluciones a problemas habituales gracias al uso de una arquitectura común para toda la información.

Por ejemplo, en este proyecto, esta tecnología ha permitido automatizar todo el proceso de recogida de información de las diferentes especies de setas de una manera rápida y eficaz.

### ¿Para qué sirve?

A la Web actual que todos conocemos le podemos atribuir dos problemas que han surgido del enorme crecimiento que ha tenido. Estos problemas son

una sobrecarga de información, mucha de la información esta repetida en diferentes lugares siendo la misma, y que la información no se muestra bajo una estructura común, sino de una forma heterogénea.

La web semántica busca solucionar estos problemas delegándolos en el software, este software es capaz de procesar su contenido, combinarlo y realizar deducciones lógicas para solucionar problemas habituales.

## ¿Cómo funciona?

El funcionamiento de la web semántica se basa esencialmente en RDF, SPARQL y OWL, mecanismos que permiten convertir la estructura de la Web para que se adecue al funcionamiento de una Web Semántica.

- RDF: RDF permite introducir metadatos descriptivos en los distintos tipos de información.
- SPARQL: Es el lenguaje de consultas que nos permite realizar búsquedas sobre la información que contiene estos metadatos.
- OWL: OWL permite crear vocabularios para asociar los diferentes recursos.

El uso conjunto de estos mecanismos es el que nos permite razonar sobre los datos de la Web.

## 3.11. Web Scraping

La herramienta del Web Scraping es una herramienta para extraer información o datos de páginas web de una manera rápida, eficiente y automatizada. La información extraída se presenta de una forma más estructurada y fácil de usar.<sup>[1]</sup>

Los programas para realizar Web Scraping se pueden programar en diferentes lenguajes de programación, los más populares son Java, Python, Ruby y Node. Además existen diversos programas para realizar estas acciones mediante un interfaz gráfico.

Aproximadamente el 70 % de la información publicada en Internet esta en formato PDF, un formato no estructurado y difícil de manejar. Sin embargo, las páginas web si que tienen un formato estructurado ya que están

programadas en código HTML, pero aun así no están presentadas de una manera totalmente reutilizable ya que no todas siguen el mismo esquema.

La técnica más extendida de Web Scraping es analizar la estructura del código HTML propio de una página web para poder extraer la información buscada. Por ejemplo, extraer todos los atributos text de los label <p>que se encuentren en una determinada página Web.

## 3.12. Clave Dicotómica

Las claves dicotómicas son herramientas que permiten identificar organismos. Las claves pueden determinar animales, plantas, hongos, moneras, protistas o cualquier otro ser vivo. Las claves pueden alcanzar diferentes profundidades identificando el género, especie, familia, entre otros.

La clave dicotómica se basa en ir mostrando al usuario diferentes dilemas (afirmaciones contrapuestas) entre los que deberá ir eligiendo el que más se adecue al organismo a identificar hasta alcanzar la categoría taxonómica<sup>7</sup> deseada. Estas afirmaciones están enumeradas a lo largo de la clave.

### ¿Cómo usar una clave?

Usar una clave consiste simplemente en leer los diferentes dilemas y optar solamente por uno de ellos. El dilema elegido no volverá a aparecer en el desarrollo de la clave. [23]

---

<sup>7</sup>Grupos en los que se clasifican los seres vivos.





---

## Técnicas y herramientas

---

### 4.1. Java

Java es un lenguaje de programación desarrollado por James Gosling de Sun Microsystems y publicado en 1995.

Es un lenguaje de programación orientado a objetos, de propósito general y orientado a objetos que fue diseñado con el objetivo de tener la menor cantidad de dependencias de implementación posibles. Un programa java es un programa multiplataforma que solo necesita ser compilado una vez para ser ejecutado en diferentes máquinas. Esto se consigue ya que el programa se ejecuta sobre una máquina virtual java (JVM), elemento que debe estar disponible en el sistema sobre el que se va a ejecutar el programa.

Java cuenta con una gran cantidad de documentación en línea así como una gran variedad de librerías de código abierto creadas por la comunidad que extienden el uso de Java a muchas aplicaciones diferentes, mejorando la funcionalidad original de Java. En este proyecto por ejemplo nos ha servido para realizar la parte de Web Scraping y de la Web Semántica.

Url de la herramienta:

[https://www.java.com/es/download/faq/what\\_is\\_java.xml](https://www.java.com/es/download/faq/what_is_java.xml)

### 4.2. Python

Python es un lenguaje de programación interpretado, tipado dinámicamente y multiplataforma. Soporta programación orientada a objetos, programación imperativa y en menor medida, programación funcional. Su filosofía hace hincapié en una sintaxis que favorezca un código legible.[32]

Url de la herramienta: [<https://www.python.org/>](https://www.python.org/)

### 4.3. Eclipse

Eclipse es un entorno de desarrollo integrado (IDE, Integrated Development Environment) disponible para varios lenguajes de programación aunque el de uso más extendido sea el de Java.

En este proyecto se ha trabajado con el IDE de Java denominado Java EE (Eclipse IDE for Java EE Developers) en su versión "Neon.3".

Url de la herramienta: [<https://www.eclipse.org/ide/>](https://www.eclipse.org/ide/)

### 4.4. Apache Maven

Apache Maven es una herramienta para manejar e interpretar nuestros proyectos software mediante un archivo de configuración XML. Maven se basa en el concepto de *project object model (POM)* para describir como y en que orden se van a construir los elementos de un proyecto y para manejar las dependencias de ese proyecto.

