UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA INFORMÁTICA

Aplicación de recomendaciones de moda basada en redes de aprendizaje profundo

Application for fashion recommendations based on deep learning networks

Realizado por
Ismael Pineda Palencia

Tutorizado por
Rafael Marcos Luque Baena
Miguel Ángel Molina Cabello

Departamento
Lenguajes y ciencias de la computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2019

Fecha defensa:

Fdo. El/la Secretario/a del Tribunal

Resumen

Los sistemas de búsqueda tradicionales de las webs o aplicaciones móviles que realizan las búsquedas mediante un simple texto han evolucionado, dando paso así a los nuevos sistemas basados en la búsqueda por imágenes.

En el caso de las empresas del sector de la moda, la búsqueda por imágenes permite encontrar recomendaciones de productos similares al que aparece en una imagen subida por un usuario. La gran mayoría de estos sistemas utilizan técnicas de aprendizaje profundo, ideales para trabajar con datos no estructurados como lo son las imágenes. A pesar de ello, algunos de estos modelos presentan ciertos problemas, sus recomendaciones no son lo precisas que deberían ser.

El presente trabajo detalla la generación de un sistema completo, siguiendo para ello la metodología de desarrollo del "ciclo de vida clásico" del software. El sistema permite a un usuario obtener mejores recomendaciones en tiempo real que las ofrecidas por algunos de los sistemas actuales.

Para lograrlo se ha investigado e implementado varios modelos de redes neuronales convolucionales (CNN) en Python, siguiendo una metodología propia para esta tarea CRISP-DM. Para el entrenamiento y la validación se hace uso de un gran paquete de imágenes etiquetadas, mientras que para la generación de las propias redes se utiliza la API de Keras. Cada red extrae uno de los atributos del producto de una imagen, gracias a la combinación de estos, es posible encontrar otros productos similares.

El sistema se compone de un servidor en Python que hace uso de las redes neuronales y de un cliente para Android desarrollado en Android Studio, creado a partir de algunos bocetos. Integradas ambas partes, se muestran los satisfactorios resultados de las recomendaciones obtenidas en la aplicación móvil para diferentes productos.

Palabras Clave

Redes neuronales, visión por computador, perceptrón, aprendizaje profundo, aplicación Android, sistema de recomendación.

Abstract

Traditional searching systems of websites and mobile applications that perform searches by using a simple text have evolved, thus opening way to new image search systems.

In the case of fashion companies, the search by images allows users to find recommendations for similar products to those that appear in an image that they have uploaded. The vast majority of these systems use deep learning techniques, ideal for working with unstructured data such as image files. However, some of these models present certain problems, e.g. recommendations are not as precise as they should be.

The present work describes the creation of a complete system, following the development methodology known as "classical life cycle" of the software. The system allows users to obtain better recommendations in real time than those offered by some of the current searching systems.

In order to achieve this, several models of convolutional neural networks (CNN) have been investigated and implemented in Python, following the methodology CRISP-DM for this task. For training and validation, a large package of labeled images is used, while the Keras API is used for the generation of the neural networks. Each network extracts one of the attributes of the product from an image, thanks to the combination of these it is possible to find other similar products suggestions.

The system consists of both a Python server that makes use of neural networks and an Android client developed in Android Studio, created from some wireframes. Once both parts are integrated, the satisfactory results of the recommendations obtained in the mobile application for different products are shown.

Keywords

Neural network, computer vision, perceptron, deep learning, android application, recommendation system.

Índice

1. Introducción

La actividad de la sociedad de la información actual genera cada día un mayor volumen de datos, en especial datos codificados con formatos de imagen que son almacenados en nuestros dispositivos y en multitud de servidores. El principal motivo que ha originado este crecimiento casi exponencial producido durante los últimos años, es el uso de los dispositivos móviles y tabletas digitales con conexión permanente a Internet.

Medianas y grandes empresas están deseando explotar nuestros datos. Además, ya es posible afirmar que los datos que generamos se han convertido en una de las fuentes de información y de ventajas competitivas más valiosas para las empresas, tanto que algunos los consideran el petróleo del siglo XXI. La dificultad que están encontrando es cómo encontrar la forma de sacarles provecho, ya sea bien para resolver sus problemas de negocio, para generar conocimiento que suponga una mejora en el proceso de toma de decisiones o simplemente para crear nuevos servicios para los usuarios.

Las empresas buscan encontrar todo tipo de datos sobre sus clientes actuales o clientes potenciales. Entre ellos se distinguen tres tipos: datos estructurados, semi-estructurados y no estructurados. Estos últimos son los más complicados de analizar debido a que no comparten una estructura o formato común, destacar las imágenes, videos, ficheros de audio etc.

En el caso de las empresas del sector de la moda, estas conviven en un mundo cada vez más digitalizado y competitivo. Sus puntos de venta físicos se han visto reducidos en los últimos años, dado el aumento en la tendencia por parte de los clientes a realizar la adquisición de productos de forma online, ya sea bien mediante la web o las aplicaciones digitales de las propias empresas. El comercio electrónico se ha consolidado así como una fuente de ingresos en auge para la facturación de estas empresas, es además la innovación dentro de este área del negocio uno de los focos de atención principales.

Una de las recientes funcionalidades incluidas en las aplicaciones o sitios web de las empresas que comercializan productos de moda, consiste en poder realizar obtener recomendaciones de productos utilizando la búsqueda por imágenes. Dicha funcionalidad hace uso internamente de modelos de aprendizaje profundo. Esto se debe a los satisfactorios resultados que muestran al trabajar con datos no estructurados como lo son las imágenes, en comparación con los sistemas de aprendizaje automático tradicionales.

La aplicación práctica de la inteligencia artificial y en especial de las redes neuronales, puede traer consigo importantes ventajas competitivas a las empresas, Esta está siendo la tendencia actual de los nuevos proyectos que proponen y desarrollan las empresas, así como uno de los nuevos focos de investigación más sustanciales en universidades y centros de investigación por sus numerosas aplicaciones.

1.1. Motivación

Los nuevos sistemas de recomendación implementados en las aplicaciones móviles de las principales empresas comercializadoras de productos de moda, han llegado como una interesante novedad y como un importante cambio con respecto a la tradicional forma de realizar búsquedas que tenemos los usuarios.

Sin embargo, los resultados que ofrecen por el momento estos sistemas presentan un gran nivel de mejoría, para poder ser considerados como verdaderamente útiles y efectivos. Por ejemplo, algunos son incapaces de detectar qué artículo es el aparece en una fotografía aunque este se pueda distinguir con claridad y se trate de una simple camiseta. En otros casos las recomendaciones son algo lentas y además no corresponden del todo con el producto buscado, ya que en el proceso de generación de la recomendación solo se tienen en cuenta aspectos como lo son el género y la categoría del producto pero no así el color.

La motivación de este trabajo surge por tanto de la necesidad de crear un sistema de recomendaciones de artículos de moda que ofrezca respuestas en un tiempo de respuesta bajo y cuyas recomendaciones tengan en cuenta más factores para poder ofrecer recomendaciones más adecuadas que las ofrecidas por los sistemas actuales. Este sistema se podrá utilizar mediante una aplicación móvil creada para la ocasión, la cual se conectará a un servidor desde la que se hará uso de las redes neuronales.

La implementación permitirá demostrar que en un tiempo razonable es posible generar un sistema que puede utilizarse como una ventaja competitiva real de negocio para una empresa del sector de la moda.

1.2. Objetivos del TFG

Los objetivos a conseguir tras la realización del diseño, implementación y puesta a prueba del producto final de este trabajo son:

1. Diseñar y **programar varias redes neuronales** funcionales mediante el lenguaje de programación Python, con objeto de utilizarlas para generar etiquetas a imágenes analizadas. Durante su construcción se explorará y hará uso de diversas librerías de extensión de Python que permitan el desarrollo de las redes neuronales.

   Realizar con cada red una fase de entrenamiento mediante un conjunto elevado de imágenes (de artículos de moda), adecuado y representativo de la población, entendiendo población como el catálogo de productos a poder clasificar. Cada red neuronal deberá ser por tanto capaz de asignar a cada imagen que procese las etiquetas correspondientes según su similitud con los grupos de etiquetas de las imágenes utilizadas para el entrenamiento.

   Se comprobará además que el nivel de precisión en las predicciones cumple con un mínimo o nivel aceptable, como para considerar al modelo generado por la red adecuado para utilizarlo en un posible entorno real mediante técnicas de validación.

   En caso de que la calidad del modelo no se considere aceptable se volvería a repetir el entrenamiento con mayor duración o número de imágenes, a rediseñar la red neuronal, a utilizar distintos parámetros para las distintas capas de la red… de forma que mediante algunos cambios se encuentre la configuración adecuada para obtener un modelo que resulte medianamente preciso para productos que no se hayan utilizado durante el entrenamiento.

2. Crear **una aplicación móvil** para el S.O. (Sistema operativo) Android cuyo aspecto visual y funcionalidades permitan mostrar los productos de moda recomendados en base a una imagen que proporcionen los usuarios a la aplicación.

   Esta aplicación deberá poder conectarse a una red para enviar la imagen a un servidor, en él se procesará la imagen en tiempo real mediante las redes neuronales diseñadas. Se generarán las etiquetas oportunas para la imagen recibida, etiquetas que posteriormente serán enviadas a la aplicación junto a varias imágenes de productos similares para mostrar así las recomendaciones de moda.

3. Elaborar un programa que se ejecutará y actuará en un equipo a modo de **un servidor**. Diseñado en Python, permitirá utilizar las redes neuronales para generar las etiquetas apropiadas a una imagen que podrá recibir, elaborar las recomendaciones y realizar el envío a la aplicación cliente de las etiquetas e imágenes recomendadas, todo ello en tiempo real.

4. Preparar una **arquitectura** que permita utilizar la solución mediante una conexión segura y estable entre la aplicación móvil y el servidor. La conexión debe realizar de forma segura y correcta para permitir el envío de una imagen desde la aplicación, garantizando la seguridad de la información enviada por el usuario. Los datos serán cifrados mediante algún algoritmo de cifrado seguro.

1.3. Organización de la memoria

La memoria de este trabajo se estructura en base a diferentes capítulos, en los que podemos es posible encontrar lo siguiente:

2. **Estado del arte y tecnologías utilizadas**: En él se estudian y analizan algunas de las soluciones existentes para el problema que ha motivado este trabajo. Se marcan una serie de objetivos a alcanzar una vez se complete el desarrollo de la solución, junto con el conjunto de herramientas y la infraestructura utilizada para su desarrollo.

3. **Introducción a las redes neuronales**: Un primer acercamiento a modo de investigación introduce el concepto de las redes neuronales, pasando por conocer sus orígenes, los elementos que las componen, estudiar cómo funcionan y cómo aprenden… hasta llegar a analizar cómo es un modelo de red completa.

4. **Redes convolucionales**: Con motivo de aplicar las redes neuronales sobre imágenes, en este capítulo se detalla y describe las técnicas, elementos y componentes que describen un tipo particular de red neuronal conocida como red neuronal convolucional.

5. **Estudio del problema de negocio**: La adquisición general de conocimientos de negocio y el estudio del problema que las redes neuronales pueden resolver en el negocio de la moda. Es a partir de este índice a partir del cual comenzaría una de las fases cuya equivalencia correspondería a seguir la primera fase de la metodología CRISP-DM.

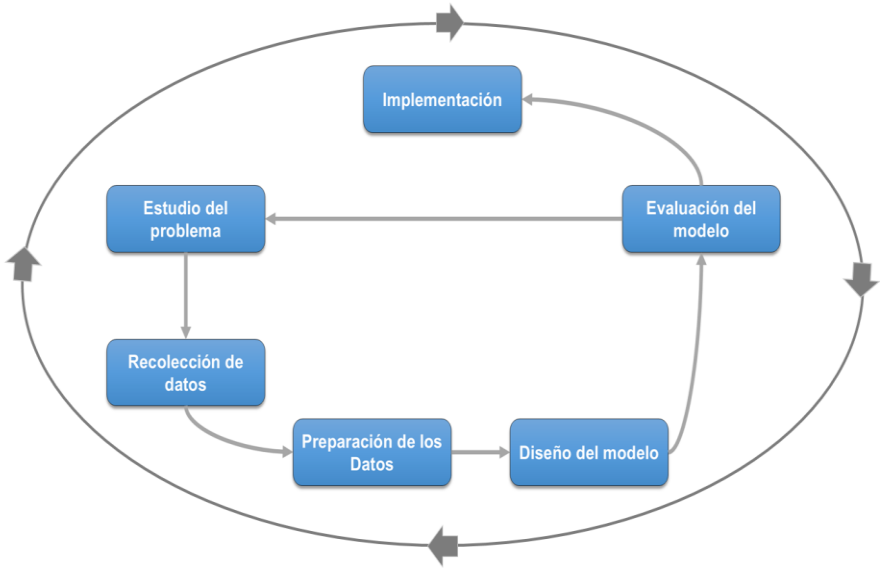


Figura 1. Esquema de la metodología CRISP-DM (Diseño propio a partir de Piatestsky, 2019).

6. **Planificación, análisis y diseño**: Comenzando por la elaboración de la planificación temporal del proyecto, se pasará al análisis de las funcionalidades y los requisitos que el sistema deberá cumplir, marcando así los objetivos o pautas del desarrollo y el propio diseño de la aplicación móvil. Por último se buscará una fuente de datos para el sistema.

7. **Desarrollo, pruebas y despliegue del sistema**: Se describe en primer lugar el proceso de preparación de la fuente de datos para entrenar y validar las redes neuronales que se creen. Se detalla el proceso de desarrollo de los elementos que componen el sistema (servidor y cliente), así como el proceso de integración entre ambas partes y su posterior despliegue.

8. **Conclusiones**: En él se pone de manifiesto si se ha logrado completar con éxito los objetivos marcados en el capítulo dos. Se plasman las dificultades, los conocimientos adquiridos y se evalúa si el sistema podría ser utilizado en un entorno real de producción o de las ampliaciones que debería sufrir para que esto ocurra.

9. **Referencias**: Capítulo en el que aparecen las fuentes utilizadas para la elaboración de la memoria y la obtención de algunos de los conocimientos necesarios para la elaboración completa del sistema.

10. **Apéndices**: Después del propio cuerpo de la obra, en este último capítulo se incluyen los elementos que complementan a la memoria y que han sido generados durante las distintas fases que componen la creación del sistema. Asimismo se detalla mediante un manual la forma de replicar y poder ejecutar el sistema de recomendaciones de forma sencilla.

2. Estado del arte y tecnologías utilizadas

En este segundo capítulo analizaremos el estado actual de las soluciones existentes para el problema anteriormente propuesto. Destacaremos sus fortalezas y sus carencias para poder formalizar de este modo una propuesta distinta que será desarrollada mediante diversas tecnologías y herramientas como solución alternativa al problema.

Este estudio permite comenzar la primera fase de la metodología *CRISP-DM* que abarca el hecho de comprender el problema y el negocio, fase que puede considerarse como una de las más importantes. Una mala decisión en la elección de las herramientas escogidas para el desarrollo podría impedir por ejemplo que se pueda migrar la solución a un entorno real de producción.

Sin la correcta comprensión del problema ni una suficiente exploración de las tecnologías y las herramientas, se pueden acarrear problemas y obligarnos a comenzar nuevamente casi desde cero, con objeto de evitarlo prestaremos mucha atención a esta fase.

2.1. Análisis del estado del arte

Durante los últimos 5 años el interés por el aprendizaje automático (*machine learning)* y en particular por el aprendizaje profundo (*deep learning)* como forma de encontrar una solución a problemas reales de negocio ha crecido considerablemente (ver figura 2). Este interés comenzó a aumentar a partir del año 2016 motivado por las publicaciones y aportaciones de las grandes empresas tecnológicas: *Google Now, Cortana, Deep Mind, Chatbots…*

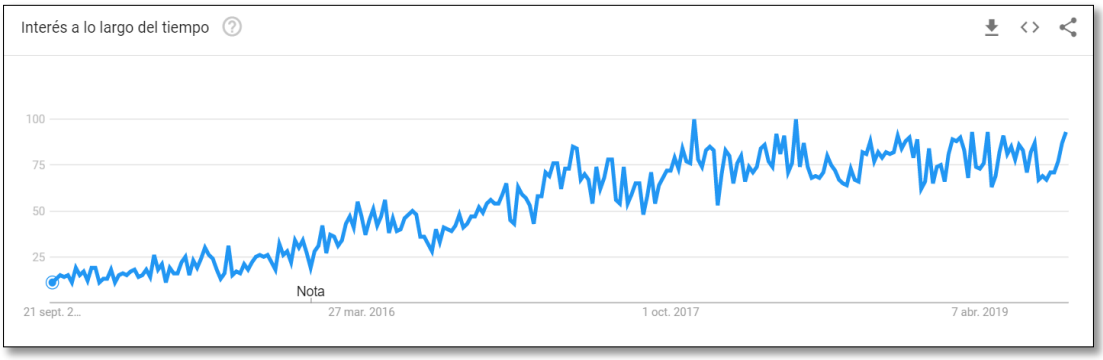


Figura 2. Evolución del interés en búsquedas por las convolutional neural networks (Google Trends, 2019).

Un gran número de empresas de todos los sectores, incluido el de la moda, se han sumado a la tendencia y ya han encontrado la forma de implantar sus soluciones basadas en modelos de aprendizaje profundo en sus aplicaciones o webs. Destacar entre estas soluciones a los asistentes inteligentes, los sistemas de recomendación productos o de contenido etc. Como se puede notar el número de aplicaciones prácticas es muy amplio.

Entre las aplicaciones relacionadas con el aprendizaje profundo una de las que más importancia ha conseguido es la identificación de objetos en imágenes. Atendiendo al sector de la moda, algunas grandes empresas han integrado en sus aplicaciones móviles una funcionalidad que permite a los usuarios realizar búsquedas y obtener recomendaciones de productos mediante la subida de imágenes.

La empresa sueca H&M ofrece la posibilidad a sus usuarios de buscar artículos en su tienda mediante una imagen que se sube desde su aplicación móvil. La búsqueda se convierte así en una tarea más cómoda y rápida (ver figura 3). Por los resultados de las búsquedas, es posible afirmar que en la actualidad su modelo se enfoca en mayor medida en encontrar productos de la misma categoría y género, más que aquellos que también compartan el color.

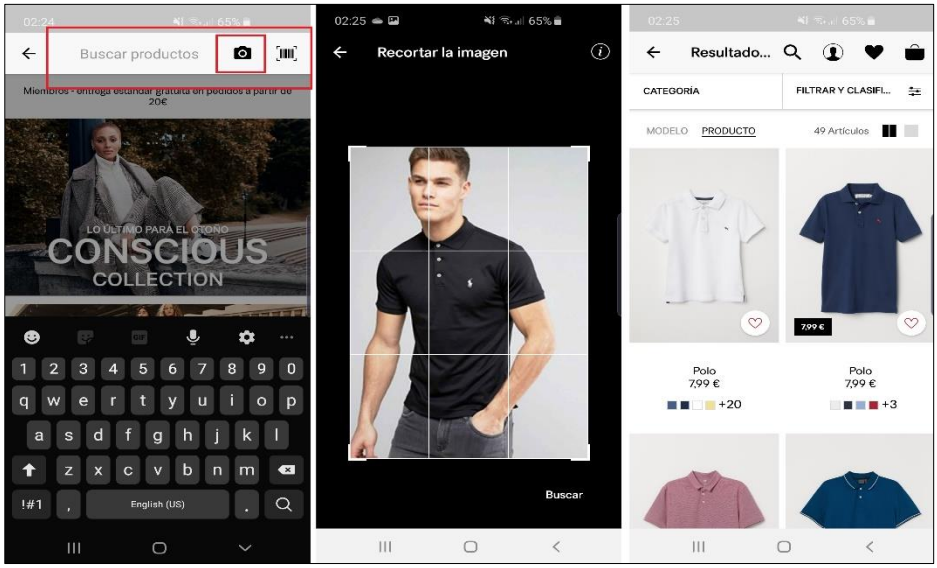


Figura 3. Resultados de una búsqueda mediante imagen en la App de H&M (Diseño propio).

Amazon Shopping forma parte de la empresa Amazon. Esta empresa también ha implementado la búsqueda de productos por imágenes, no solo de moda sino de todos los que vende. En términos de calidad de búsqueda relativos a productos de moda no ofrece resultados del todo satisfactorios (ver la figura 4).

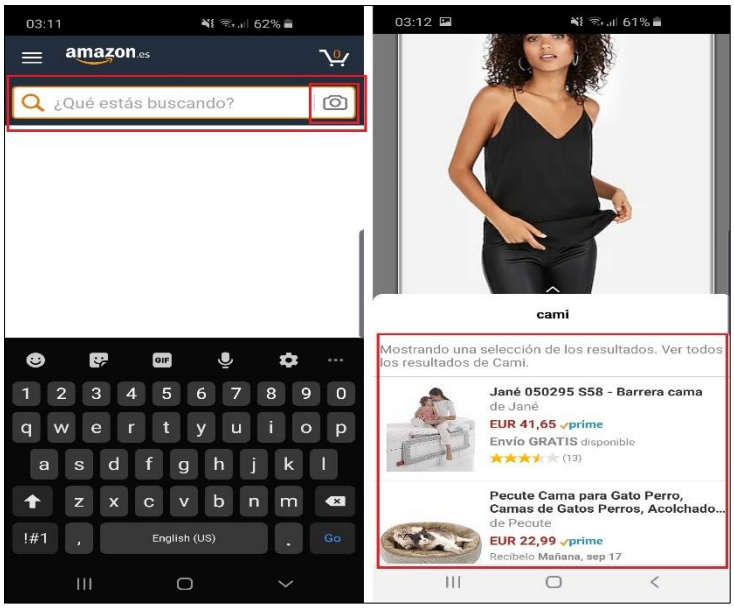


Figura 4. Resultados de una búsqueda mediante imagen en la App de Amazon (Diseño propio).

La aplicación móvil de la empresa británica de moda New Look presenta la búsqueda mediante imágenes con una interfaz ligeramente distinta pero con el mismo fin. El proceso suele demorarse en la versión actual bastantes segundos y los resultados no siempre son los más adecuados. Acorde a la imagen del producto buscado nuevamente no encontramos las condiciones del color ocurriendo así lo mismo que en la de H&M (ver figura 5).

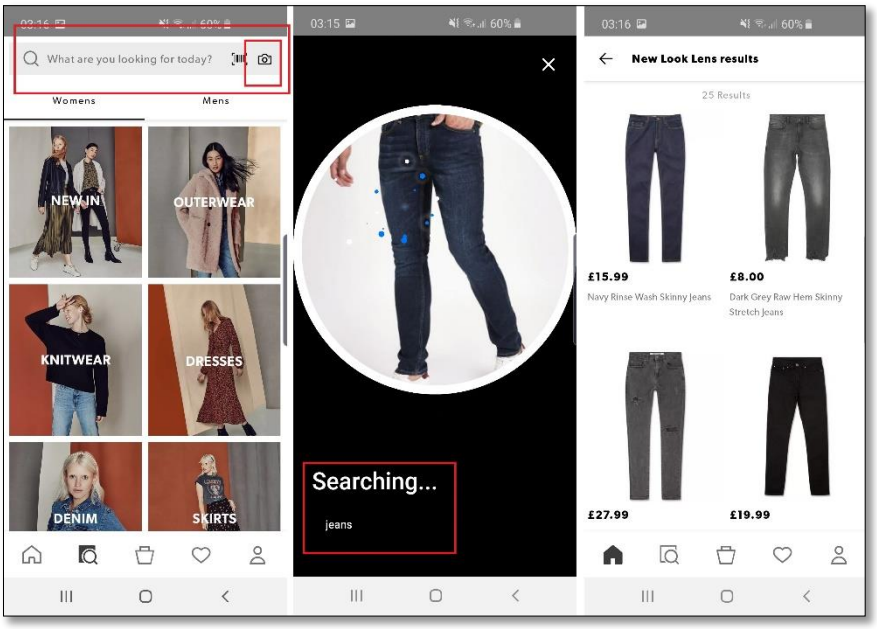


Figura 5. Funcionalidad de búsqueda mediante imagen en la App de New Look (Diseño propio).

La incógnita aún por descubrir es cómo funcionan internamente estos motores de recomendación. Para encontrar una respuesta basta con revisar publicaciones de investigación realizadas por trabajadores de estas empresas en colaboración con las Universidades. En algunos se exponen que la nueva herramienta que utilizan para trabajar con la información no estructurada, como lo son las imágenes, son las redes neuronales convolucionales (CNN).

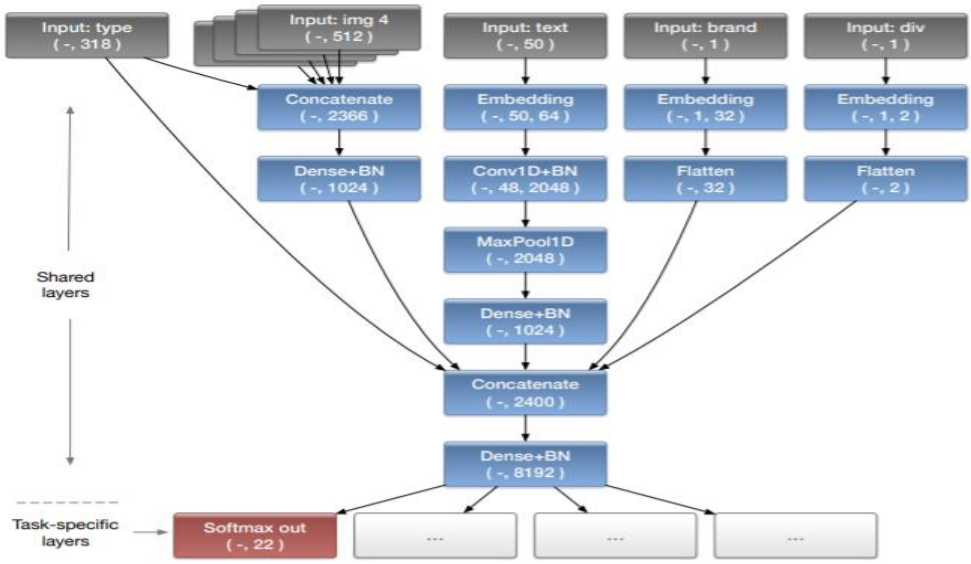


Figura 6. Esquema de la arquitectura de red neuronal de ASOS (Cardoso, Daolio, Vargas , 2018, fig. 2, p.83).

Para diseñar y programar redes neuronales disponemos de diferentes alternativas. Entre las más populares y utilizadas por los desarrolladores se encuentran Python y R, dos lenguajes enfocados en el manejo de datos, modelos de computación y cálculo estadístico. Ambos son igualmente válidos pero en el caso de Python destacar que este cuenta con un mayor número de librerías, interfaces y paquetes preparadas para el manejo de imágenes así como para el desarrollo de redes neuronales: Tensorflow, PyTorch o Keras. Esta último agiliza la creación de redes a pocas líneas de código, lo que simplifica en gran medida el desarrollo.

Tras analizar el funcionamiento de las aplicaciones anteriores podemos encontrar un patrón de funcionamiento común: subir una imagen de la galería o tomarla con la cámara, esperar a que sea procesada por sus modelos de redes y posteriormente visualizar los productos recomendados en base a la imagen facilitada.

La situación actual de estos modelos admite un amplio margen de mejora no solo en términos de identificación de artículos sino también en los criterios establecidos para realizar las recomendaciones. En este aspecto el desarrollo de este TFG tiene como objetivo crear un nuevo sistema de recomendaciones de productos de moda basándose en el esquema de funcionamiento de los anteriores.

Este sistema trabajará con imágenes de los usuarios que serán procesadas mediante redes neuronales creadas para el caso y se montará sobre una infraestructura sencilla pero segura. Dicho sistema podrá utilizarse a través de una aplicación diseñada para Android con una interfaz que permita un uso intuitivo de la misma. Se persigue además que los resultados puedan ser considerados como mejores en comparación con los que nos podrían ofrecer los sistemas actuales de las empresas que venden productos de moda.

2.2. Tecnologías y herramientas

La elección de las herramientas y las tecnologías a utilizar en un proyecto es una de las decisiones más relevantes, delicadas y más compleja de lo que puede parecer. En un primer lugar tras detallar y estudiar el problema de negocio al que se quiere hacer frente es necesario valorar en este orden los siguientes puntos que dictaminarán esta elección:

1. **Alcance del proyecto**: Esta primera fase pasa por evaluar aspectos como los conocimientos de los que se dispone relativos al mercado y a las soluciones adoptables para la ejecución del proyecto, estudiar el número estimado de usuarios que harían uso de la aplicación, el coste que supondría la explotación y mantenimiento… todo ello con el objetivo de conocer el alcance real que supone realizar el proyecto.

2. **Tecnologías de producción**: Tras evaluar el alcance, es necesario contemplar y analizar las distintas tecnologías mediante las cuales el proyecto puede realizarse. Se tendrán en cuenta aspectos tales como los requisitos de las tecnologías y herramientas software, la integración que tengan entre sí y entre ellas con los sistemas actuales en el que caso de que los haya, la flexibilidad de desarrollo y de llevar el desarrollo a un entorno de producción real, la vida esperada y el grado de obsolescencia que puede sufrir la solución… esto marcará principalmente la línea de opciones disponibles para el desarrollo.

3. **Experiencia y capacidad de desarrollo**: Cerciorarse de las limitaciones reales de los conocimientos del equipo o desarrollador en este caso, así como de su capacidad de adaptación a nuevas tecnologías es algo muy importante. En caso de no se pueda asumir ese cambio quizás sea necesario contemplar un proceso de formación, nuevas contrataciones o incluso externalizar parte del desarrollo o del mantenimiento del producto desarrollado, aunque no sea este el caso.

4. **Mantenimiento y escalabilidad**: Una vez el proyecto se realice puede surgir la necesidad de escalar la solución, debido a una mayor carga de trabajo con respecto a la valorada inicialmente. Esto puede acarrear problemas de rendimiento y mayor necesidad de inversión. Destacar también que en los entornos reales el mantenimiento de la solución es una necesidad real por lo que la tolerancia a fallos y la necesidad de solucionar los fallos o caídas que se sufran en el menor tiempo posible son aspectos que marcarán la elección de la tecnología y las herramientas a utilizar.

Siguiendo los ideales de los puntos mencionados algunos elementos han sido claves en la elección de **Python** y **Android** como **tecnologías de desarrollo** para el sistema.

Por un lado en cuanto a las tecnologías de producción (punto 2) la elección de Python y Android supone una mayor libertad de despliegue gracias a su compatibilidad con multitud de dispositivos y de sistemas operativos. Ambos disponen de buenos entornos de desarrollo y de una gran capacidad de integración tanto entre ellos como con otras herramientas.

Teniendo en cuenta el aspecto relativo a la experiencia y capacidad de desarrollo (punto 3) se parte de una base de conocimiento teórico/práctica media de las redes neuronales, que será profundizada a medida que se desarrolle este trabajo. Por parte del desarrollo en Android se cuentan con nociones muy generales pero con capacidades de adaptación rápida. Por último la elección de estas tecnologías permite escalar el proyecto (punto 4) de forma muy sencilla mediante un servidor más potente ubicado en un servidor no local, al que se le añadan más capas de seguridad y que acepte múltiples consultas.

En el caso particular de este trabajo, se ha optado por trabajar en el **desarrollo de las redes neuronales** y del propio **servidor** en una suite de desarrollo pensada para proyectos del ámbito de la ciencia de datos, conocida como **Anaconda**. En líneas generales es considerada como una distribución libre de Python con un gestor de entornos, de paquetes y colecciones. Se trabajará en el **IDE** (*Integrated Development Environment*) web incluido de **Jupyter Notebook** donde se programará en el lenguaje de programación **Python** haciéndose uso de distintos paquetes o interfaces como Numpy, Pandas, **Keras** o PIL para manejar datos y crear las **redes neuronales** y de otros como Cripto, Socket u OS para el diseño del **servidor**.

Se elaborarán una serie de diagramas con los posibles **casos de uso** (descripción gráfica de las acciones del sistema) de la aplicación móvil mediante la herramienta **MagicDraw UML**, tras ello comenzará la captación de requisitos. Los requisitos marcan las condiciones, funcionalidades y aspectos que debe cumplir la aplicación, también marcan parte del diseño de la interfaz que se verá guiado por los bocetos iniciales.

Los bocetos de la aplicación móvil serán elaborados en la herramienta web ***Moqups.*** Con esta herramienta diseñaremos de forma rápida *wireframes* (bocetos) que sirven como plantilla para el posterior desarrollo de la interfaz visual real de la aplicación. Se atenderá a una

correcta distribución de los elementos en la pantalla UI (*User Interface*) como lo serán los iconos, imágenes, campos de texto, botones, banners etc facilitando al usuario su experiencia de uso UX (*User eXperience*).

Por otro lado la **aplicación** móvil será exclusivamente para el sistema operativo Android y se desarrollará en el entorno **Android Studio**, propiedad actual de Google. Se seguirá como modelo el patrón de arquitectura **MVC** (Modelo Vista Controlador). Dentro del entorno se elaborará una interfaz visual interactiva (**Vista**) mediante ficheros **XML**, la parte que controla los eventos originados por la vista y así como el flujo de datos (**Controlador**) se desarrollará en **Java** y por último, la información que se muestra y genera (Lógica del negocio) se obtiene de la comunicación con el servidor (**Modelo**).

Destacar que para realizar una organización y gestión eficiente del tiempo a dedicar al proyecto se seguirán los diagramas de Gantt, elaborados para la ocasión mediante la herramienta gratuita **GanttProject**. El diagrama funciona a modo de herramienta gráfica permitiendo exponer el tiempo de dedicación previsto para las diferentes tareas que componen el proyecto y conforma el elemento de partida para comenzar con el proyecto.

2.3. Arquitectura

La arquitectura de un sistema describe los componentes que intervienen en el ecosistema así como las conexiones existentes entre los propios componentes.

El esquema de la arquitectura propuesta para el funcionamiento del sistema (figura 7) requiere de obligatoriamente un servidor desarrollado en Python y al menos un *smartphone* con la aplicación instalada *(Cliente)*, ambos conectados a la misma red. El cliente con Android ejecutará la aplicación, enviará y recibirá los datos e imágenes mediante una conexión establecida por sockets con el servidor.

El servidor será ejecutado en un equipo con el sistema operativo Windows en un *notebook* de Python. Comenzará a actuar tras el establecimiento de la conexión y la recepción de la imagen por parte del cliente. La imagen es facilitada a los modelos, con la información que generan, el servidor podrá buscará a modo de recomendación las imágenes más similares a la recibida. Tras ello, las imágenes seleccionadas serán enviadas, junto con la información obtenida por los modelos, de vuelta al cliente, dándose así por concluida la conexión.

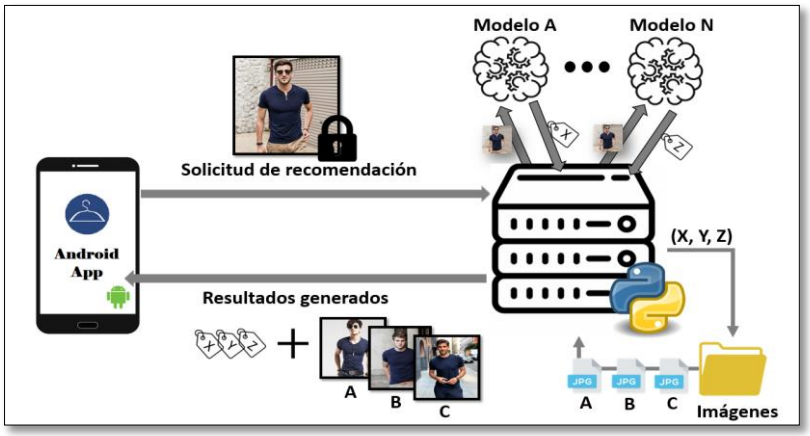


Figura 7. Esquema de la arquitectura del sistema diseñado (Diseño propio).