Apache Maven permite manejar estas dependencias por Internet descargando y actualizando los diferentes repositorios.[19]

Url de la herramienta: [<https://maven.apache.org/>](https://maven.apache.org/)

### 4.5. Android

Android es un sistema operativo basado en Linux. Fue diseñado con el objetivo de operar en dispositivos móviles con pantalla táctil, empezando por los teléfonos inteligentes y tabletas y después extendiéndose a nuevos dispositivos como relojes inteligentes, televisores y automóviles. Fue desarrollado por Android Inc., empresa respaldada económicamente por Google que en 2005 la compró. Android fue lanzado en 2007 junto a la fundación Open Handset Alliance, que es un consorcio de compañías hardware, software y de telecomunicaciones.[18]

Los principales componentes del sistema operativo Android son los siguientes:

- Aplicaciones: Todas están escritas en el lenguaje de programación Java.

- Marco de trabajo de aplicaciones: Todos los desarrolladores tienen acceso al mismo entorno de trabajo donde cualquier aplicación puede publicar sus capacidades y luego cualquier otra aplicación puede hacer uso de estas capacidades. Este mecanismo a su vez permite que los componentes sean reemplazados por el usuario.
- Bibliotecas: Android dispone de una serie de bibliotecas del sistema escritas en C/C++ cuyas características se ponen a disposición de los desarrolladores a través del marco de trabajo.
- *Runtime* de Android: Android proporciona un set de librerías que proporcionan la funcionalidad base de Java. Cada aplicación se ejecuta en una máquina virtual ART.
- Núcleo Linux: Android depende de Linux para los servicios base del sistema.

Url de la herramienta: <https://developer.android.com/index.html?hl=es-419>

## 4.6. Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial de Android. Tiene el objetivo de facilitar el desarrollo en Android proporcionando herramientas personalizadas a los usuarios así como herramientas completas de edición, depuración, pruebas y perfilamiento de código.

Url de la herramienta: <https://developer.android.com/studio/index.html?hl=es-419>

## 4.7. JavaScript

JavaScript es un lenguaje de programación orientado a objetos, basado en prototipos, débilmente tipado y dinámico. Puede ser aplicado sobre un documento HTML con el fin de crear interactividad dinámica con las páginas Web.

JavaScript se desarrolló inicialmente con una sintaxis parecida a C y adopta nombres y convenciones del lenguaje de programación Java.[27]

Url de la herramienta: <https://www.javascript.com/>

## 4.8. SLF4J

SLF4J (The Simple Logging Facade for Java) proporciona una API de registro a Java mediante un patrón de fachada que se aplica a varios marcos de registro (*logging frameworks*) que permite al usuario mostrar el marco de registro deseado en el momento de despliegue de la aplicación.

SLF4J nos permite filtrar los mensajes de log para visualizar de una forma más oportuna la información que nos muestra Java.

Url de la herramienta: [<https://www.slf4j.org/>](https://www.slf4j.org/)

## 4.9. RDF

El Resource Description Framework (RDF) es una definición de tipo de documento de XML, es decir, una series de metadatos que usan XML para poder relacionar contenidos entre las diferentes páginas web.

Los objetos de información o recursos se describen a través de un conjunto de propiedades denominadas "descripción RDF"(<rdf:description>). Estas propiedades se pueden entender como atributos de los recursos correspondiéndose a los pares atributo-valor tradicionales.

En el modelo de datos de RDF podemos distinguir tres tipos de objetos:

- Recursos: Cualquier recurso Web identificado unívocamente por un URI.
- Propiedades: Son características o atributos usadas para describir los recursos. En ellas se definen los valores permitidos y sobre que recursos se pueden aplicar.
- Descripciones: Son una triada formada por un recurso, un nombre de propiedad y un valor para esa propiedad. (Sujeto, predicado y objeto). Ver siguiente figura [4.12](#)



Figura 4.12: Descripciones RDF.

RDF usa la sintaxis básica de XML1.0. Ejemplo:

```
«xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax»"
```

El modelo de datos y la sintaxis no son necesarios para definir la relación entre los diferentes recursos y predicados, por ello hace falta un esquema. Un esquema son una serie de informaciones relativas a la clase a la que pertenecen los recursos para establecer relaciones jerárquicas entre ellos.

## 4.10. Apache Jena

Apache Jena es un entorno de desarrollo de código abierto para Java cuyo objetivo es el de permitir construir aplicaciones que hagan uso de la Web semántica y del *Linked Data*<sup>8</sup>.

El framework hace uso de diferentes APIs para extraer y escribir información en grafos RDF<sup>9</sup>.[\[28\]](#)

La arquitectura de Apache Jena contempla las siguientes características:

- Arquitectura para manejar (leer, procesar, escribir) las ontologías RDF y OWL.
- Motor para razonar sobre estas ontologías RDF y OWL.

---

<sup>8</sup>Método por el cuál se publican datos estructurados para que puedan ser interconectados con facilidad.

<sup>9</sup>Modelo de datos para los metadatos basado en triadas de información

- API para almacenar las tripletas RDF en ficheros o en memoria.
- Motor de queries para realizar consultas bajo la especificación SPARQL.[8]

En la siguiente figura se muestra un esquema de la arquitectura de Apache Jena.