3. Introducción a las redes neuronales

El fundamento de las redes neuronales artificiales está inspirado en el complejo comportamiento biológico de las redes neuronales que se encuentran en el cerebro humano. Las interconexiones electro-químicas que realizan constantemente los millones de neuronas, permiten al ser humano entre otras cosas aprender, comunicarse o retener información.

No es otro el objetivo de los científicos e investigadores de las ciencias de la computación especializados en la disciplina de la Inteligencia Artificial, que el de emular mediante máquinas y modelos el comportamiento de las neuronas de una forma más abstracta y menos compleja. Por lo general se traduce en un menor número de conexiones o de unidades neuronales utilizadas en dichas redes.

Estos modelos artificiales han demostrado ser capaces de "aprender" por si mismos , mediante conjuntos de datos destinados para su aprendizaje y de poder dar solución a problemas del mundo real, en los que las técnicas algorítmicas tradicionales no habían logrado ofrecer unos resultados del todo satisfactorios. Entre estos problemas destacar aquellos relacionados con el reconocimiento de imágenes y de voz así como el procesamiento y la traducción automática de textos.

3.1. Inspiración biológica

Todos las especies animales incluida la raza del ser humano vive con el instinto natural de la curiosidad. La principal diferencia de la curiosidad humana reside en nuestro pensamiento abstracto, este nos conduce en ocasiones a la fantasía, a la imaginación y al concepto de mímesis definido por Aristóteles. Entendido como la imitación de la naturaleza y de la información que recogen nuestro sentidos con el fin esencial del arte.

Nuestra curiosidad define nuestro comportamiento psicológico y nos impulsa a la necesidad de explorar, investigar y aprender acerca de lo desconocido en nuestro ambiente natural y en el de otros seres a nuestro alrededor. Los conocimientos estructurados, técnicas de estudio e interpretaciones que se han realizado a lo largo de la historia sobre distintas temáticas han dado lugar a las distintas ramas que componen la ciencia. Es el conocimiento científico el que responde a esos interrogantes generados por la curiosidad del ser humano, pero este es complicado y a diferencia del conocimiento común este se adquiere de una forma más compleja, hace uso del método científico definido por Francis Bacon y René Descartes.

La neurociencia es uno de los campos de la ciencia más complejos y con más incógnitas aún sin respuesta, dentro de la ciencia. Su foco de estudio es el sistema nervioso, junto con la interacción de este con el resto de elementos del organismo, estas conexiones dan lugar a las cognición y la conducta biológica. En el caso de los seres humanos el órgano que define su comportamiento, que origina sus habilidades comunicativas, la percepción de la realidad, le dota de capacidad aprendizaje o de retención de información mediante la memoria es el cerebro. En palabras de Hipócrates de Cos, figura considerada como uno de los padres de la medicina:

> *Los hombres deben saber que el cerebro es el responsable exclusivo de las alegrías, los placeres, la risa y la diversión, y de la pena, la aflicción, el desaliento y las lamentaciones. Y gracias al cerebro, de manera especial, adquirimos sabiduría y conocimientos, y vemos, oímos y sabemos lo que es*

repugnante y lo que es bello, lo que es malo y lo que es bueno, lo que es dulce y lo que es insípido (Hipócrates)

En el año 1971, el médico y biólogo Luigi Galvani descubrió la existencia de actividad eléctrica en el cerebro humano, esta actividad es generada por las células del sistema nervioso conocidas como neuronas, elemento destacado en la teoría "doctrina de la neurona" propuesta por Santiago Ramón y Cajal a finales del siglo XIX.

Las neuronas son células dispuestas e interconectadas que se encuentran en el organismo más complejo del ser humano, el cerebro. Un ser humano adulto puede albergar aproximadamente unos cien millones de neuronas, que conectadas entre sí forman una red de neuronas. Esta red puede debilitarse o fortalecerse mediante nuevas conexiones a través del aprendizaje. Cada neurona se compone principalmente de tres partes, detalladas a continuación y cuya disposición puede observarse en la figura 8.

- **Cuerpo celular:** Representa la fuente del origen y el núcleo de la energía que se genera para el funcionamiento la neurona, denominados impulsos. La forma del cuerpo puede ser variable.

- **Dendritas**: Son las prolongaciones del cuerpo de la célula en forma de ramificaciones. Su principal función es detectar, recoger y hacer llegar al cuerpo de la célula los impulsos de otras células cercanas.

- **Axón**: Esta fibra nerviosa es una prolongación del cuerpo que se sitúa en la parte opuesta a las propias dendritas, cuya terminación nerviosa también es ramificada. Acomete la función de enviar y conducir los estímulos de la neurona hasta otras neuronas, músculos o glándulas del organismo mediante el punto de contacto final de sus ramificaciones, denominado sinapsis.

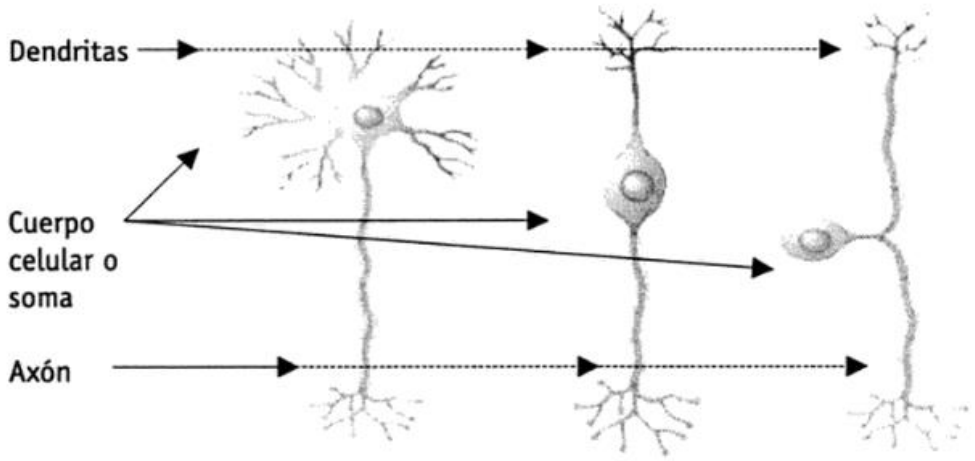


Figura 8. Diagrama esquemático de una neurona biológica (Florez López, Fernandez Fernandez, 2008, p. 13).

El comportamiento de la red de neuronas el objetivo a imitar, de una forma más simple y abstracta, por los modelos de computación. Estos modelos hacen uso la idea de la interconexión entre neuronas, elementos que pasarán a ser llamados perceptrones. Esta conexión conforma una red cuyo comportamiento depende así del conjunto del sistema.

3.2. Las redes neuronales artificiales

Las redes neuronales artificiales deben su origen a las aportaciones que realizó el considerado como el padre de la IA, Alan Mathison Turing con su trabajo *Turing´s Connectionisn: An Investigation of Neural Network Architectures* publicado en 1936. Turing fue uno de los primeros investigadores que se dedicó a estudiar el cerebro desde un punto de vista computacional.

En 1943, Warren McCulloch y Walter Pitts, neurofisiólogo y matemático respectivamente publicaron el primer modelo matemático abstracto basado en simulaciones lógicas formales que imitan el funcionamiento de una red neuronal mediante circuitos eléctricos. Unos años más tarde, en 1949, el iniciador de la biopsiología, Donald O. Hebb definió el principio o "Ley de Hebb" el cual es fundamental para comprender la relación entre la psicológia y la neurociencia, este principio explica los procesos del aprendizaje humano y se conforma como una regla que formaliza el funcionamiento de este aprendizaje. Considerada como la predecesora de la mayoría de los algoritmos de aprendizaje de las redes neuronales actuales.

El aumento del número de aportaciones y el creciente interés de los investigadores que perseguían crear máquinas que simularan cada característica de la inteligencia humana dio lugar a que en 1956 se celebrase el *Dartmouth Summer Reserarch Project on Aritificial Intelligence* considerado el primer evento que reunía a expertos en el emulgente campo de la IA. En 1958, el filósofo Frank Rosenblantt sumaría su contribución a este campo mediante un modelo estadístico simplista que denominó Perceptrón, este modelo formalizaba con una simple neurona el modelo anteriormente propuesto por McCulloch y Pitts. Esta nuevo modelo era bastante limitado pero supuso un paso importante, ya que comenzaba a simular la inteligencia humana mediante una regla de aprendizaje que corregía el error. Permitía reconocer patrones y aprender a reconocer otros nuevos que no haya tratado previamente, este estudio lo publicó más ampliado junto en su obra *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms (1962).*

Para solventar las limitaciones del modelo del perceptrón, el profesor Bernard Widrow junto a su alumno estudiante de doctorado Ted Hoff, desarrollaron juntos en 1960 un algoritmo que mejoraría el aprendizaje del perceptrón gracias a técnicas de mínimos cuadrados (LMS) . Esta técnica permite justar los pesos de la red según sea necesario. Este avance les permitiría desarrollar el primer tipo de RNA compleja denominada *ADALINE* (*ADAptative LINear Element*), la cual se utilizó para resolver un problema real de aquel entonces relacionado con las señales de las líneas telefónicas y el eco durante las llamadas.

Estas esperanzadoras aportaciones llevaron a muchos a pensar que la verdadera revolución en este campo no había hecho más que empezar, lamentablemente el interés comenzó a decaer por parte de numerosos investigadores tras la demostración de las limitaciones reales con las que contaba el perceptrón. En el libro "*Perceptrons*" escrito por Marvin Lee Minsky y Seymour Papert se critica con dureza a las redes neuronales y se demuestra la incapacidad de aprender funciones que no sean separables de forma linealmente.

No fue hasta mediados de los años 80 cuando resurgió el interés, gracias a nuevos eventos divulgativos sobre las redes neuronales donde se presentaron nuevos descubrimientos. El algoritmo de aprendizaje denominado *backpropagation* (retropropagación) del error publicado en 1986, cuyo autor material principal es David Everett Rumelhart junto a Geoffrey Hinton y Ronald J. Williams, supuso una importante aportación ya que se consolidaba como la solución ante los problemas del aprendizaje limitado del perceptrón simple. Dicha regla de aprendizaje es una de las más utilizadas durante el entrenamiento de redes multicapa.

Ese interés por el campo de la inteligencia artificial mediante las redes neuronales volvió para quedarse. Prueba de ello han sido las creaciones de diversas sociedades entre ellas la ENNS (*European Neural Network Society*) o de las nuevas conferencias que se llevan realizando anualmente desde 1987 como "*Neural Information Processing System*". Los objetivos de investigación de los estudios actuales centran sus objetivos en lograr una sinergia entre la capacidad de procesamiento y de aproximación en las redes neuronales artificiales. Se persigue encontrar soluciones considerables como buenas, en tiempos cortos y con pocos datos de partida para el aprendizaje, de esta forma es posible hacer uso de ellas en entornos de producción en tiempo real.

3.3. Aplicaciones prácticas

Las RNA se encuentran entre las técnicas de IA con más futuro y con un constante crecimiento en cuanto al número de investigadores, repartidos por todo el mundo. Sus aplicaciones prácticas son muy numerosas pero sobre todo las redes destacan por su eficiencia y su flexibilidad. Ya existen multitud de aplicaciones que disponen de RNA implantadas en los dispositivos que podemos adquirir como usuarios o bien en los propios sistemas que procesan nuestros datos. Utilizarlas permite encontrar patrones, reconocer tendencias o procesar con relativa facilidad principalmente datos no estructurados que otras técnicas no aceptan o sobre los que no ofrecen resultados satisfactorios.

Algunas de los campos y principales aplicaciones actuales de las redes neuronales son:

- **Medicina**: El uso de las redes neuronales en la medicina ha supuesto un increíble avance en este campo, especialmente en la detección de patrones que puedan inducir a claros síntomas de una enfermedad, predicciones de las probabilidades de reacción ante combinaciones de diversos medicamentos y de una posible parada cardíaca en personas con marcapasos… estos son solo algunos de los muchos usos que se le está dando, en un campo en el que cada día cobran mayor importancia.

- **Administrativo**: La detección de falsificaciones en las firmas de los documentos así como el reconocimiento de los caracteres que aparecen en los textos o documentos escritos tanto a mano como en el caso de los digitales. La tarea que podía ser rudimentaria y manual, las redes neuronales la han permitido agilizar.

- **Energético**: La predicción de los momentos de mayor y menor consumo de energía por los usufructuarios de energía ya sea en forma de luz, agua o gas entre otros también es gestionada en algunas empresas por modelos de redes neuronales.

- **Geología**: La predicción y prevención ante posibles riesgos naturales con consecuencias devastadoras para los humanos son de vital importancia, por ello los laboratorios que estudian estos fenómenos ya han decidido incorporar RNA.

- **Telecomunicaciones**: La posibilidad de clasificar y separar las señales recibidas o trasmitidas por un mismo canal o la predecir la demanda de servicios por parte de los usuarios para estudiar la necesidad de nuevos emisores de señales hacen a las RNA especialmente interesantes.

- **Automovilístico**: La conducción autónoma requiere de multitud de sensores y del análisis de los elementos reconocidos por las cámaras incorporadas para la toma de decisiones. Para lograrlo es necesario reconocer y distinguir a otros vehículos, a personas, señales… en fotografías tomadas en tiempo real.

- **Comercio**: Los *e-commerce* (comercio electrónico o digital) buscan nuevas formas de conocer a sus clientes para sus campañas de marketing, predecir la demanda de productos para estimar necesidad de reposición de stock… las redes neuronales suponen una de esas nuevas formas de mejorar la rentabilidad de este tipo de negocios.

3.4. Perceptrón simple

El filósofo Frank Rosenblantt presentó durante los años 50 el primero de los modelos de RNA actuales, el perceptrón simple. Este modelo consta de una arquitectura sencilla debido a que tan solo existe una salida, se dispone de una única capa (monocapa) compuesta por una o múltiples entradas (Ver figura 9).

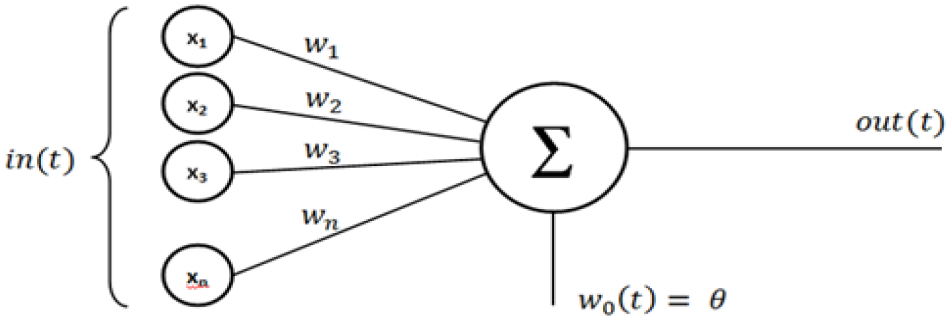


Figura 9. Perceptrón simple con 1 entradas (Diseño propio a partir de Mayranna, 2013).

Como analogía entre el perceptrón y la neurona humana, es posible ver como el perceptrón se compone de un sumatorio central. Este sumatorio actúa como núcleo de la neurona siendo este el encargado de procesar las distintas señales recibidas por las dendritas, además de un valor de umbral o sesgo $\theta \in \mathbb{R}$. Las señales son modeladas mediante los datos recibidos a través de las distintas entradas X_i considerados como los parámetros de entrada (*inputs*) de la función los cuales se ven afectados por sus respectivos pesos sinápticos asociados w_i. Conectado al núcleo se encuentra el axón en el que según el caso se coloca por lo general una función de activación σ y tras él se encuentra la salida (*output*) que se puede conectar si se desea con uno o más perceptrones (sinapsis).

El objetivo principal de entrenamiento de una neurona o en general de una red neuronal no es otro que el de encontrar los valores más adecuados para los pesos w_i y el sesgo θ para que la salida obtenida se aproxime lo máximo a la esperada. En el caso del perceptrón donde solo hay una salida y unos parámetros de entrada predefinidos la elección correcta de los pesos y el sesgo es la pieza fundamental que definirá su correcto funcionamiento.

Por regla general la descripción matemática que describe el comportamiento común de un perceptrón (función de entrada), se calcula mediante la suma de las entradas multiplicadas por sus pesos junto al sesgo y se dispone de la siguiente forma:

$$in(t) = y' = \sum_{j=o}^{n} w_j x_j - \theta$$

No obstante el valor real o la salida del perceptrón lo determina un elemento más, la función de activación. Esta se coloca tras la función de entrada, puede ser de distinta según el uso que se requiera del perceptrón, siendo el esquema final de un perceptrón el siguiente:

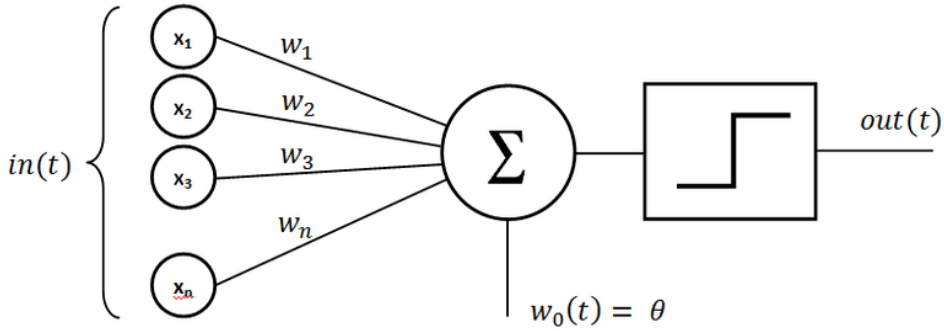


Figura 10. Perceptrón simple con n entradas con función de activación umbral (Mayranna, 2013).

3.5. Funciones de activación

Las neuronas biológicas no solo transmiten la información que reciben, también la procesan, de esto se encarga la función de activación. Los valores obtenidos en la función de entrada, función mencionada en la formula anterior, abarcan todos los posibles valores ya que $y' \epsilon (-\infty, +\infty)$. Para marcar una neurona como activada o no, de forma simplificada se marca una tasa de potencial que establece si se activa o no, dentro un rango de valores.

La salida final de un perceptrón viene dada por el resultado de la salida de la función de activación, la ecuación que define dicha salida se observa a continuación:

$$out(t) = y = \sigma(y') = \sigma\left(\sum_{j=o}^{n} w_j x_j - \theta\right)$$

Una de las funciones de activación más comunes es la función umbral o escalonada:

$$\sigma(x) = \begin{cases} 1 \;\; si \left(\sum_{j=o}^{n} w_j x_j \; - \; \theta \right) > 0 \\ 0 \;\; si \left(\sum_{j=o}^{n} w_j x_j \; - \; \theta \right) < 0 \end{cases}$$

La función umbral al igual que cualquier otro tipo de función de activación se suele expresar no solo mediante su ecuación sino también mediante su representación gráfica para facilitar su compresión tal y como se aprecia en la siguiente figura.

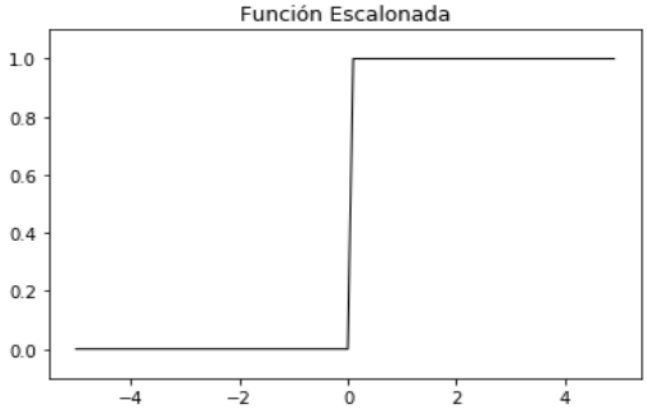


Figura 11. Representación de la función escalonada (Diseño propio).

Situándonos en el eje de abscisas se comprenden a todos los posibles valores que genere la función de salida siendo estos los comprendidos por $y' \epsilon (-\infty, +\infty)$ y en el eje de ordenadas contemplamos a las dos salidas que genera la función de activación escalonada $y' \epsilon \{0,1\}$. Según la función utilizada la disposición que presente el eje de ordenadas será distinta, ya que de igual forma los valores de entrada a la función comprendidos en la función de salida no cambian.

Tras comprender el contexto teórico y matemático que hay tras una función de activación, toca explorar su utilidad real en un contexto práctico. Retomando la función de activación escalonada, la función contempla dos valores de salida siendo estos el 0 o el 1, este tipo de función es ideal para desarrollar por ejemplo un clasificador binario. Si se asocia el valor 1 en caso de que se detecte que los parámetros del patrón de entrada $(x_1, x_2 \dots, x_n)$ corresponden a un elemento de una clase A y el valor 0 en caso de que dichos parámetros permitan deducir que el elemento no pertenece a esa clase, la función escalonada se presta como una buena solución ante este tipo de problemas.

El tipo de problema a resolver es un aspecto a tener en cuenta como pauta durante la selección de la función de activación a utilizar, para contextualizarlo con más detalle véase la siguiente tabla en la que se detallan algunas de las principales funciones de activación, su representación y sus características prácticas.

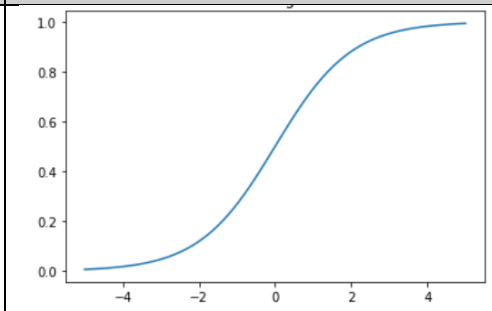
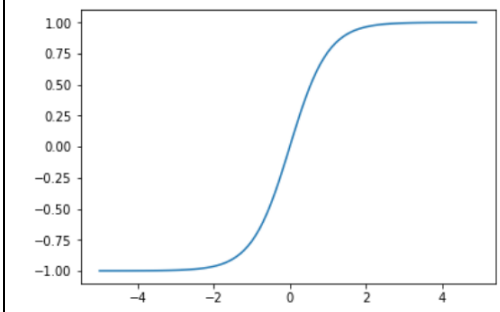
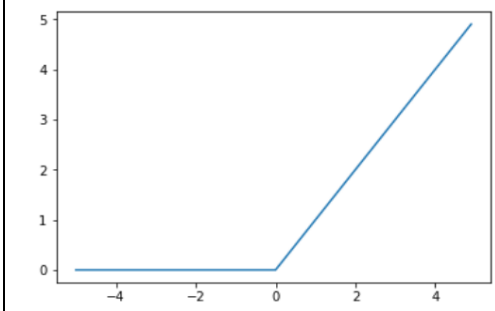
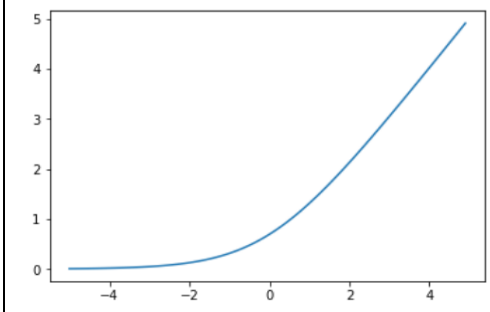
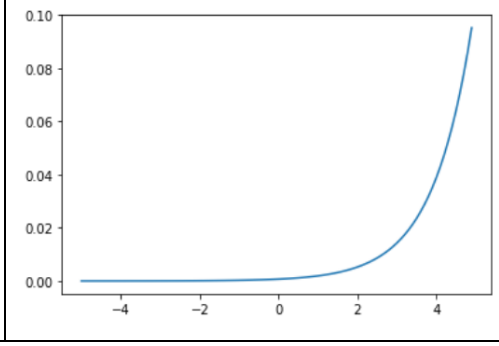
Función de Activación	Representación gráfica	Características prácticas
Sigmoide o Logística: $\sigma(x) = \frac{1}{1 + e^{-x}}$ $\sigma: \mathbb{R} \rightarrow (0,1)$		- Usable en regresiones logísticas. - Permite implementar clasificadores binarios o múltiples.
Tangencial o tangente hiperbólica $\sigma(x) = \tanh(x)$ $\sigma: \mathbb{R} \rightarrow (-1,1)$		- Versión mejorada de la sigmoide o logística. - Genera valores negativos útiles en modelos multicapa.
Rectificador (ReLU) $\sigma(x) = \max(0, x)$ $\sigma: \mathbb{R} \rightarrow [0, +\infty)$		- Es la más utilizada en los redes neuronales convolucionales. - No todas las neuronas se activan lo que permite una velocidad de aprendizaje mayor.
Rectificador Suavizado o *SoftPlus* $\sigma(x) = \ln(1 + e^{x})$ $\sigma: \mathbb{R} \rightarrow (0, +\infty)$		- Mismos uso que RELU y sufre de menor saturación promedia
SoftMax $\sigma(x_i) = \frac{e^{x_i}}{\sum_{k=0}^{n} e^{x_k}}$ $\sigma: \mathbb{R} \rightarrow (0, +\infty)$		- Útil para clasificadores logísticos múltiples. - Extendida para el procesamiento natural de texto o del lenguaje (*NPL*)

Tabla 1. Comparativa de las funciones de activación (Diseño propio).

3.6. Aprendizaje

Los seres humanos comenzamos nuestra fase de aprendizaje desde el nacimiento y no finaliza hasta nuestra propia muerte, aprendemos mediante el ensayo y el error. Requerimos de la observación de una acción, conducta o comportamiento de los demás para aprender y también de la propia imitación ya que es esta la que configura nuestra experiencia.

El neurobiólogo Giacomo Rizzolatti descubrió mediante un estudio realizado en 1996 , que se podía afirmar que disponemos de un tipo especial de células llamadas "neurona espejo" o "neuronas especulares" las cuales se relacionan con los comportamientos sociales e imitativos. La enseñanza que recibimos de las experiencias que vemos e imitamos hacen posible este aprendizaje. El aprendizaje en términos neurológicos se traduce en una reestructuraciones de las conexiones neuronales.

En concreto la estructura y las conexiones entre neuronas no son estáticas sino que pueden sufrir modificaciones a lo largo de su actividad comunicativa con otras neuronas que se realiza mediante la sinapsis. Algunas conexiones se debilitarán y otras se potenciarán, estos cambios son debidos a la plasticidad sináptica la cual es esencial para el aprendizaje.

Encontramos este equivalencia entre el aprendizaje humano y el que realizan las RNA mediante el reajuste del valor de los pesos sinápticos w_i . A medida que la red procesa nuevos conjuntos de datos para obtener una respuesta se continúa a su vez el proceso de aprendizaje.

> *[…] el aprendizaje consistiría en realizar un cambio aleatorio de los valores de los pesos y determinar la energía de la red […]. Si la energía es menor después del cambio, es decir si el comportamiento de la red se acerca al deseado, se acepta el cambio. Si por el contrario la energía no es menor, se aceptaría el cambio en función de una determinada y preestablecida distribución de probabilidades (R. Hilera José y J.Martinez Victor, 1995, p. 69)*

El aprendizaje requiere de un conjunto de datos preparado previamente para facilitárselo a la red neuronal. Una vez se dispone de un conjunto de datos (*dataset*) asociado al problema a resolver este se divide en dos nuevos conjuntos: conjunto de entrenamiento (*training set*) y conjunto de validación (*test set*).

Existen multitud de formas distintas de escoger estos conjuntos destinados a servir como entrenamiento y validación de la red, pero independientemente de la técnica escogida el conjunto de aprendizaje debe ser lo más heterogéneos y representativos posible de la población de datos. Esto se debe a que los conjuntos con los que se valida que el aprendizaje se ha completado con éxito serán independientes y en ocasiones notablemente distintos a los utilizados durante el entrenamiento.

El aprendizaje por tanto se puede dividir en dos fases:

1. **Fase de entrenamiento**

   Previamente se construye la red neuronal determinando para ello su arquitectura y sus pesos ya sean estos aleatorios, nulos o se establezcan manualmente. Tras esto, la red comienza un proceso de entrenamiento mediante el procesamiento de un conjunto de datos destinados a ello. Este entrenamiento consiste en ir

ajustando el valor de los pesos sinápticos iniciales a medida que se procesan los datos de dicho conjunto para lograr que la red sea capaz de resolver o dar solución a un problema, para ello se suele hacer uso de la técnicas de la programación del error hacia atrás (*backpropagation*).

Existen diversas reglas que marcan el tipo de entrenamiento según la forma de ajustar o actualizar el valor de los pesos. Las reglas vienen marcadas por la presencia o ausencia de un factor externo que permite controlar el proceso de aprendizaje. Según sea este entrenamiento podemos distinguir entre:

- **Aprendizaje supervisado**: Este aprendizaje se distingue por contar con un agente que permite controlar o "supervisar" el aprendizaje. En el conjunto de datos de entrenamiento se dispone de una serie de atributos que se pasan a la red para que los procese (*inputs*) y otro atributo más que es la propia salida o solución al problema (*expected output*).

  Una vez que la red procesa los datos de entrada (inputs) se obtiene una serie de salidas (*generated outputs o* $\hat{Y}_i$) . Estas salidas se comparan con las salidas conocidas (*expected output o* Y_i) para estimar mediante una función, el error cometido. Además también se ajusta proporcionalmente el valor de los pesos (Véase la función del error cuadrático medio):

  $$ECM = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

  En base a estos ajustes se espera que el error obtenido para los nuevos datos procesados durante el entrenamiento se reduzca poco a poco.

- **Aprendizaje no supervisado**: A diferencia del aprendizaje supervisado en este no se dispone de la salida asociada (*expected output*) a cada registro de del conjunto de datos de entrenamiento. La red debe encontrar patrones (correlaciones, peculiaridades, similitudes en las distribuciones…) en los datos que permitan ajustar los valores de los pesos para distinguir así los distintos tipos de datos que componen el conjunto de datos.

  La RNA cuyo aprendizaje sea no supervisado puede realizar por ejemplo un proceso de *clustering* (categorización por grupos) de los datos, en base a la similitud que muestren los diversos grupos de datos existentes.

- **Aprendizaje por refuerzo**: Este aprendizaje es considerado por algunos expertos como un subtipo dentro del aprendizaje supervisado. En él se recogen características de los dos aprendizajes anteriores. La red procesa los datos de entrada del conjunto de entrenamiento y tras ello no se compara la salida estimada con la esperada, simplemente se le hace saber a la red si la salida que esta ha generado es o no correcta.

2. Fase de validación o evaluación

Completado el entrenamiento, la red está preparada para procesar los registros de conjunto de datos de validación originando para cada uno su correspondiente salida. En el caso de realizar un proceso de aprendizaje no supervisado la salida estimada no podrá validarse o ser contrastada, ya que es la red la que genera esa información o conocimiento a partir de los registros utilizados.

La finalidad de la validación no es otra que evaluar si la red realmente ha "aprendido" a generalizar a través de los ejemplos utilizados o si simplemente, ha encontrado patrones para obtener las salidas lo más precisas posible para datos de entrenamiento. Para comprobarlo se hace uso del conjunto de validación, el cual presenta registros que la red no conoce, es decir, registros que no han sido procesados o tratados durante la fase de entrenamiento.

El "error de aprendizaje" y el "error de generalización" obtenidos durante las fases de aprendizaje y de evaluación se comparan. Por lo general podemos dar por satisfactoria esta fase si el error de la generalización es inferior al de aprendizaje, ya que ello será indicativo de que el sistema ha encontrado correctamente las relaciones subyacentes en los datos *(Florez López, Fernandez Fernandez, 2008, pag 38).*

Resulta conveniente intentar evitar que se produzca un aprendizaje excesivo, lo cual puede parecer contradictorio pues una red muy entrenada debe ser capaz de generar mejores predicciones pero esto no ocurre siempre así. Como se aprecia en la siguiente figura 12, a medida que más se entrena la red mejora y el error de generalización baja, aunque esto ocurre hasta cierto punto. La red deja de ser capaz de generalizar los conocimientos del aprendizaje (*overtraining*), se ajusta a las peculiaridades y los patrones encontrados en el conjunto de aprendizaje. Reacciona generando unas salidas cada vez más distintas a las esperadas cuando los datos de validación dejan de parecerse a los de entrenamiento.

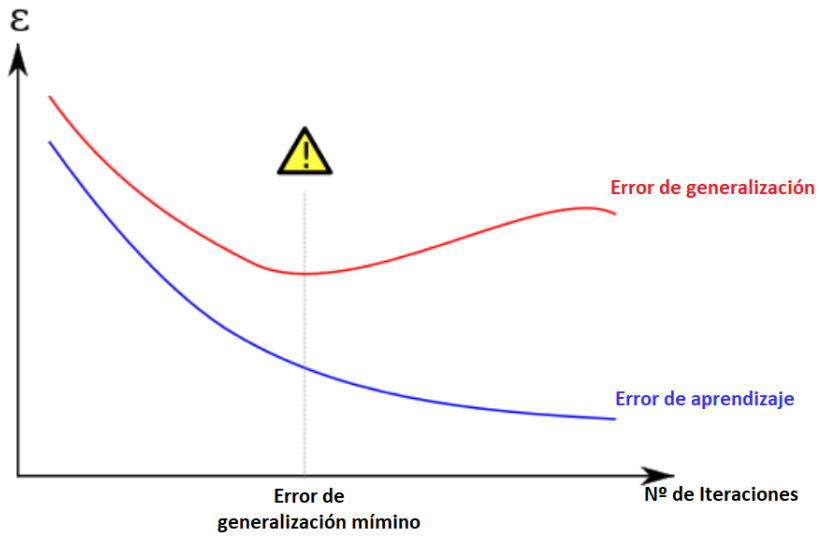


Figura 12. El problema del overtrainning durante el entrenamiento de una RNA *(Diseño propio a partir de Gringer, 2007).*

El proceso de aprendizaje es un proceso iterativo sobre el conjunto de datos de entrenamiento que no siempre requiere procesarlos todos un alto número de veces, esto puede dar lugar a un sobre-entrenamiento. Con objetivo de evitarlo se hace uso de técnicas como la validación cruzada (*cross-validation*) la cual garantiza que las estadísticas obtenidas sobre los resultados son independientes de los conjuntos de entrenamiento y de prueba.

Para evitar utilizar el procesamiento completo de datos durante el entrenamiento es posible establecer ciertos criterios que den el entrenamiento por completado, estos criterios se denominan "criterios de parada". Entre los criterios más utilizados destacar:

- Cuando se produzcan varios ajustes del valor de los pesos y estos no conlleven a una reducción significativa del error se puede suponer que la red no aprenderá más como para mejorar la precisión de sus resultados estimados.

- El hecho de marcar un valor umbral de error que permita comparar el obtenido a medida que se realiza el entrenamiento y cuando este sea inferior al umbral se considere que la red está suficientemente entrenada.

3.7. Modelos de red multicapa

El potencial real de las RNA se obtiene cuando el diseño de arquitecturas de red comienza a ser más complejo que el de una red formada por un único perceptrón solucionando así las limitaciones del mismo. Este tipo de redes al igual que las complejas redes de neuronas que conforman el cerebro, se componen de muchas neuronas simples conectadas entre sí.

La representación formal más extendida de los modelos de RNA se realiza mediante el uso del concepto de conexiones de un grafo. Los grafos son una forma de representación diseñada por el matemático, físico y filósofo Leonhard Paul Euler en el año 1736 como forma de dar solución al problema matemático de los puentes de Königsberg. En los grafos de las RNA contamos con diversos vértices (perceptrones), generalmente redondos y conectados en un solo sentido hacia otros vértices mediante aristas o flechas que representan las sinapsis entre las neuronas.