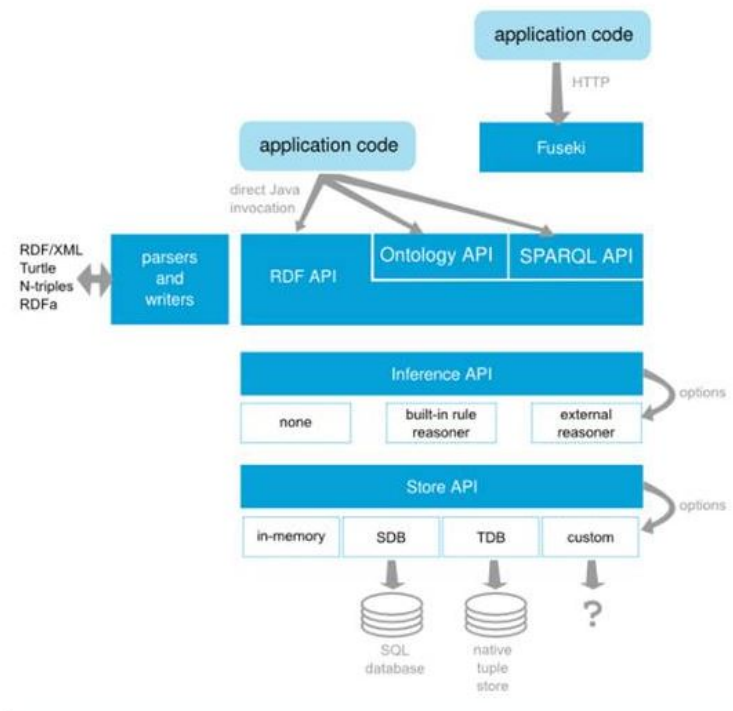


Figura 4.13: Arquitectura de Apache Jena

Url de la herramienta: [<https://jena.apache.org/>](https://jena.apache.org/)

## 4.11. JSON

JSON (acrónimo de JavaScript Object Notation) es un formato de texto ligero para el intercambio de datos. Es un subconjunto de la notación de JavaScript pero debido a la extensión de su uso como alternativa a XML se considera como un formato de lenguaje independiente.

La principal ventaja que ofrece JSON es la facilidad para crear analizadores sintácticos (parsers). [29]

Url de la herramienta: [<https://www.json.org/json-es.html>](https://www.json.org/json-es.html)

## 4.12. Jaunt

Jaunt es una librería gratuita de Java desarrollada para tareas de Web Scraping y automatización Web así como la realización de consultas JSON. Con Jaunt los programas Java pueden operar a nivel de navegador, documento o operaciones DOM (del inglés Document Object Model).

JSON esta diseñado para realizar las siguientes tareas:

- Completar y enviar formularios Web.
- Crear programas automáticos Web o programas de Web-Scraping.
- Escribir clientes http para REST-APIS o Web-apps (JSON,HTML, XHTML o XML).

Url de la herramienta: [<http://jaunt-api.com/>](http://jaunt-api.com/)

## 4.13. SQLite

SQLite es un sistema de gestión de bases de datos relacional que contempla las características ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad/Persistencia). SQLite esta contenida en una pequeña librería escrita en C que se enlaza con el propio programa pasando a ser parte del mismo, a diferencia de otros sistemas gestores de bases de datos que se ejecutan en un proceso separado e independiente al programa.

SQLite destaca por crear bases de datos ligeras que se almacenan en un único fichero, en la propia máquina donde se ejecuta. Además reduce la latencia ya que los accesos a la base de datos se realizan mediante subrutinas

o funciones que implementan la funcionalidad SQL, en vez de realizar comunicación entre procesos como ocurre con otros gestores.[34]

Url de la herramienta: [<https://www.sqlite.org/>](https://www.sqlite.org/)

## 4.14. SPARQL

SPARQL (del inglés Protocol and RDF Query Language) es un lenguaje de consultas para grafos RDF. SPARQL permite crear consultas sobre diferentes fuentes que contengan la información nativamente en formato RDF o que la muestren en este formato mediante algún middleware.[11]

SPARQL soporta una serie de capacidades sobre los grafos como las uniones, intersecciones, agregaciones, sub-consultas entre otros. Los resultados obtenidos por estas consultas pueden ser o nuevos grafos RDF o *Result sets*.

Url de la herramienta: [<https://www.w3.org/TR/rdf-sparql-query/>](https://www.w3.org/TR/rdf-sparql-query/)

## 4.15. DBpedia

La DBpedia es un proyecto comunitario que tiene como objetivo poder extraer de forma estructurada la información contenida en la Wikipedia y crear una Web semántica con esta información. Este proyecto está siendo realizado por la Universidad de Leipzig, Universidad Libre de Berlín y la compañía OpenLink Software. [24]

La DBpedia se fundamenta en la estructura RDF y las consultas en SPARQL para lograr estos objetivos.

Url de la herramienta: [<http://wiki.dbpedia.org/>](http://wiki.dbpedia.org/)

## 4.16. Tensorflow

Tensorflow es una biblioteca de código abierto para realizar cálculos numéricos usando grafos de flujo de datos. Los nodos de los grafos representan operaciones matemáticas mientras que los enlaces representan conjuntos de datos multidimensionales que relacionan los nodos. Estos arrays son referidos como "Tensores".[35]

La arquitectura flexible de Tensorflow permite ejecutar los cálculos en una o varias CPUs o GPUs, ya sea en servidores, ordenadores de sobremesa o dispositivos móviles mediante una sola API.



Tensorflow ha sido desarrollado para crear y entrenar diferentes modelos de redes neuronales.

TensorFlow fue desarrollado originalmente por investigadores e ingenieros que trabajan en el equipo Brain de Google dentro de la organización de investigación de Inteligencia Artificial de Google para realizar investigaciones de aprendizaje automático y redes neuronales profundas.

Actualmente el sistema es lo suficientemente general para aplicarse a otra serie de dominios.