A partir del concepto del grafo, el diseño se basa en conexiones "hacia delante" (*feedforward*) en las que contamos con una estructura de capas compuestas por 1 a N perceptrones, siendo N libre para cada capa. La primera es la capa inicial (*inner layer*) cuyos perceptrones se encuentran conectados exclusivamente a los perceptrones de la siguiente capa pudiendo ser una capa intermedia o capa oculta (*hidden layer*) denominada de esta forma solo si no corresponde a la capa de salida (*output layer*). La forma de conectar capas se repite en toda la red hasta llegar a la capa de salida (*output layer*).

Para comprender de forma visual lo mencionado hasta el momento véase la siguiente figura. En ella se detallan los elementos de la arquitectura de una RNA más compleja a la vista anteriormente en el caso del perceptrón simple.

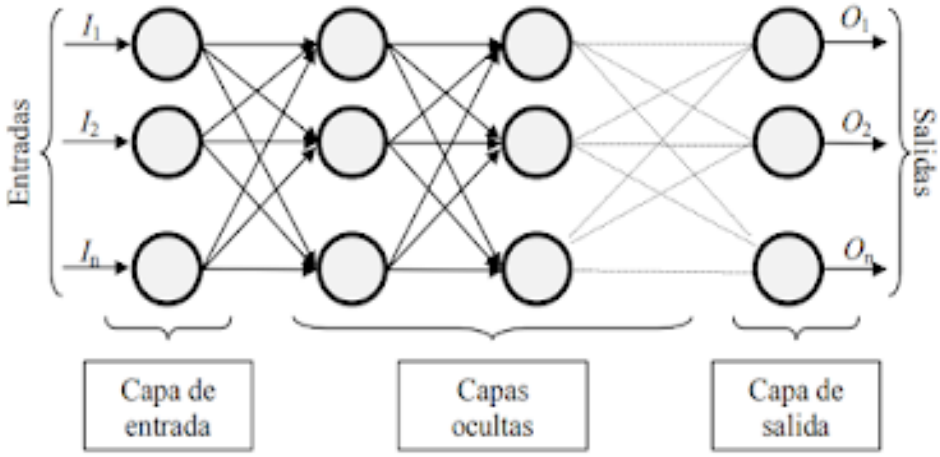


Figura 13. Esquema de la arquitectura de una RNA *(Salcedo, 2018).*

Destacar que lo más común es que todas las neuronas de la capa anterior estén conectadas a todas las neuronas de la siguiente capa, si se cumple esto para toda la red, se dice que la red está completamente conectada (*fully connected*). Además no es necesario incluir en la representación la entrada del sesgo al perceptrón o las funciones de activación, se dan por hecho que existen y que eso se detalla más en la descripción formal de la arquitectura utilizada para el diseño de la RNA.

A diferencia del esquema del perceptrón simple en el que la capa de entrada era la propia capa de salida y la salida era siempre un único valor generado por la función de activación, las RNA permiten que la capa de salida esté compuesta por más de un perceptrón. La principal ventaja que supone es la capacidad de construir clasificadores de clases múltiples ya sean exclusivos o no exclusivos.

La generalización matemática del recorrido que hacen los datos desde que su entrada a la red hasta obtener las salidas de la red, pasa por el cálculo de las salidas de cada perceptrón de las distintas capas, ya que cada uno de estos depende de las entradas recibidas. Quedando la fórmula de la capa de salida de la siguiente forma:

$$y_i = \sigma_i \left(\sum_{j=o}^{n} w_{ij} s_{ij} - \theta_i \right)$$

los elementos que componen la ecuación al igual que subíndices lo siguiente:

- y_i es la salida obtenida para el perceptrón i de la capa de salida.
- σ_i es la función de activación de cada perceptrón de la capa de salida.
- w_{ij} es el peso sináptico que recibe la sinapsis que conecta al perceptrón emisor j de la capa anterior con el perceptrón i.
- s_{ij} es el valor que envía el perceptrón emisor j de la capa anterior al el perceptrón i el cual se verá afectado por su correspondiente peso w_{ij}.
- θ_i es el valor del sesgo que afecta al perceptrón i.

4. Redes convolucionales

Las redes neuronales basadas en operaciones de convolución o también denominadas CNN (*Convolutional Neural Networks*) son redes neuronales especializadas en el procesamiento de imágenes. Estas siguen el esquema del funcionamiento de las neuronas biológicas que se encuentran en la corteza visual del cerebro. El objetivo de estas redes es resolver problemas de reconocimiento y clasificación, problemas en los cuales los datos de entrada vienen dados por imágenes. Los ficheros de imágenes gracias a los avances tecnológicos en las cámaras digitales han sufrido un aumento su tamaño en disco, este aumento ha complicado las tareas a los métodos clásicos.

Los resultados que obtienen este tipo de redes en comparación con métodos tradicionales han conseguido que dentro de la IA el campo visión por computador está de nuevo en auge. Esto se debe a sus numerosas aplicaciones prácticas (reconocimiento facial, reconocimiento de señales de tráfico para conducción autónoma, clasificadores de objetos…).

El científico de la computación francés llamado Yann LeCun desarrolló en 1998 una red neuronal conocida por el nombre de LeNet-5 capaz de realizar el reconocimiento de dígitos en una imagen digitalizada (véase figura 14). Esta red tuvo una gran aceptación y fueron diversos los bancos los que la utilizaron para el reconocimiento numérico de los dígitos de los cheques que venían procesando manualmente.

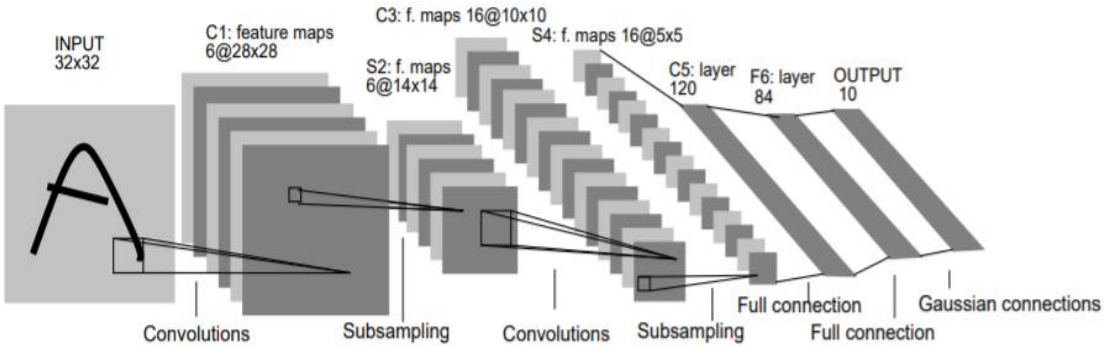


Figura 14. Esquema de la arquitectura de LeNet-5 para reconocimiento de caracteres *(LeCun Yan, 1998, fig. 2, p. 2).*

Esta red ya entrenada procesa una imagen como entrada, la imagen sufre de diversas transformaciones a lo largo de las distintas capas ocultas de la red hasta la llegada a la capa de salida, en esta última se dispone de 10 salidas una para cada uno de los dígitos numéricos (0-9). Cada salida ofrece un valor real que indica la probabilidad de pertenencia de a cada dígito gracias a una activación *softmax*, la salida con el valor más alto será la que indique el dígito al que pertenece.

La arquitectura de LeNet-5 ha servido de modelo para posteriores arquitecturas de CNN como la diseñada por Alex Krizhevsky en el año 2012 cuyo trabajo de investigación es uno de los más relevantes a día de hoy. La importancia de su red no viene tanto por su complejidad sino por su eficiencia, pues a pesar de trabajar con *ImageNet*, una de las bases de datos con mayor número de imágenes en alta resolución, la red hace uso de la potencia de las unidades de procesamiento gráficas (*GPU*). Logrando así que un entrenamiento con más de un millón de imágenes sea posible en tiempos más reducidos.

De forma resumida una CNN se compone de distintas operaciones a lo largo de las capas. Como las que podemos observar en la siguiente tabla que describe la arquitectura de la red de LeNet-5. Las operaciones persiguen reducir el tamaño del dato de entrada (imagen original) procesando a lo largo de las capas matrices de datos cuya información resume las características de la entrada original facilitando así el cómputo y el aprendizaje.

Capa		Tamaño del dato transmitido	Tamaño núcleo	Función de activación
Entrada	Imagen	32x32	-	-
1	Convolución	28x28	5x5	Tangencial
2	*Pooling*	14x14	2x2	Tangencial
3	Convolución	10x10	5x5	Tangencial
4	*Pooling*	5x5	2x2	Tangencial
5	Convolución	1x1	5x5	Tangencial
6	Clasificación	84	-	Tangencial
Salida	Clasificación	10	-	*Softmax*

Tabla 2. Resumen de la arquitectura de LeNet-5 *(Diseño propio).*

4.1. La imagen digital

La primera imagen digital registrada no proviene de la que se consideró como la primera cámara digital de la historia, la cual fue inventada por la empresa Kodak en el año 1975, esta imagen fue originada por un escáner. El ingeniero Russell A. Kirsch diseño en 1957 un escáner digital el cual utilizó por primera vez escáner una fotografía de su hijo recién nacido, generando de esta forma la primera imagen digital con una resolución de 176 x176 pixeles.

Actualmente las imágenes digitales se componen de los metadatos (codificación utilizada, dimensiones, extensión, componentes de color…) y de los datos codificados de la propia imagen cuya representación más extendida es mapa de *bits/pixels*. Este mapa sigue una representación dimensional basada en una matriz numérica la cual puede aceptar distintos rangos de valores según el esquema de colores de la imagen:

- **Blanco-negro** : I(x,y) en el que $x, y \in \{0,1\}$.
- **Escala de grises**: : I(x,y) en el que $x, y \in \{0,1,2, \dots ,255\}$.

Para utilizar una imagen de alguno de los esquemas de anteriores como entrada a la red neuronal dicha imagen se puede procesar por ejemplo como una sola secuencia de datos de entrada haciendo uso de su estructura bidimensional. Se extraen ordenadamente sus valores por columnas y por filas desde arriba hacia abajo formando una larga secuencia numérica, de forma que cada pixel que compone la imagen tiene su propia entrada en la red neuronal (ver figura 15). Esto requiere el reajuste de multitud de pesos, afortunadamente existen otras técnicas de tratamiento previos sobre la imagen para reducir el procesamiento que realice la red.

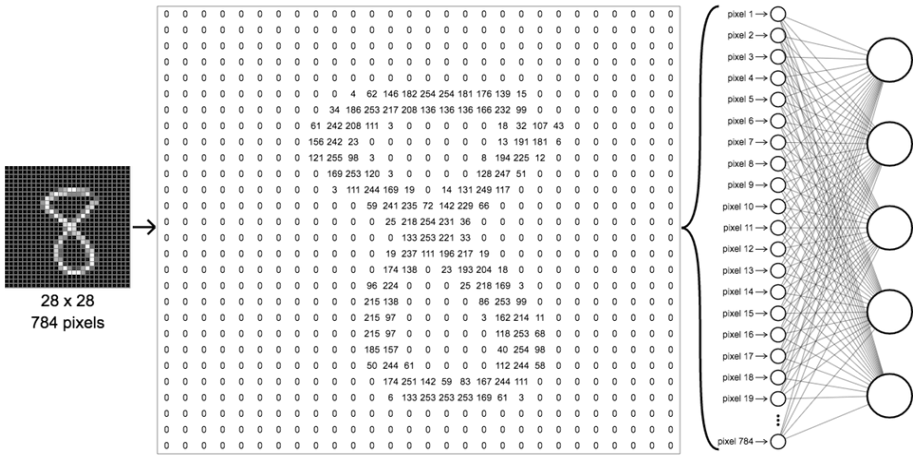


Figura 15. Representación de los píxeles como entrada a una red neuronal *[Train a MobileNet Wihthing 60 lines of Code] (2019).*

- **Color**: I (x,y,c) en el que $x, y \in \{0,1,2 \dots 255\} \wedge c \in \{R, G, B\}$

  Las imágenes en color se componen de tres canales de información tridimensionales (*Channel last representation*), a diferencia de los dos anteriores este tipo de esquema no trata las imágenes como matrices sino como tensores. La siguiente figura muestra cada una de dimensiones del tensor: R(*Red*) para el rojo, G(*Green*) para el verde y el B(*Blue*) para el azul.

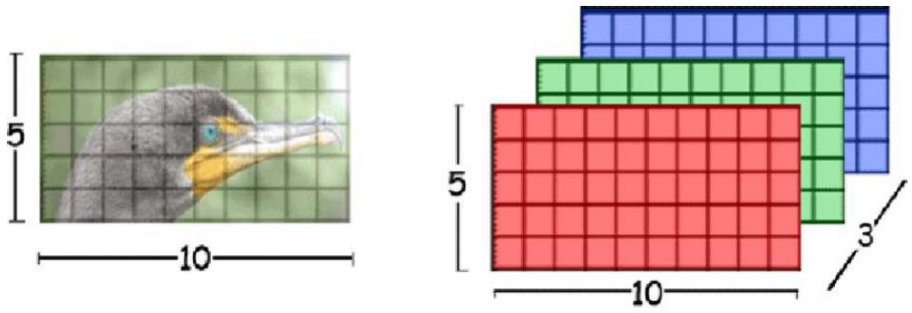


Figura 16. Imagen RGB como tensor de dimensión (5,10,3) *(González Jiménez, 2016, p.10).*

4.2. Convolución

Una de las formas que utiliza nuestro cerebro internamente para saber si un objeto que ya ha visualizado es igual o distinto a otro que ha visualizando previamente, es mediante el reconocimiento y la comparación de las características o patrones identificativos de este segundo sobre los que aprendió del primero.

En términos de imagen digital las operaciones de convolución tienen como objetivo la búsqueda de esas características o patrones que se encuentran en la propia imagen. Dichas características pueden ser secuencias de valores de píxeles que por lo general se repiten ya sea en vertical, horizontal o en diagonal, definiendo de esta forma líneas verticales, bordes, zonas con cambios marcados etc.

Para realizar la convolución se requiere de dos elementos principales: una imagen y un *kernel* (núcleo o máscara). El núcleo k es una matriz cuyos valores permiten encontrar si se da un determinado patrón en la imagen. Este núcleo se opera a lo largo de la matriz de valores I de la imagen siendo el sentido de izquierda a derecha y de arriba hacia abajo según el paso (*stride*) marcado para generar así una nueva matriz de características como resultado de la convolución.

La forma matemática de expresar la convolución es la siguiente:

$$Y[m,n] = k * I[n,m] = \sum_{j=-a}^{a} \sum_{i=-b}^{b} I[i,j]k[m-i,n-j]$$

los elementos que componen la ecuación al igual que subíndices lo siguiente:

- Y es la matriz asociada a la imagen obtenida tras la convolución (I*k).
- k es el núcleo de la operación de convolución.
- x es la matriz asociada a la imagen inicial sin procesar.
- Cada elemento del núcleo se considera como $-a \leq j \leq a \ \wedge -b \leq i \leq b$

De esta forma la operación consiste en el sumatorio de los valores obtenidos tras el producto de cada uno de los índices con la submatriz de I en la que se realice la operación en el orden indicado anteriormente, véase un ejemplo mediante la siguiente figura.

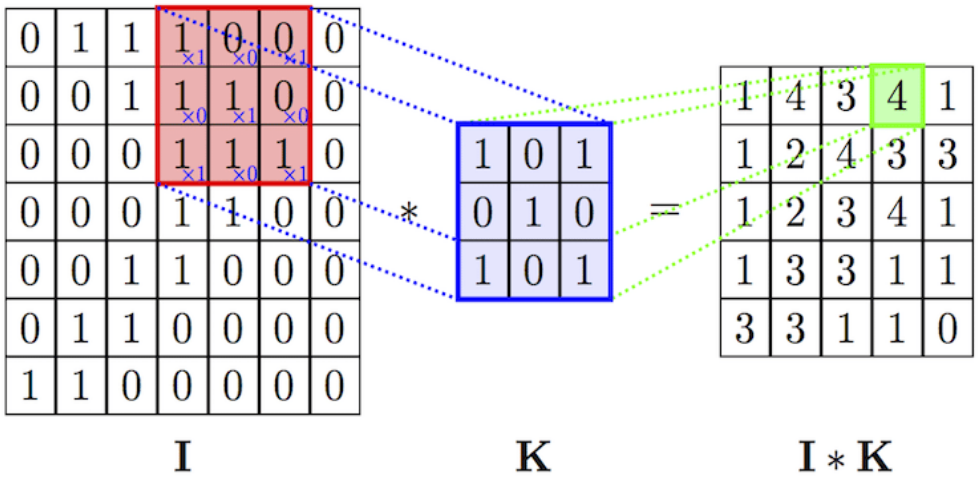


Figura 17. Imagen de una convolución completa(5,10,3) (Portilla, J., Francesco, M., 2018).

Según el caso es conveniente normalizar el resultado de ese sumatorio obtenido, para ello tan solo es necesario dividir el resultado por el número de elementos que componen el núcleo k. Como se aprecia en la figura 17, la convolución se expresa generalmente con el símbolo ⊗ , una vez realizada se obtiene la matriz de la izquierda. La matriz generada dispone de sus valores normalizados, destacando que a medida que más cercano es a 1 sea el valor obtenido mayor es la similitud con el núcleo y por tanto con la presencia de ese patrón o característica buscada en la submatriz con la que se ha realizado la convolución.