Url de la herramienta: [<https://www.tensorflow.org/>](https://www.tensorflow.org/)

## 4.17. Git

Git es un sistema de control de versiones distribuido de código abierto. El control de versiones nos permitirá retornar a algún punto anterior del desarrollo de nuestra aplicación en caso de sufrir errores o pérdidas de información y nos permitirá llevar un registro de cuando y por quién se modifican los archivos en todo momento.

Actualmente Git es uno de los sistemas más utilizados para este propósito y nos proporciona todas las herramientas que necesitamos para llevar el control de versiones de una manera simple y eficiente.

Url de la herramienta: [<https://git-scm.com/>](https://git-scm.com/)

## 4.18. GitHub

Github es un sistema de código abierto que nos permite alojar nuestro repositorio central del proyecto usando el control de versiones Git. Además permite la integración con diferentes herramientas como Zenhub, que se explicará a continuación, y nos ofrece las características que necesitamos para llevar a cabo esta tarea.

Url de la herramienta: [<https://github.com/>](https://github.com/)

## 4.19. Zenhub

Zenhub es una herramienta de gestión de proyectos integrada nativamente con GitHub, lo que hará que no necesitemos configuraciones extra a la hora de integrarla en nuestro repositorio.

Esta herramienta nos ayuda a la hora de planificar nuestros proyectos mediante un tablero Kanban integrado y una serie de diagramas (Burndown y Velocity tracking) que se corresponden con las historias de usuarios y tareas que hayamos definido en nuestro proyecto.

Con el podremos definir puntos de historia en nuestras tareas para poder planificar de una manera más eficaz nuestros sprints.

Url de la herramienta: [<https://github.com/marketplace/zenhub>](https://github.com/marketplace/zenhub)

## 4.20. SourceTree

SourceTree es una herramienta que nos proporciona una interfaz gráfica para poder realizar las tareas relacionadas con el control de versiones de nuestro repositorio, como pueden ser realizar operaciones de commits, pull, push, Fetch, Branch, Merge entre otras.

SourceTree nos permite ver de una manera fácil y cómoda las ramas de nuestro proyecto, así como todos los commits realizados y los cambios producidos en cada archivo modificado.

Url de la herramienta: [<https://es.atlassian.com/software/sourcetree>](https://es.atlassian.com/software/sourcetree)

## 4.21. Latex

Latex es un sistema de composición de textos de código abierto, orientado a crear documentos con una alta calidad tipográfica, como pueden ser artículos o libros científicos.[30]

Url de la herramienta: [<https://www.latex-project.org/>](https://www.latex-project.org/)

---

# Aspectos relevantes del desarrollo del proyecto

---

En este apartado se va a recoger el ciclo de vida del proyecto, detallando los aspectos más relevantes que se han tratado y como se han resuelto las diferentes dificultades encontradas a lo largo de su desarrollo.

Se irán presentando diferentes secciones que concuerdan con el orden cronológico seguido en el proyecto y muestran la justificación de las decisiones tomadas.

## 5.1. Propuesta del proyecto

La propuesta de este proyecto consistía en crear una aplicación Android para el reconocimiento de setas que se dividía en las siguientes 3 tareas principales:

- Clasificador de imágenes: La tarea principal pedida para realizar este proyecto era la de construir un clasificador visual de imágenes que a través de la foto realizada a una seta nos devolviera un listado de las especies más probables.
- Web Semántica: Conseguir información a través de una web semántica para documentar las especies de setas incluidas en el clasificador.
- Clave dicotómica: Incorporar una clave dicotómica que reforzara la tarea del clasificador para el reconocimiento de la especie.

Ante estas tareas se empezó a investigar que alternativas había para construir una aplicación que fuera lo suficientemente precisa y de fácil uso para el usuario. Se barajaron las siguientes posibilidades:

- Incorporar sólo un clasificador visual: Si sólo se implementaba un clasificador visual en la aplicación, esta sería muy fácil de usar pero sería poco precisa ya que clasificar la especie de una seta por una única foto es una tarea casi imposible.
- Clasificador visual con elección manual entre las candidatas: La idea de esta propuesta es la de mostrar al usuario un listado con las cinco especies más probables clasificadas para esa foto y que este las pueda comparar mediante fotografías proporcionadas por la aplicación de esas especies con su foto. Esta propuesta puede ser un poco más fiable pero depende de los conocimientos del usuario y la hace más compleja.
- Clave dicotómica única: Incluir una clave dicotómica aislada del clasificador podía proporcionar una gran fiabilidad pero suelen ser claves de difícil uso para el usuario.
- Clave dicotómica más el clasificador: Esta propuesta consistía en filtrar las preguntas de la clave dicotómica en base a los resultados obtenidos por el clasificador de imágenes.

Con estas propuestas en mente se empezó a estudiar como se podía implementar el clasificador de imágenes y como podíamos implementar las diferentes propuestas en base a lo que se iba desarrollando.

## **5.2. ¿Cómo implementar el clasificador?**

Se decidió empezar por la tarea de generar el clasificador de imágenes ya que era la tarea más complicada de realizar y de la que dependían las demás tareas. En la asignatura de minería de datos ya había adquirido conocimientos sobre como entrenar y usar clasificadores de imágenes pero ahora surgía la duda de como hacer esto mismo pero en una plataforma nueva para mí que era Android. En las primeras reuniones del proyecto los profesores me propusieron las siguientes posibilidades para realizarlo:

- Entrenar un clasificador de imágenes desde cero que se ejecutara en un servidor Web y mostrara los resultados en el teléfono móvil.

- Seguir la propuesta anterior pero reentrenando una red neuronal en vez de entrenar un modelo desde cero.
- Reentrenar los nuevos modelos Mobilenet mediante Tensorflow, lo que nos permitiría ejecutar los clasificadores en el propio dispositivo móvil.

Elegí empezar por estudiar la tercera propuesta y ver si era posible ejecutar los modelos en el propio teléfono móvil. Esta propuesta tiene la ventaja de que el usuario no necesita estar conectado a un servidor Web, característica importante si pensamos que esta aplicación se usara en zonas con poca cobertura.

La preocupación principal era saber si los modelos se iban a poder ejecutar en el propio móvil de una manera rápida, ya que es un proceso costoso, y que nos proporcionara los resultados deseados. Después de estudiar la tecnología Tensorflow y cómo implementarla en una aplicación Android probé varios modelos preentrenados de prueba que proporcionan en los ejemplos de Tensorflow<sup>10</sup> y vi que la aplicación se ejecutaba correctamente en diferentes modelos de móvil simulados y en mi propio teléfono.

Esta etapa del proyecto me permitió empezar a familiarizarme con el entorno de programación Android Studio y con las librerías de Tensorflow.

Después de realizar pruebas con modelos preentrenados y ejemplos empecé a estudiar como poder entrenar mi propio modelo de Mobilenet o Inception, con nuestro repositorio de imágenes de setas, para que diferenciara entre las diferentes especies. La solución la encontré en el mismo repositorio de Github<sup>11</sup> de Tensorflow en el que proporcionan un script en python (-etrain.py) para reentrenar ambos modelos pudiendo modificar gran variedad de parámetros y ajustes para crear un modelo personalizado.

Gracias a estos scripts y a los ejemplos encontrados de Tensorflow en Android conseguí reentrenar los primeros modelos y clasificar las primeras imágenes en el propio móvil.

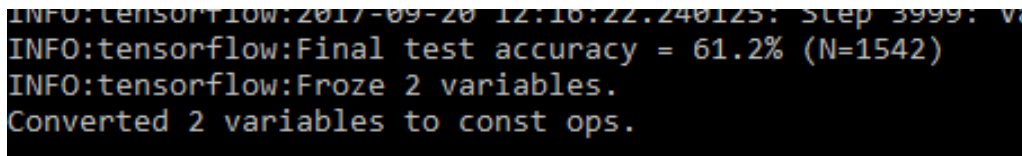
Las imágenes usadas para entrenar los modelos se sacaron de los repositorios Imagenet y Encyclopedia of Life, recopilando aproximadamente entre 30 y 80 fotos por cada especie de seta. En total se han usado 9400 fotografías para reentrenar el clasificador completo divididas en 172 especies de setas.

---

<sup>10</sup><https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android>

<sup>11</sup>[https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/image\\_retraining](https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/image_retraining)

Las primeras pruebas se ejecutaron con un modelo Mobilenet v1-224 reentrenado, que diferenciaba 118 especies de setas, en un teléfono móvil Samsung galaxy S5. La aplicación lograba ejecutarse fluidamente y se consiguió una precisión (61,5 %) 5.14 en la validación cruzada con 4000 pasos de reentrenamiento:



```
INFO:tensorflow:2017-09-20 12:16:22.240125: Step 3999: va
INFO:tensorflow:Final test accuracy = 61.2% (N=1542)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
```

Figura 5.14: Mobilenet v1-224, 4000 steps, 118 especies.

Aunque la precisión de estos modelos no sea relativamente alta, nos basta con que la especie correcta se encuentre entre los cinco primeros resultados del clasificador, ya que este clasificador se complementará con la clave dicotómica.

### 5.3. ¿Qué modelo de clasificador usar?

Para elegir el modelo a usar se tuvieron que tener en cuenta diferentes factores como el peso del modelo, su carga de computo y su precisión. En teoría el modelo Inception es más preciso que el Mobilenet pero requiere mayor capacidad de computo y espacio de almacenamiento, recursos que son limitados en los teléfonos móviles.

Para Mobilenet contábamos con los modelos mostrados en la figura 5.15 mientras que de Inception solo podía reentrenar el modelo Inception v3.

| Model Checkpoint      | Million MACs | Million Parameters | Top-1 Accuracy | Top-5 Accuracy |
|-----------------------|--------------|--------------------|----------------|----------------|
| MobileNet_v1_1.0_224  | 569          | 4.24               | 70.7           | 89.5           |
| MobileNet_v1_1.0_192  | 418          | 4.24               | 69.3           | 88.9           |
| MobileNet_v1_1.0_160  | 291          | 4.24               | 67.2           | 87.5           |
| MobileNet_v1_1.0_128  | 186          | 4.24               | 64.1           | 85.3           |
| MobileNet_v1_0.75_224 | 317          | 2.59               | 68.4           | 88.2           |
| MobileNet_v1_0.75_192 | 233          | 2.59               | 67.4           | 87.3           |
| MobileNet_v1_0.75_160 | 162          | 2.59               | 65.2           | 86.1           |
| MobileNet_v1_0.75_128 | 104          | 2.59               | 61.8           | 83.6           |
| MobileNet_v1_0.50_224 | 150          | 1.34               | 64.0           | 85.4           |
| MobileNet_v1_0.50_192 | 110          | 1.34               | 62.1           | 84.0           |
| MobileNet_v1_0.50_160 | 77           | 1.34               | 59.9           | 82.5           |
| MobileNet_v1_0.50_128 | 49           | 1.34               | 56.2           | 79.6           |
| MobileNet_v1_0.25_224 | 41           | 0.47               | 50.6           | 75.0           |
| MobileNet_v1_0.25_192 | 34           | 0.47               | 49.0           | 73.6           |
| MobileNet_v1_0.25_160 | 21           | 0.47               | 46.0           | 70.7           |
| MobileNet_v1_0.25_128 | 14           | 0.47               | 41.3           | 66.2           |

Figura 5.15: Comparativa modelos Mobilenet

Se realizaron pruebas con los diferentes modelos y al final opté por usar el modelo Mobilenet v1 224 ya que se ejecutaba de forma suficientemente fluida y proporciona una precision superior a los demás modelos Mobilenet.