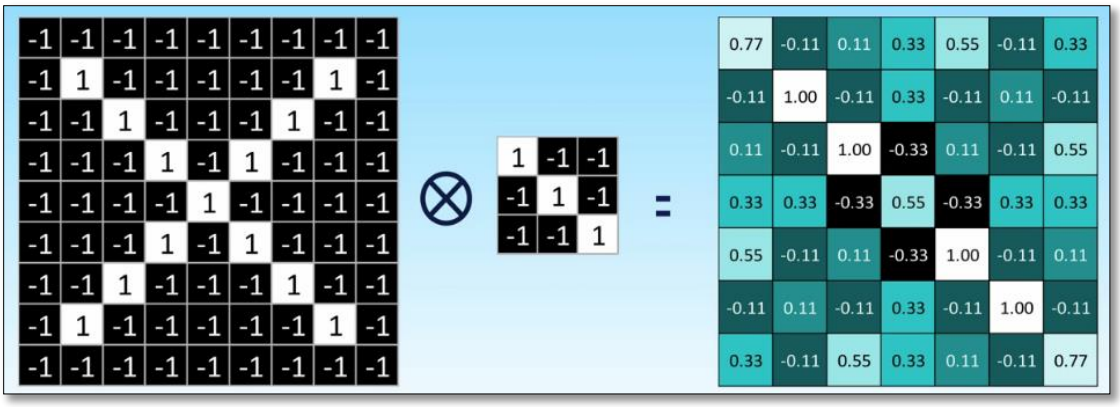


Figura 18. Imagen de una convolución completa normalizada (Portilla, J., Francesco, M., 2018).

Véase por último la siguiente tabla con distintos ejemplos, en ellos se pretende encontrar distintos patrones o características mediante diferentes núcleos aplicados a una misma imagen:

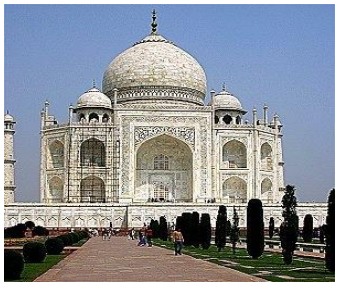
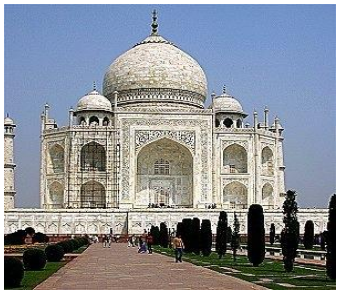
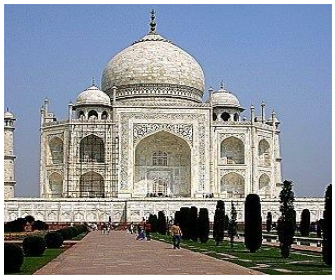
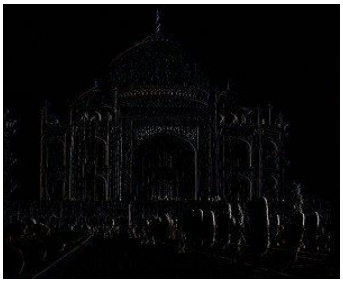
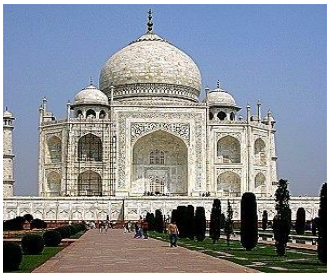
Imagen Original	Núcleo aplicado	Resultado de la convolución
	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ Identidad	
	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ Realzado de bordes	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ Detección de bordes	

Tabla 3. Ejemplos de convoluciones con distintos núcleos (Portilla, J., Francesco, M., 2018).

4.3. Strides

Para obtener las sub-matrices con las que se opera el núcleo basta con desplazar una ventana deslizante a lo largo de la matriz de características, esta ventana se colocada en primera instancia sobre el índice (0,0).

El *stride* o paso $p(fila, columna)$ con el que esta se desplaza puede escogerse. Por defecto se utiliza el (1,1), es decir, la ventana selecciona cada submatriz desplazándose de izquierda a derecha de 1 en uno y una vez llegue al final de la fila comienza nuevamente desde la izquierda desplazándose tan solo una fila hacia abajo.

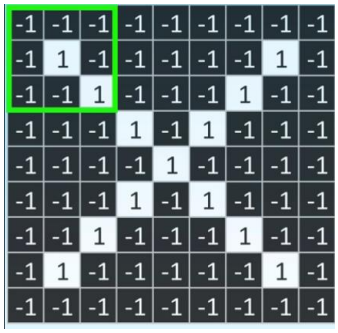
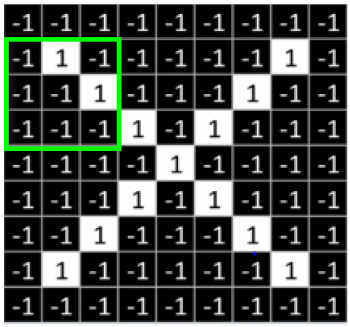
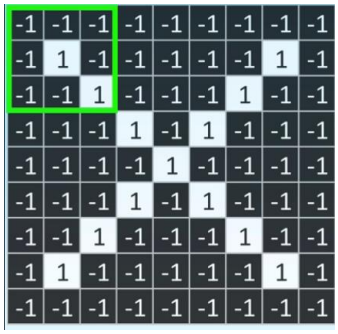
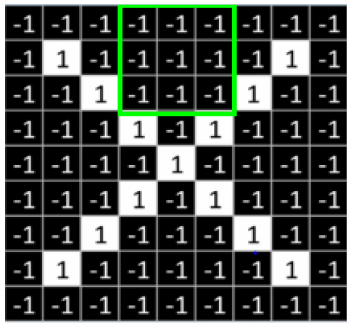
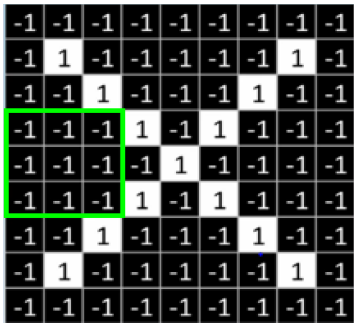
Posición inicial y paso	Desplazamiento por columna	Desplazamiento por filas
 **Paso : (1,1)**		
 **Paso : (3,3)**		

Tabla 4. Ejemplos de convoluciones con un nucleo de 3x3 para distintos pasos (Portilla, J., Francesco, M., 2018).

A mayor sea el paso, menos información se recoge de la imagen pero más fácil es de procesar la obtenida ya que el tamaño de la salida tras la convolución no solo depende del tamaño del núcleo sino que también del paso. En el caso de la matriz de la tabla 4, según el paso la matriz obtenida tras la convolución tendrá distintas dimensiones:

- Núcleo de 3x3 y paso (1,1): 7 filas x 7 columnas.
- Núcleo de 3x3 y paso (3,3): 3 filas x 3 columnas.

4.4. Padding

Según sea de importante preservar los valores de los bordes de la matriz o controlar el tamaño de la matriz resultante tras la convolución, es posible utilizar para ello la técnica del *padding*. Esta sencilla técnica consiste en colocar algún valor generalmente ceros (*zero-padding*) alrededor de los píxeles situados bordes de la matriz de características.

La figura 19 muestra cómo con esta técnica la convolución no comienza en la posición inicial vista en la anterior tabla para esa misma matriz de la imagen. Al aplicarla la matriz obtenida con un núcleo de 3x3 pasa a ser de 9x9 conservando así su tamaño original en lugar de ser 7x7, que es el tamaño correspondiente a la matriz resultante sin aplicar conjuntamente esta técnica.

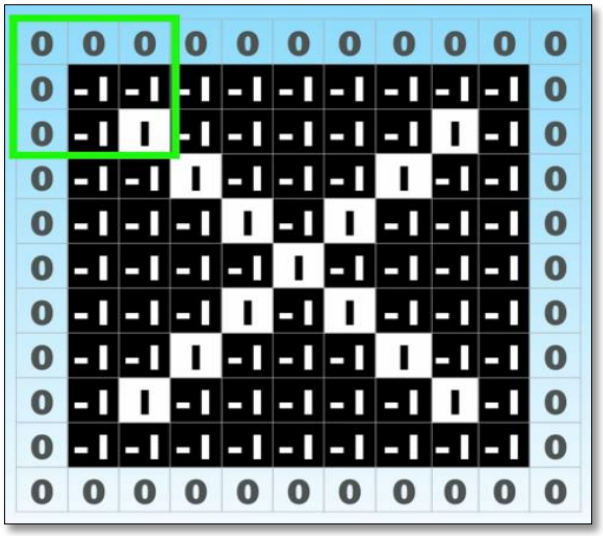


Figura 19. Imagen de una convolución completa (Portilla, J., Francesco, M., 2018).

4.5. Pooling

La técnica de *pooling* o de agrupación de píxeles consiste en reducir la dimensionalidad de la matriz de características mediante la substracción de la información que consideremos más relevante de la matriz gracias al uso de una ventana deslizante.

A diferencia de la convolución en esta técnica no se produce solapamiento durante los desplazamientos de la ventana, la cual es de tamaño libre. La importancia de la misma proviene de su capacidad de hacer a la red menos sensible a imágenes con distorsiones o pequeños cambios ya que solo se conservan por ejemplo los valores de los píxeles más relevantes sin importar los valores de cercanos o los valores promedios de cada zona o región. Permitiendo además disponer de una imagen más manejable y casi a escala de la original.

Véanse a continuación los resultados de aplicar algunos de los distintos tipos de agrupamientos que existen sobre una matriz de características.

Matriz de características	Agrupamiento y tipo
$\begin{bmatrix}1 & 0 & 1 & 1\\1 & 2 & 1 & 0\\4 & 0 & 0 & 0\\0 & 0 & 0 & 8\end{bmatrix}$ **Dimensiones : 4x4**	$\begin{bmatrix}2 & 1\\4 & 8\end{bmatrix}$ **Tipo : Valor máximo**
	$\begin{bmatrix}1 & 1\\1 & 2\end{bmatrix}$ **Tipo :Valor promedio**
	$\begin{bmatrix}4 & 3\\4 & 8\end{bmatrix}$ **Tipo : Suma**

Tabla 5. Aplicación de la técnica de *pooling (Diseño propio).*

4.6. *Flattening*

En los puntos anteriores de este capítulo se han visto técnicas para la simplificación y extracción de las características más importantes de una imagen, técnicas que pueden incorporarse en las capas iniciales de una CNN. Aplicarlas evita que la red tenga que aprender directamente de la imagen inicial completa consiguiendo así que el aprendizaje sea más sencillo y rápido.

Una vez que la imagen sea procesada por estas capas se suele hacer uso de una capa adicional de *flattening* o aplanado. Esta capa se encarga de transformar las matrices de características en un solo vector, de manera que este vector pueda facilitarse ahora a una capa *fully connected* que permitirá comenzar con la fase de clasificación. En la siguiente figura se aprecia el resultado de su aplicación sobre las matrices obtenidas como salida de la última capa que componga la etapa de extracción de características:

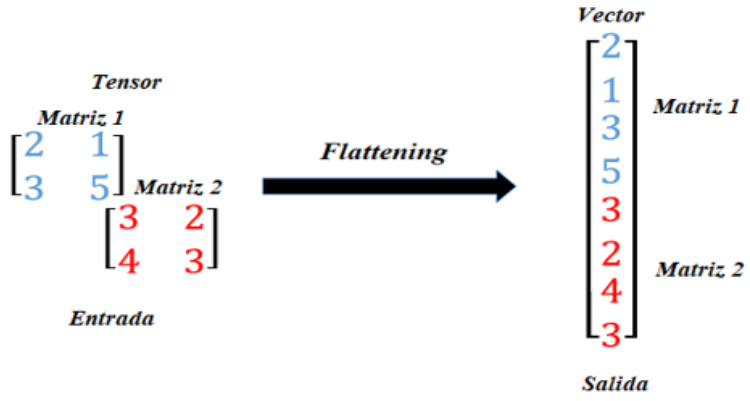


Figura 20. Flattening sobre un tensor de matrices de carácterísticas (Diseño propio).

4.7. Tensores y arquitecturas

La palabra tensor ha aparecido a lo largo de los apartados anteriores en varias ocasiones, esta palabra se utiliza como forma de denominar a la agrupación o generalización de las matrices cuya representación debido a su estructura puede o no, ser representable.

Esta abstracción capaz de generalizar a los escalares, vectores y las matrices hace que estos sean independientes de su marco de referencia o de cualquier sistema de coordenadas.

- Escalares (números): 0, 1, 23, 45655, π , 4/3.
- Vectores (lista de números): [0] , [1, 2, 3], [23, 34, 54, e].
- Matrices (vector bidimensional) : $[5]$, $\begin{bmatrix}1 & 1\\1 & 2\end{bmatrix}$ $\begin{bmatrix}1 & 0 & 1 & 1\\1 & 2 & 1 & 0\\4 & 0 & 0 & 0\\0 & 0 & 0 & 8\end{bmatrix}$.
- Tensor : $\left[[1,2,3], \begin{bmatrix}1 & 1\\1 & 2\end{bmatrix}\right]$, $\left[\begin{bmatrix}1 & 2 & 3\\4 & 4 & 5\\5 & 2 & 3\end{bmatrix}, \begin{bmatrix}6 & 4 & 9\\7 & 1 & 8\\5 & 2 & 2\end{bmatrix}, \begin{bmatrix}9 & 1 & 3\\3 & 4 & 6\\7 & 7 & 0\end{bmatrix}\right]$.

En términos de CNN un tensor se utiliza de igual forma para denominar a la agrupación de distintas matrices. Pueden ser las asociadas a las distintas capas que conforman a una imagen a color o por lo general para la agrupación de las matrices de entrada o salida de cada capa que compone la extracción de características.

Una vez definidos algunos de los diversos conceptos que componen y explican el funcionamiento de las CNN podemos comprender la estructura general de la arquitectura en este tipo de redes neuronales dividiéndola en dos etapas:

1) **Extracción de características**: Se definen las capas a utilizar (convolución, *pooling*, *padding*…) para la extracción de características de la imagen de entrada, seleccionando el número de nodos de cada una e inicializándose con los parámetros y pesos necesarios. A medida que se aumenta el número y tipo de características extraídas de las imágenes utilizadas durante el entrenamiento la red suele mejorar así en su capacidad de clasificación. Esta etapa comienza con la propia imagen como entrada y tras procesarla por las diversas capas se dispone de un tensor compuesto por matrices de características.

2) **Clasificación**: Se comienza con una capa de aplanamiento para generar un único vector a partir del tensor obtenido en etapa anterior, a continuación se propaga este vector a través de una única capa *fully connected* o de una serie de capas apiladas para acabar realizando así un proceso de clasificación mediante la función de activación por ejemplo de *softmax* para conocer la clase a la que pertenece la imagen procesada.

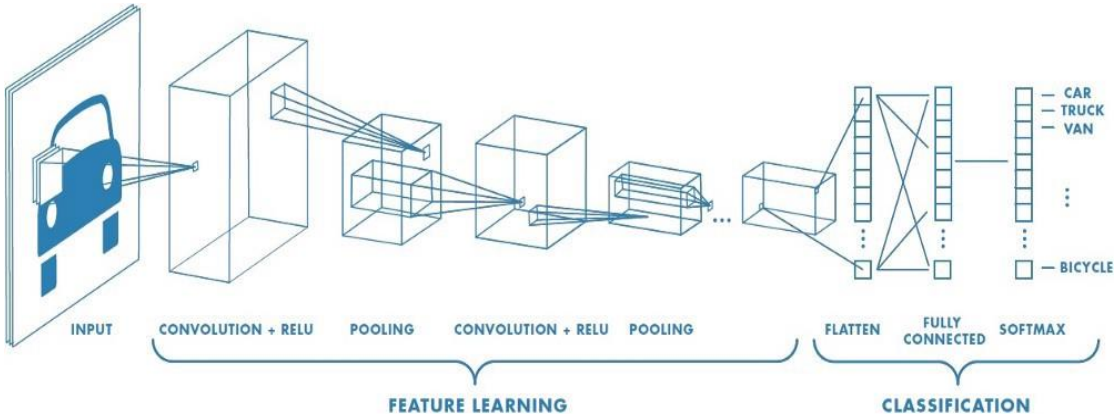


Figura 21. División conceptual de la arquitectura de una CNN *(Prabhu, 2018).*

Utilizar esta arquitectura supone una notable mejoría en términos de rendimiento y aprendizaje. En comparación con la figura 15 en la que se utilizaba una *fully conected* directamente como entrada en la red, si aplicamos las técnicas de la extracción de características previamente sobre la imagen podemos apreciar lo siguiente en esta tabla comparativa:

	Entrada	1º Capa	Aprendizaje	Entrada	2º Capa	Aprendizaje
Sin extracción de características	Imagen 28x28	*Fully conected* con 32 nodos.	25.088 reajustes	- - - - - -	- - - - -	- - - - - - - - -
Con extracción de características	Imagen 28x28	Convolución con 32 nodos (*Núcleo 3x3*)	288 reajustes	Imagen 7x7	*Fully conected de 32 nodos*	1568 reajustes

Tabla 6. Comparativa del aprendizaje con y sin extracción de características de una imagen *(Diseño propio).*

No solo reducimos notablemente el número de pesos a actualizar durante el aprendizaje lo que claramente supone una reducción de tiempos y de memoria, sino que también al extraer la información más característica que ofrece cada imagen evitamos realizar un aprendizaje susceptible de que los ligeros cambios de los valores en los píxeles menos relevantes puedan afectar negativamente a los resultados obtenidos por la clasificación.

5. Estudio del problema de negocio

Las empresas cuyo modelo de negocio consiste en la venta de productos de moda, como los son las prendas y complementos de vestir han visto un notable incremento en el volumen de ventas realizadas a través de internet así como del número de consumidores digitales. Consumidores que pasan a combinar sus compras presenciales con compras a través de las plataformas proporcionadas por las empresas o a únicamente realizarlas por esta segunda vía.

Uno de los principales problemas de negocio que origina esta tendencia en las empresas que disponen de *e-commerce* y que se ha mencionado anteriormente es la incapacidad real de las empresas para conocer a sus clientes virtuales. La actividad de los usuarios en internet es masiva, no siempre rastreable y sobre todo es evolutiva y cambiante. "*El mercado cambia, el gusto cambia, por tanto las compañías e individuos que eligen competir en esos mercados deben cambiar" – An Wang (Bello, C., s.f., p.58).*

La necesidad de solucionar este problema surge entre otro motivos, con el objeto de poder realizar recomendaciones de los productos más adecuados a sus gustos, sus necesidades o a su comportamiento. De esta forma al visualizar dichas recomendaciones, surgirá en los clientes la necesidad de adquirir también alguno de los productos que les sean mostrados.

En este trabajo se propone crear una posible ventaja competitiva para una empresa del sector, ventaja que a su vez de solución al problema anterior o mejore su actual sistema y permita mejorar la experiencia de los clientes a la vez que la facturación de la empresa. Esta solución permitirá realizar recomendaciones, principalmente gracias a la utilidad que proporcionan las CNN en términos de clasificación de imágenes, ya que en este negocio predomina la información almacenada con formatos de imagen.

5.1. La revolución del comercio electrónico

El comercio tradicional se remota a la etapa del Neolítico, momento en el que surge la agricultura. Según la Real Academia Española el comercio se define como la "compraventa o intercambio de bienes o servicios" (Real Academia Española [RAE], 2019). La compra o intercambio requiere del acercamiento físico entre el vendedor y el comprador, pero con la llegada de internet surgieron nuevas formas de comercio sin la necesidad de ese contacto o presencia física.

Esta nueva forma de comerciar en la que se prescinde de la presencia física de ambas partes se denomina comercio electrónico:

> *El comercio electrónico o e-commerce es cualquier tipo de operación comercial en la que la transacción se realiza mediante algún sistema de comunicación electrónico, por lo que no se requiere el "contacto físico" entre comprador y vendedor* (Seoane Balado, 2005, p. 1).

Las empresas de productos de moda adquieren sus productos a otras empresas distribuidoras (B2B – *Business to Business*) ya sea bien los propios productos ya fabricados o los materiales que se necesitan para fabricarlas. Por otro lado una vez que las empresas disponen de los productos en sus almacenes, se prepara su puesta en venta de forma física o

digital. Los consumidores digitales puede formalizan la adquisición de estos artículos mediante una aplicación móvil o a través de la página web de la empresa o distribuidor (B2C – *Business to Client*). Para conocer a este nuevo consumidor las empresas han optado por utilizar la inteligencia artificial como forma de estimar por ejemplo su CLTV (valor del tiempo de vida del cliente) o poder ofrecerle mejores recomendaciones de productos para fomentar la compra.

Las aportaciones de la inteligencia artificial se han consolidado como la verdadera revolución en el comercio electrónico durante estos últimos años más allá de las mejoras de diseño, interactividad de las aplicaciones o de los sitios web o de la propia facilidad de acceso y compra por internet.

5.2. Las recomendaciones de moda

Cada vez las empresas se preocupan en mayor medida por las recomendaciones a la hora de ofertar de otros productos durante el proceso de exploración de artículos que realizan los usuarios, previamente a la compra. Este énfasis se denota con las nuevas soluciones basadas en sistemas de recomendación híbridos, sistemas basados en la actividad o los que funcionan mediante imágenes que los usuarios facilitan a la aplicación o sitio web.

La gran innovación en términos de inteligencia artificial para mejorar las recomendaciones en muchas de las empresas ha pasado por la implantación de modelos basados en redes neuronales. La siguiente tabla muestra un posible ejemplo de las recomendaciones que ofrecen los sistemas clásicos, en ella se aprecia que solo se tienen en cuenta por ejemplo el tipo de producto (pantalón). En contraste las recomendaciones que pueden ofrecerse gracias al uso de redes neuronales son notablemente mejores por el hecho de haber detectado más detalles como lo son longitud y el color.

Recomendación aleatoria por categoría	Recomendación con red neuronal
	

Tabla 7. Comparativa de recomendaciones (Diseño propio a partir Miller, 2018).

Para realizar mejores recomendaciones es necesario contar con toda la información posible de los productos. El uso de las redes neuronales permite generar de forma automática las etiquetas a aquellos productos que no dispongan de estas, ya sea porque el fabricante no las facilita, porque no concuerdan con las que trabaja la empresa o porque se hayan perdido durante las fases de ETL (extracción, transformación y carga) de los datos a las bases de datos de productos.

Producto	Tipo	Uso/Ocasión	Patrón	…
A	Vestido	Formal	Floral	…
B	Pantalón	Casual	?	…
C	Chaqueta	?	Lunares	…
D	Pantalón	Etiqueta	?	….
E	Camiseta	Casual	Liso	…
…	…	…	…	…

Tabla 8. Matriz de atributos de los productos de una empresa con cierta información desconocida (Diseño propio a partir de Cardoso, Daolio, Vargas, 2018, p.81).

El tipo de red neuronal más utilizada en este ámbito corresponde a las CNN, estas redes son capaces de aprender la configuración más adecuada para identificar así el valor correspondiente a algún atributo de la prenda mediante un entrenamiento previo con un conjunto de fotografías ya etiquetadas.

Este aprendizaje muestra además una gran notable mejoría si en lugar de entrenarse la CNN únicamente con las imágenes, se le incluye a cada imagen la propia descripción disponible del producto. Observar la siguiente tabla en la que se muestra una posible entrada a una red neuronal, en ella se combina como entrada la imagen y la descripción del producto para que la red genere los atributos del mismo:

Composición de cada entrada de la red neuronal		Salida de la red neuronal
Imágenes del producto	Descripción del producto	Atributos del producto
	**ASOS Mini Tea Dress In Yellow Floral Print** Dress by ASOS Collection • Floral print fabric • Crew neck • Button-keyhole back • Fit-and-flare style • Regular fit - true to size • Machine wash • 94% Viscose, 6% Elastane • Our model wears a UK 8/EU 36/US 4 and is 176 cm/5'9.5" tall	**DRESS TYPE** casual dresses **PATTERN** floral **BACK TO SCHOOL/COLLEGE** no **CUSTOMER SEGMENT** girly girl **USE/OCCASION** day casual **STYLE** tea dresses **SEASONAL EVENT** high summer **RANGE** main collection **PRICE RANGE** high street **CATEGORY** day dresses - jersey print **LENGTH** mini ...

Tabla 9. Ejemplo en alto nivel de una instancia de entrada y salida de una red neuronal (Diseño propio a partir de Cardoso, Daolio, Vargas, 2018, p.82).

6. Planificación, análisis y diseño

A lo largo los dos próximos capítulos se detallará paso a paso la elaboración de la solución propuesta para el problema de negocio mencionado anteriormente. Se creará así un sistema que permita responder a peticiones sobre posibles recomendaciones para el producto que aparezca en una imagen recibida.

La metodología de trabajo se basa la división en las fases propias del ciclo de vida clásico del software ("Cascada"). En este capítulo se comenzará por la fase de ***Planificación*** , se diseñará así una planificación temporal de las posibles tareas que compondrán el proyecto completo mediante un diagrama de *Gantt*.

En la fase de ***Análisis*** se plasmarán los distintos casos de uso del sistema, permitiendo obtener así los requisitos mínimos que este debe cumplir. Conocidos estos requisitos se crearán en la fase de ***Diseño*** los bocetos previos al desarrollo de la aplicación, permitiendo orientar la interfaz gráfica durante su desarrollo. Por último en esta fase se gestionarán las fuentes de datos disponibles para el funcionamiento del sistema.

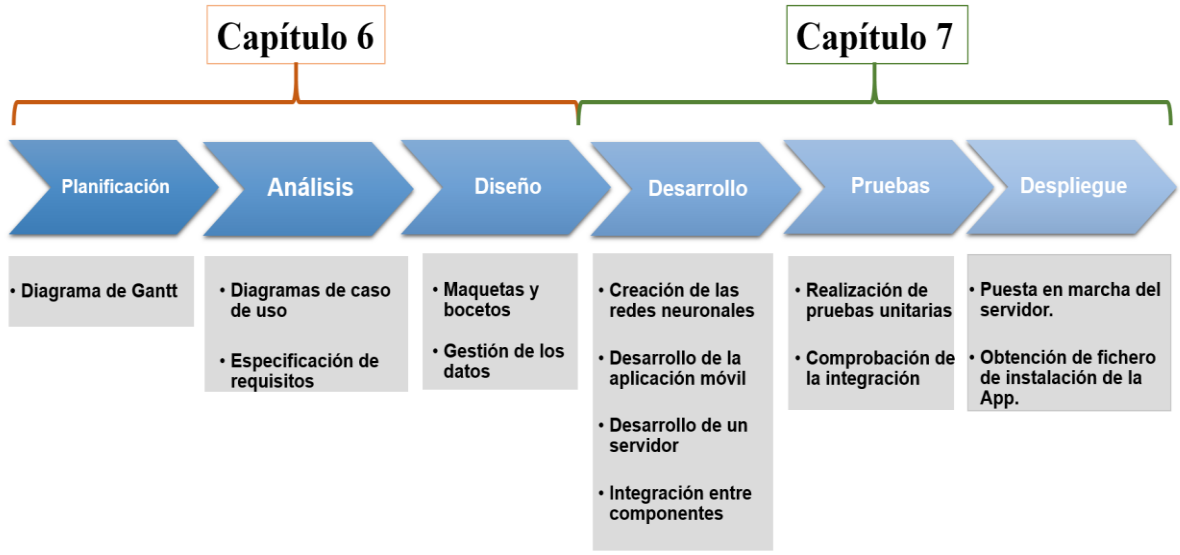


Figura 22. Componentes de cada una de las fases del ciclo de vida (Diseño propio).

6.1. Planificación temporal del proyecto

Antes de comenzar cualquier proyecto de desarrollo software hay que tener claro que la tarea de completar el proyecto debe dividirse en sub-tareas más pequeñas, cada una de las tareas resultantes a su vez puede subdividirse en tareas más pequeñas. Esta división de las tareas permite organizar a los distintos miembros o equipos para realizar aquellas tareas en las que están más especializados para que solo se centren en ellas, las saquen adelante con el mayor éxito y en el menor tiempo posible.

A cada tarea se le asignará un plazo de realización (horas, días, semanas, meses…) a la vez que un número estimado de horas de trabajo. La suma de las horas de todas las tareas del

proyecto permite calcular el coste económico del equipo de trabajo, siempre que todo vaya dentro de plazo.

> *"La gestión del proyecto se puede definir como una forma de desarrollar la estructura en un proyecto complejo, donde las variables independientes de tiempo, costo, recursos y comportamiento humano se juntan" – Roby Burke (Aston, 2019)*

Atendiendo a este trabajo solo una persona es la encargada de la realización de la completitud de las tareas y el periodo de trabajo comenzará a partir del mes de Junio del año 2019, con fin en el mes de Septiembre de ese mismo año. Teniendo en cuenta ese periodo de tiempo las tareas serán divididas en varios días o semanas, de lo contrario una división demasiado detallada en horas o días únicos que se vea afectada por algún retraso obligaría a volver a planificar todo del proyecto.

La planificación y organización temporal de las tareas del proyecto puede plasmarse en un diagrama conocido como diagrama de Gantt. Este diagrama permite exponer de forma visual las tareas del proyecto en un eje temporal, sin prestar demasiada atención a las relaciones entre las propias tareas. La elaboración del diagrama se ha realizado con la herramienta mencionada en el capítulo dos, **GanttProject**.

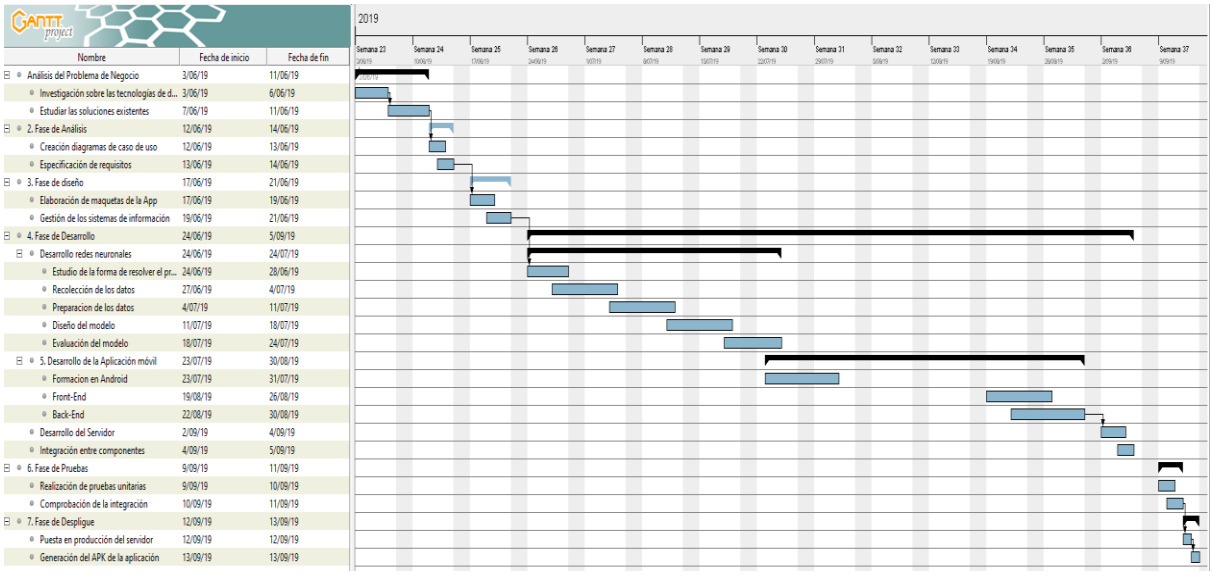


Figura 23. Diagrama de Gantt con la planificación del proyecto (Diseño propio).

El proyecto abarca aproximadamente unas **doce semanas de trabajo**, siendo la fase que más tiempo requiere la fase del desarrollo y por otro lado la relativa al despliegue es la más breve. Como se observa en el diagrama se sigue en todo momento una estructura lineal consecutiva sin evidentes solapamientos, todas las tareas recaen en una sola persona lo que impide la ejecución de tareas de forma concurrente.

La planificación por su parte no contempla ningún entregable ya que el único que se debe generar es el propio proyecto una vez se finalice. Es importante que se preste atención a la planificación de vez en cuando para comprobar si es posible cumplir con la entrega, en caso de retrasos se deberá actualizar y valorar el retraso de la entrega, la reestructuración de tiempos o la dedicación de un tiempo extra para poder cumplirla.

6.2. Casos de uso y captación de requisitos

Una vez da comienzo la segunda fase, se debe comenzar a cuestionar lo que verdaderamente debe hacer el sistema de información, qué necesita ser capaz de hacer para que podamos considerarlo como un sistema completo capaz de satisfacer la solución de negocio.

Ya que el rol del cliente lo asume el propio desarrollador, no será necesario consultar sobre a otras personas sobre las funcionalidades o características que debe tener el sistema. Aun así lo importante es establecer aquellos requerimientos que el sistema de información debe cumplir para marcar así unas buenas directrices para los bocetos y el desarrollo.

Dejar lo más claro posible desde el principio cómo debe ser y cómo debe comportarse el sistema, ayuda en muchos casos a evitar desviaciones en la planificación o a evitar errores en etapas más tardías del proyecto. A medida que se avanzan en las fases del proyecto más difícil es subsanar errores que provienen de las fases o tareas iniciales.

Para formalizar desde un punto de vista ingenieril qué debe hacer el sistema se utilizarán los casos de uso. Un caso de uso es un elemento clave de la ingeniería del software, permite describir de forma clara cada acción que puede realizar el actor con el sistema. En su defecto un diagrama de casos de uso es la agrupación gráfica de todos los distintos casos de uso que componen el funcionamiento del sistema (Ver el siguiente diagrama para entender los elementos de estos diagramas).

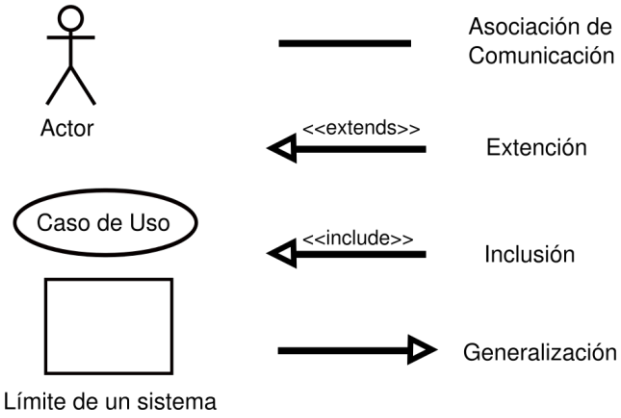


Figura 24. Notación de los elementos pertenecientes a los diagramas de casos de uso (photographer, 2006).

El sistema de información a diseñar solo contemplará a dos posibles **actores**: **Administrador** y el **Usuario** final. Para elaborar los casos de uso se debe tener en cuenta las distintas acciones que pueden realizar los actores sobre el sistema así como las interacciones que deben completarse para cada que cada acción se pueda completar.