Respecto al modelo Inception su peso era de 75 Megabytes en contra de los 17 Megabytes que ocupaba el modelo Mobilenet y los resultados obtenidos 5.16 en validación cruzada eran parecidos entrenados en las mismas condiciones.

```
INFO:tensorflow:Final test accuracy = 55.6% (N=1542)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

(tensorflow) C:\Users\adrit\Desktop\image_retraining>
```

Figura 5.16: Inception v3, 4000 steps, 118 especies.

Una vez elegido el modelo se procedió a escalar el modelo aumentando el número de especies a diferenciar, pasando de clasificar 118 especies a un

total de 172. Este aumento provoco un ligero descenso en la precisión del clasificador que se solvento aplicando técnicas de Data augmentation a la hora de entrenar el modelo. La precisión obtenida final fue de un 59,4% **5.17** en validación cruzada de media entre todas las categorías de especies de setas.

Se probó a aumentar el número de pasos de entrenamiento pero esto no provocaba un ascenso de la precisión sino que se mantenía constante y provocaba el riesgo de que se sobre-entrenara el modelo por lo que se decidió dejar el parámetro en 4000 pasos.

El tiempo necesario para entrenar este modelos con un procesador i7 4790, fue de 5 horas 55 minutos.

```
INFO:tensorflow:2017-12-02 17:16:02.127500: Step 3990: Validation accuracy = 63.0% (N=100)
INFO:tensorflow:2017-12-02 17:16:51.365671: Step 3999: Train accuracy = 99.0%
INFO:tensorflow:2017-12-02 17:16:51.365671: Step 3999: Cross entropy = 0.117547
INFO:tensorflow:2017-12-02 17:16:51.443801: Step 3999: Validation accuracy = 48.0% (N=100)
INFO:tensorflow:Final test accuracy = 59.4% (N=2030)
```

Figura 5.17: Mobilenet v1-224, 4000 steps, 172 especies, con data augmentation.

## 5.4. Web Semántica

La siguiente tarea que se decidió afrontar después de implementar el clasificador de imágenes fue la parte de Web semántica. En esta tarea había que recopilar información de las diferentes especies de setas, de alguna Web semántica, con el objetivo de crear fichas para cada especie que pudiera consultar el usuario en cualquier momento.

La principal dificultad que se encontró en este apartado y que determinó el uso de la DBpedia como Web semántica, fue encontrar un repositorio que contuviera información suficiente de todas las especies sin excepción. El alto número de especies que podía clasificar la aplicación dificultaba esta tarea. Este requisito era necesario para automatizar la extracción de datos y que todas las especies tuvieran su ficha de información.

La DBpedia era la única que recopilaba información suficiente de todas las especies al completo y nos proporcionaba la información mínima deseada como era una descripción de la especie y la comestibilidad de esta.

Otra opción que se contempló fue la de usar *web scraping* en vez de consultas a una web semántica, pero esta opción se descarto ya que no se



logró encontrar un repositorio con todas la especies y que contuviera más información que la que proporcionaba por la DBpedia.

Gracias al lenguaje de consultas SparQL y la librería Apache Jena para Java se implemento un programa en Windows que realizaba consultas a la DBpedia y nos devolvía la información necesaria.

El siguiente problema a afrontar era como almacenar y transferir esta información a la aplicación Android. Después de estudiar diferentes bases de datos se opto por usar la base de datos SQLite. Esta base de datos tiene soporte tanto para Android como para Windows por lo que es fácil exportar los datos de una plataforma a otra. Además no requiere de configuración de usuarios ni conexiones lo que simplifica su uso. Aunque proporcione una funcionalidad básica, esta base de datos nos permitía almacenar nuestra información y cumplía con los requisitos de la aplicación.

Para mostrar imágenes comparativas de las especies se usaron una parte de las imágenes ya recopiladas, que se habían usado para entrenar el clasificador. Estas imágenes se formatearon en una resolución menor que la original para ahorrar espacio en la aplicación. Estas imágenes se almacenaron dentro de la propia estructura de la aplicación Android.

Una vez recopilada la información de las diferentes especies, la aplicación era capaz de clasificar las cinco especies más probables respecto a la imagen de la seta introducida, mostrar imágenes comparativas de las especies y mostrar información de cada especie.

De cara a la internacionalización de la aplicación se implementó un método que tradujera automáticamente mediante el traductor de Google la información de la DBpedia en caso de no estar disponible en el idioma deseado. Por ejemplo, había veces que la información no se encontraba disponible en Español por lo que se traducía desde la versión en inglés.

## 5.5. Clave dicotómica

El último apartado a tratar era el de implementar una clave dicotómica que nos permitiera diferenciar entre todos los géneros contenidos en el clasificador. Esta tarea tenia las dos siguientes dificultades:

- Conseguir una clave dicotómica que discriminara entre todos los géneros de especies del clasificador.

- Estructurar esa clave dicotómica de una forma que se pudiera manejar para implementarse en la aplicación Android.

Después de buscar en diferentes páginas web, se encontró una (<http://www.avelinasetas.info/claves.php>) que contenía una serie de claves dicotómicas para diferentes especies y una clave que discriminaba entre 108 géneros de setas. De las claves encontradas fue la que más géneros contemplaba. Para que los géneros de la clave y del clasificador coincidieran se recopilaron imágenes de las especies que no se contemplaban en el clasificador para reentrenarlo y que pudiera discriminar entre 172 especies distintas.

Esta solución tiene la limitación de que el número de setas discriminado por el clasificador no es el mismo que el de la clave dicotómica.

Para extraer las claves de la página Web se desarrollo un programa en Java que mediante técnicas de Web Scraping extrajera las claves dicotómicas y las almacenar en estructuras de datos que nos permitieran usarlas para nuestra aplicación Android.

Además de la clave general de géneros, se extrajeron las claves dicotómicas disponibles para diferenciar entre especies concretas dentro de un mismo género. Actualmente la aplicación dispone de 39 claves dicotómicas de géneros concretos para clasificar la especie concreta.

Esta tarea se pudo automatizar mediante la herramienta Jaunt para Java revisando el código html de la página Web.

## 5.6. Aspectos de diseño

Para diseñar la interfaz de la aplicación se siguieron las recomendaciones de Material Design <sup>12</sup>. Se elaboró un prototipado inicial de las diferentes actividades y a partir de ahí se fue elaborando la interfaz final integrando los diferentes módulos de la aplicación (clasificador de imágenes, muestra de información y las claves dicotómicas).

Se han intentado usar colores planos siguiendo las indicaciones de Material Design así como iconos intuitivos para los diferentes botones de la aplicación.

La aplicación está desarrollada para que se pueda visualizar en vertical como en horizontal. Además se han usado medidas relativas de los elementos para que la aplicación se pueda adaptar a diferentes tamaños de pantalla.

---

<sup>12</sup><https://material.io/guidelines/>

La aplicación cuenta con un menú lateral que nos permite navegar desde cualquier punto de la aplicación a las opciones principales.

Se ha traducido la aplicación al Inglés y Español traduciendo tanto textos extraídos de la DBpedia cómo las cuarenta claves dicotómicas implementadas. El idioma de la aplicación se puede cambiar en cualquier momento desde la opción del menú lateral.

Cada actividad de la aplicación cuenta con un menú de ayuda que explica la funcionalidad completa de la actividad en la que te encuentras. El menú lateral cuenta con su propia ayuda.

Respecto al tamaño de la aplicación, este se ha conseguido reducir a un total de 42,6 Megabytes. Para ello se tuvieron que formatear las imágenes de las 172 especies de setas (5 imágenes por especie) a una resolución menor, así como usar el modelo Mobilenet en vez del Inception.



---

## Trabajos relacionados

---

### 6.1. Otros proyectos

Este proyecto partía del trabajo de Máster *Reconocimiento de setas mediante visión artificial y sistema experto IberoSetas 2.0* realizado por Iñaki Arroyo Nebreda en el año 2014. Aunque el trabajo actual contemplaba objetivos parecidos, decidí empezar mi trabajo desde cero ya que se han usado tecnologías diferentes y se ha generado la aplicación desde una perspectiva diferente. [14]

Para la parte del clasificador, en el trabajo anterior se creo un clasificador de imágenes en un servidor remoto al que debía conectarse la aplicación móvil. Para ello se tuvieron que preprocesar las imágenes para extraer sus características mediante técnicas como el *Bag of words* y a partir de esta información entrenar un clasificador como pudieran ser el J48 o el KNN (*k-nearest neighbor*).

En nuestro caso se han usado las recientes versiones de redes neuronales de Mobilenet e Inception desarrolladas por Google. Estos modelos realizan la extracción de características y están diseñados para ejecutarse de manera eficiente en las arquitecturas de los teléfonos móviles.

Gracias a estos avances tecnológicos se han conseguido clasificar 171 especies respecto a las 10 que clasificaba el trabajo anterior, realizando la tarea en el propio dispositivo móvil sin necesidad de una conexión a Internet a un servido externo.

La parte de la clave dicotómica en el trabajo anterior se realizó codificando una clave de 10 géneros sobre un formato xml. En este trabajo se han realizado técnicas de Web Scraping que nos han permitido extraer diferentes

claves que cubren un mayor número de géneros y especies, codificandolas directamente en estructuras de datos Java.

La parte de web semántica se ha realizado de igual forma aplicando consultas sobre la DBpedia.

## 6.2. Aplicaciones relacionadas

Se ha encontrado una aplicación, que se lanzó a mediados de Octubre de 2017, en la que se puede clasificar setas a través de imágenes de manera similar a lo propuesto en este proyecto. La aplicación se puede encontrar en el siguiente link <https://play.google.com/store/apps/details?id=com.pingou.champignouf&hl=es>

La principal diferencia de esta aplicación es que necesitas conexión a Internet para clasificar la imagen de la seta, a diferencia de nuestra aplicación, que se clasifica en el propio móvil.

No se han encontrado más aplicaciones que realicen este tipo de clasificación de setas en los dispositivos móviles.

---

# Conclusiones y Líneas de trabajo futuras

---

## 7.1. Conclusiones

Respecto a los requisitos del proyecto, creo que se han cumplido ofreciendo un clasificador de especies que puede servir de ayuda en la práctica de la micología, de una manera rápida y accesible para los usuarios, apoyándose en las diferentes claves dicotómicas e información disponible de manera local.

A nivel personal, he adquirido nuevos conocimientos relacionados con la minería de datos, clasificación de imágenes, Web scraping, web semántica, planificación, uso de Latex y el desarrollo en Android, entre otros, permitiéndome manejarlos con mayor soltura en estos campos.

Respecto a las dificultades encontradas, algunas han surgido por intentar abarcar una gran cantidad de especies, lo que ha significado tener que encontrar un gran número de imágenes y de información de setas, además de dificultar la tarea de implementar una clave dicotómica que contuviese todos los géneros clasificados.

La escasez de imágenes de algunas especies se ha resuelto gracias a aplicar técnicas de *data augmentation* así como poder usar diferentes claves dicotómicas gracias a la aplicación de web scraping. El problema de automatizar la recolección de las claves dicotómicas es que no todas se presentaban de manera uniforme en la página web y este hecho ha dado problemas para poder recuperarlas, teniendo que suprimir algunas de estas claves disponibles. Aún así, considero que se han conseguido suficientes claves para las especies disponibles.

En general estoy satisfecho con el trabajo realizado y con las aplicaciones propuestas.

## **7.2. Líneas de trabajo futuras**

A continuación se muestra un listado de aplicaciones y mejoras que se podrían incorporar en futuros proyectos:

- Implementar un sistema de localización en el que el usuario pueda compartir donde ha encontrado una especie de seta en concreto.
- Implementar nuevas bases de datos que suministren información de las diferentes setas además de la DBpedia.
- Expandir el clasificador aumentando el número de especies o imágenes
- Implementar un sistema de aprendizaje en el que la aplicación pregunte al usuario sobre una seta, y éste tenga que adivinar la especie.
- Generar un sistema en el que el usuario se pueda registrar y llevar un listado de las setas que ha ido clasificando a lo largo de su uso con la aplicación.
- Almacenar la información recopilada por los usuarios en un servidor para usarla en otras aplicaciones como podría ser, indicar en que zonas hay determinadas especies y en que momento se han encontrado.
- Generar un mercado virtual en el que los usuarios puedan vender y comprar diferentes tipos de setas.



---

## Bibliografía

---

- [1] Osmar Castrillo-Fernández. Web scraping: Applications and tools, Diciembre, 2015. [Internet; descargado 12-noviembre-2017].
- [2] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.
- [3] conceptodefinicion.de. Definición de micología, 2015. [Internet; descargado 28-noviembre-2017].
- [4] cs231n. Convolutional neural networks (cnns / convnets). [Internet; descargado 30-noviembre-2017].
- [5] Universidad de Salamanca. Redes neuronales. [Internet; descargado 9-noviembre-2017].
- [6] dipualba. El reino de los hongos, 2010. [Internet; descargado 8-noviembre-2017].
- [7] Florencio. Las setas y sus características. [Internet; descargado 8-noviembre-2017].
- [8] Luis Miguel Gracia. Qué es apache jena, 27 julio 2012. [Internet; descargado 15-noviembre-2017].
- [9] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

- [10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [11] SPARQL 1.1 Query Language. Steve harris, garlik, a part of experian andy seaborne, the apache software foundation, 21 Marzo 2013. [Internet; descargado 30-noviembre-2017].
- [12] Patricio Loncomilla. Deep learning: Redes convolucionales, 2016. [Internet; descargado 30-noviembre-2017].
- [13] Tommy Mulc. Inception modules: explained and implemented, 25 Septiembre 2016. [Internet; descargado 30-noviembre-2017].
- [14] Iñaki Arroyo Nebreda. Reconocimiento de setas mediante visión artificial y sistema experto iberosetas, 2014. [Internet; descargado 13-diciembre-2017].
- [15] novemberteardrops. partesdelaseta, 23 noviembre, 2014. [Internet; descargado 9-enero-2018].
- [16] Sergio Ventura. El proceso en la minería de datos, 18 julio 2011. [Internet; descargado 11-noviembre-2017].
- [17] W3C. Guía breve de web semántica. [Internet; descargado 12-noviembre-2017].
- [18] Wikipedia. Android — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 13-noviembre-2017].
- [19] Wikipedia. Apache maven — wikipedia, the free encyclopedia, 2017. [Online; accessed 13-November-2017].
- [20] Wikipedia. Aprendizaje automático — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 10-noviembre-2017].
- [21] Wikipedia. Aprendizaje no supervisado — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 10-noviembre-2017].
- [22] Wikipedia. Aprendizaje supervisado — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 10-noviembre-2017].
- [23] Wikipedia. Clave dicotómica — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 12-noviembre-2017].

- [24] Wikipedia. Dbpedia — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 30-noviembre-2017].
- [25] Wikipedia. Fungi — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 8-noviembre-2017].
- [26] Wikipedia. Inteligencia artificial — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 10-noviembre-2017].
- [27] Wikipedia. Javascript — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 13-noviembre-2017].
- [28] Wikipedia. Jena (framework) — wikipedia, the free encyclopedia, 2017. [Online; accessed 13-November-2017].
- [29] Wikipedia. Json — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 13-noviembre-2017].
- [30] Wikipedia. Latex — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 30-noviembre-2017].
- [31] Wikipedia. Micología — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 8-noviembre-2017].
- [32] Wikipedia. Python — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 13-diciembre-2017].
- [33] Wikipedia. Red neuronal artificial — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 9-noviembre-2017].
- [34] Wikipedia. Sqlite — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 13-noviembre-2017].
- [35] Wikipedia. Tensorflow — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 30-noviembre-2017].