La elaboración del diagrama de casos de uso es posible realizarla gracias a la herramienta **MagicDraw UML** propiedad de No Magic. En su versión de demostración gratuita podremos crear en el lenguaje gráfico de modelado UML (*Unified Modeling Language*) una visión general del sistema que nos permita definirlo. En la siguiente figura se muestra el resultado obtenido tras la elaboración del diagrama para el sistema:

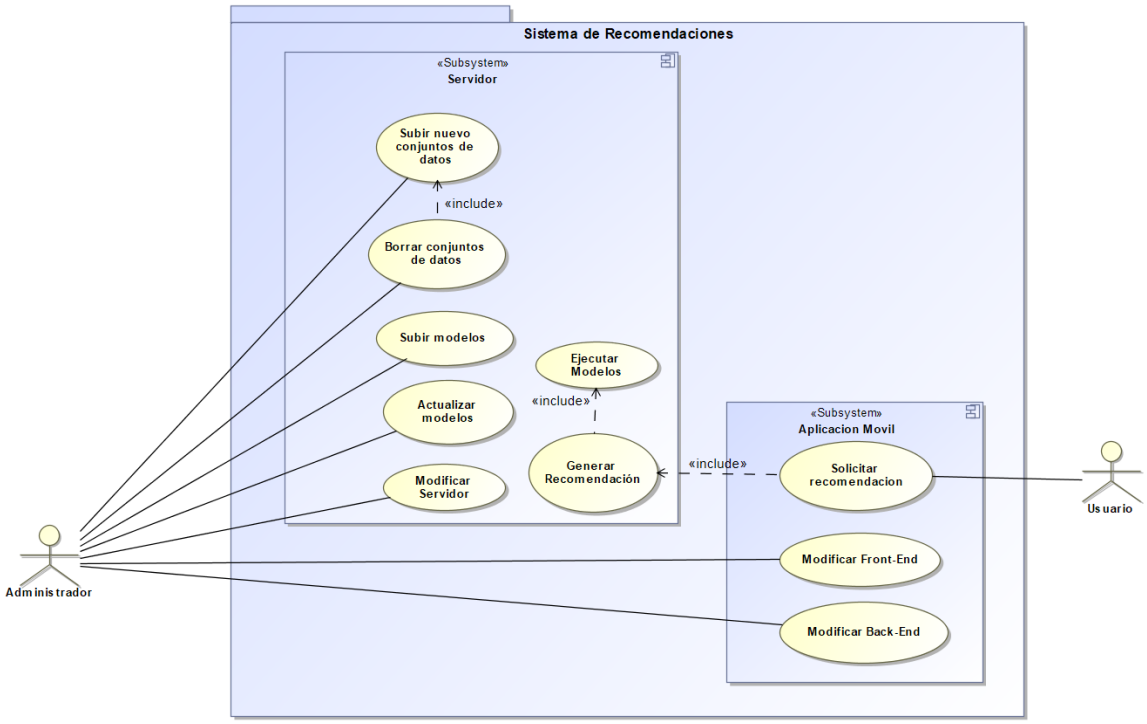


Figura 25. Diagrama de casos de uso del sistema de recomendaciones (Diseño propio).

Para facilitar la compresión del diagrama cabe destacar que el sistema de recomendaciones se descompone a su vez en **dos subsistemas**: el subsistema del **servidor** y el de la **aplicación móvil**. Ambos se limitan a los casos de uso que agrupan, estos además interactúan entre sí para que el usuario pueda obtener así recomendación desde la aplicación.

En el diagrama apreciamos las distintas acciones que pueden realizar los dos actores. Por un lado el usuario final solo dispone de la posibilidad de solicitar una recomendación, para completarla se debe interactuar con el servidor debido a la asociación de <<*Include*>> entre los casos de uso. Esta relación obliga a que se complete también el caso de uso de la generación en el servidor, el cual a su vez requiere de la ejecución de los modelos de redes neuronales.

El Administrador del sistema por su parte dispone de más acciones y permisos sobre el sistema pudiendo modificar los *datasets*, los modelos, el funcionamiento del servidor y la propia aplicación móvil.

Gracias al diagrama de la figura 25 es posible redactar una lista de los requisitos sobre cada uno de los casos de uso. Este proceso se conoce como **especificación de requisitos de software** (ERS), consiste en dar una descripción detallada del comportamiento o funcionalidades y de las propiedades (diseño, seguridad, estándares…) que tendrá el sistema.

Los **requisitos** pueden ser **funcionales** (casos de uso o **RF**) o **no funcionales** (propiedades o **RNF**). A continuación se detallan a modo de ejemplo algunos de los requisitos del sistema, para ver la lista completa acudir a los anexos de este trabajo:

Nombre	**RF-01:** Subir nuevo conjunto de datos
Versión	1.0
Actor	Administrador
Descripción	• Se debe permitir el almacenamiento de nuevos conjuntos de datos en el sistema. Esto se pueden destinar a los resultados de las recomendaciones o al entrenamientos/validación de las redes neuronales.

Tabla 14. Requisito funcional número uno del sistema (Diseño propio).

Nombre	**RF-06:** Solicitar recomendación
Versión	1.0
Actor	Usuario
Descripción	• Desde la aplicación móvil el usuario será capaz de subir una imagen. La foto será enviada al servidor para que allí se generen las etiquetas y recomendaciones utilizando para ello las redes neuronales disponibles.

Tabla 19. Requisito funcional número seis del sistema (Diseño propio).

Nombre	**RNF-01:** Garantizar seguridad durante la recomendación
Versión	1.0
Descripción	• La imagen facilitada por el usuario para obtener la recomendación debe enviarse cifrada al servidor, garantizando así la confidencialidad e integridad de la misma.

Tabla 24. Requisito no funcional número uno del sistema (Diseño propio).

6.3. Creación de bocetos para la aplicación móvil

Previo al desarrollo en Android Studio se diseñarán distintos *wireframes* o esquemas visuales que servirán de esqueleto para la interfaz gráfica. En ellos se esquematiza la distribución de los elementos, botones e imágenes y sobre él se realizan anotaciones a modo de puntualizaciones.

El boceto a utilizar para el desarrollo en Android Studio debe hacerse de forma que permita conocer y mostrar a simple vista las funcionalidades que ofrece la aplicación, debe ser lo más modular posible, con pocas dependencias entre elementos para un mantenimiento más sencillo y sobre todo debe adaptarse a los requisitos descritos en el anexo.

En el diseño de las aplicaciones móviles según la plataforma y el fin de la aplicación es posible basar el desarrollo en los conocidos como patrones de diseño software. Estos patrones son muy diversos: existen patrones estructurales que pueden establecer cómo se relacionan entre sí las clases de Java, los de comportamiento para la gestión de algoritmos y relaciones entre los propios objetos, de diseño para la distribución y colores de las imágenes, botones etc.

En este diseño se seguirá un patrón MVC (Modelo, Vista, Controlador) detallado con mayor detalle en el siguiente capítulo durante el propio desarrollo, además se seguirá un patrón de diseño basado en la tutorización. Los comentarios e imágenes de la propia aplicación describen en forma de tutorial su principal funcionalidad y la manera de utilizarla.

El sitio web de Moqups permite utilizar su herramienta web para elaborar bocetos de una forma sencilla mediante el "arrastre y suelte" de los elementos disponibles en el panel derecho sobre una plantilla inicialmente vacía. En su versión gratuita el número de elementos que permite utilizar se ve limitado pero es suficiente para elaborar las maquetas que permitan modelar la interfaz de la aplicación.

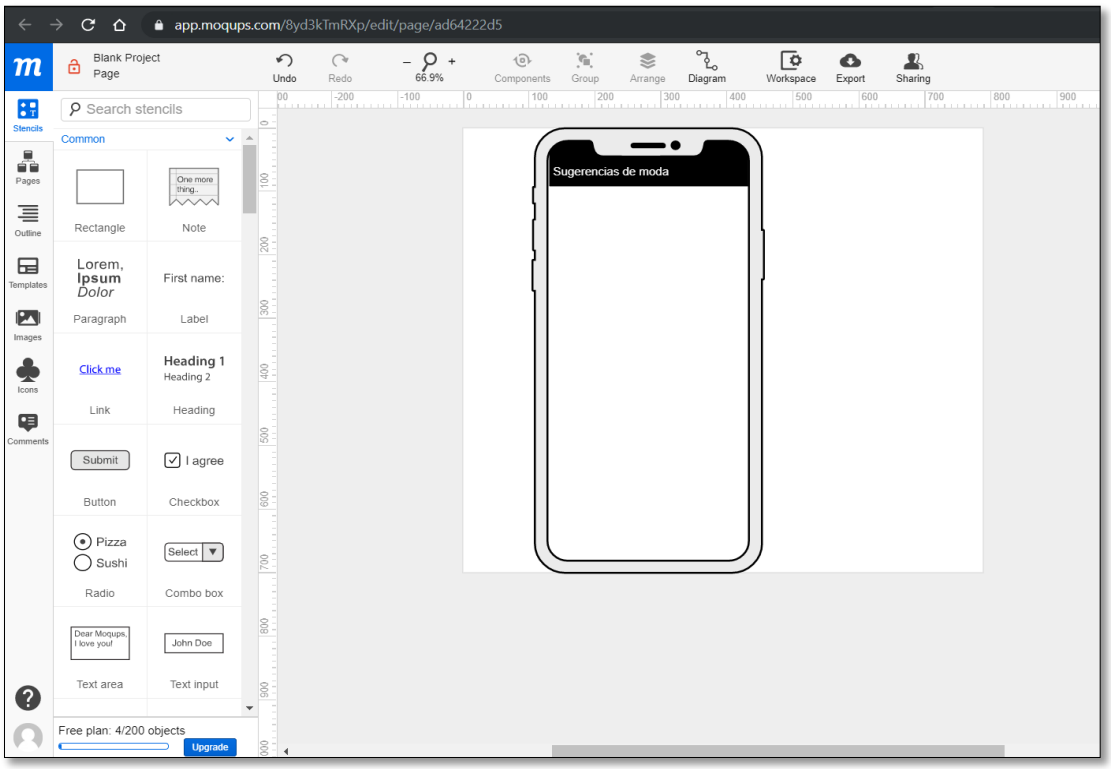


Figura 26. Visualización interna de la herramienta Moqups durante la creación de bocetos (Diseño propio).

Descubiertas las posibilidades que ofrece Moqups con su herramienta, se ha generado tal y como se aprecia en la tabla inferior las dos vistas que compondrán la interfaz de la aplicación. La vista inicial muestra una serie de campos de texto y una imagen que describirán la forma en la que el usuario puede utilizar la aplicación y solicitar la recomendación, a su vez dispondrá de dos botones que le permitirán hacerlo: botón de cámara y botón de galería. En la segunda vista, la vista de los resultados de las recomendaciones podremos observar la información relativa al artículo que se ha subido así como varias imágenes de los posibles productos recomendados.

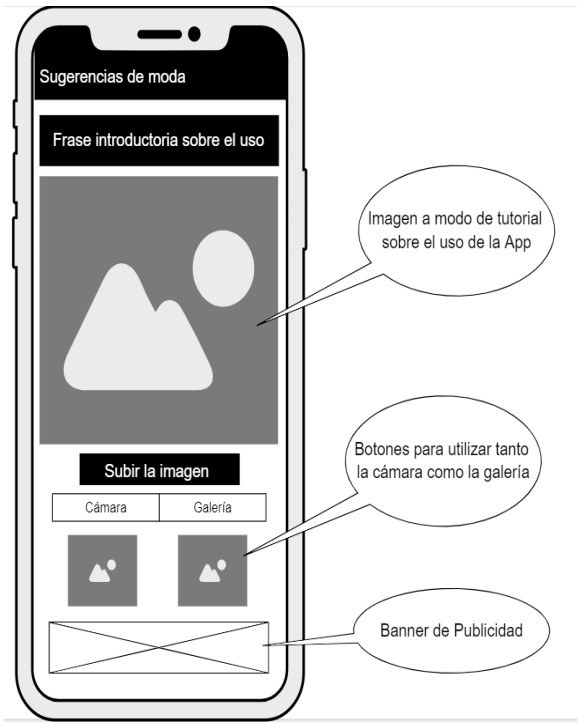
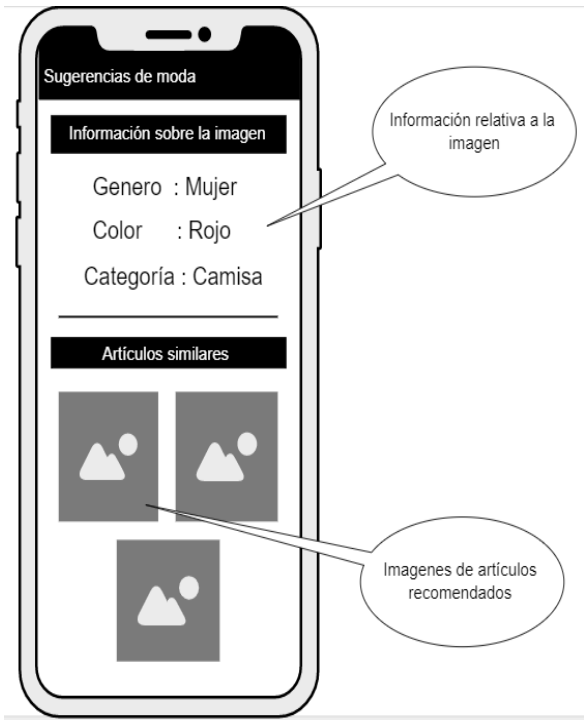
Vista Inicial	Vista Recomendaciones
	

Tabla 10. Bocetos de la interfaz de la aplicación (Diseño propio).

6.4. Gestión de los datos

Este punto se comienza abordando uno de los aspectos informáticos de los proyecto que más importancia cobra debido a los constantes cambios legislativos, el ***Data Governance***. El "gobierno de los datos" consiste en la definición de políticas de control sobre la exactitud, accesibilidad, consistencia, integridad y actualización de los datos.

Los proyectos relativos al campo de la inteligencia artificial y del *machine learning* exigen el manejo de inmensas cantidades de datos. Este proyecto no solo exige datos para las recomendaciones sino también para el entrenamiento de los modelos. Para garantizar la reproductividad de los resultados de los modelos es necesario que se propongan políticas de control como las anteriores.

Un tema a cuidar en este proyecto es la reproductibilidad de las distintas versiones de los modelos, estos deben ser reproducibles. En caso de pérdida del fichero del modelo, de deterioro del modelo tras aplicarle cambios o de que no se genere los mismos resultados por haber entrenado con otro conjunto distinto, el modelo original se debe poder volver a generar. Como política para garantizar la reproductibilidad, se almacenará de forma comprimida los datos utilizados para el entrenamiento junto con una copia de la versión del código del modelo.

Procedemos a comenzar con la búsqueda de los datos para alimentar el sistema, estos deben ser datos con formatos de ficheros de imagen. La red neuronal diseñada trabajará con este tipo de dato pues es el tipo que nos facilitará el usuario a través de la aplicación.

Una gran diferencia con respecto a otros tipos de proyectos, es que los proyectos que requieren del uso de redes neuronales por lo general necesitan de un gran conjunto de datos, en este caso de imágenes. En ocasiones, al tratarse de imágenes se pueden llegar a manejar con mucha facilidad, directorios que alcanzan grandes cantidades de *Gygabytes*.

Para la obtención de este tipo de recursos de manera gratuita debemos recurrir a la internet, comenzaremos aquí la fase de **Extracción** (Extración, Trasnformación y Carga) dentro de lo que sería un proceso ETL manual. Tras una búsqueda en inglés es posible encontrar diversos *datasets* de productos de moda (ver figura 27), algunos están disponibles en una comunidad de científicos de datos propiedad de Google, denominada Kaggle.

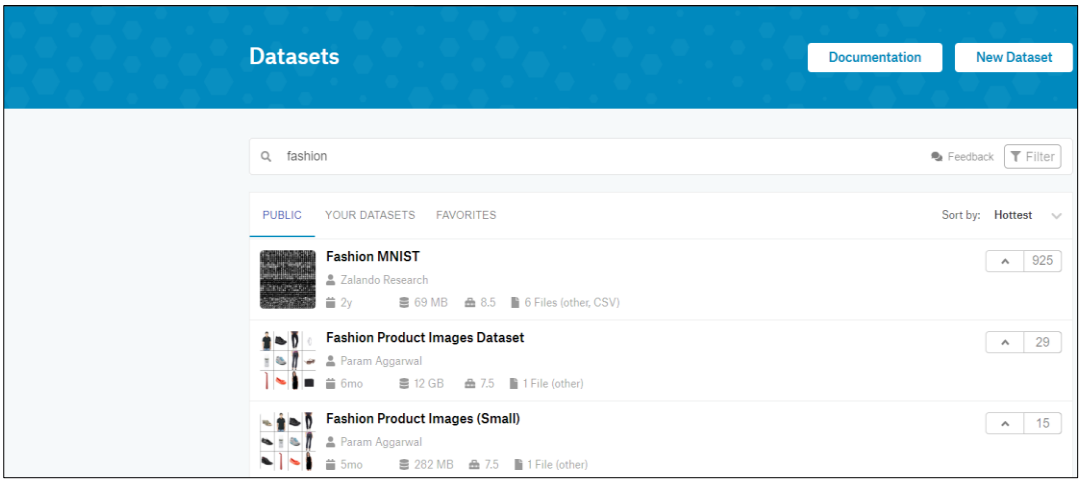


Figura 27. Resultados de la búsqueda de datasets de moda en Kaggle (Diseño propio).

Nos quedaremos con el *dataset Fashion Product Images (Small)* y descarguemos igualmente su versión completa de 12Gb. El tamaño que ocupa en disco es bajo, por lo que su almacenamiento no supondrá un problema ni su uso para las redes. Los factores decisivos para la elección como *dataset* para el sistema han sido:

- Contiene un **gran número de imágenes** (40k).
- Imágenes en **color** y con **diversidad** de imágenes para **cada categoría**.
- Su versión comprimida es muy compacta, apenas 300 Mb.
- Existe un fichero **CSV** (comma separated values) manejable con las etiquetas par distintas categorías de cada uno de los productos.

	A	B	C	D	E	F	G	H	I	J	K	L
1	id	gender	masterCategc	subCategory	articleType	baseColour	season	year	usage	productDisplayName		
2	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011	Casual	Turtle Check Men Navy Blue Shirt		
3	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012	Casual	Peter England Men Party Blue Jeans		
4	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016	Casual	Titan Women Silver Watch		
5	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011	Casual	Manchester United Men Solid Black Track Pants		
6	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012	Casual	Puma Men Grey T-shirt		
7	1855	Men	Apparel	Topwear	Tshirts	Grey	Summer	2011	Casual	Inkfruit Mens Chain Reaction T-shirt		
8	30805	Men	Apparel	Topwear	Shirts	Green	Summer	2012	Ethnic	Fabindia Men Striped Green Shirt		

Figura 28. Resultados de la búsqueda de datasets de moda en Kaggle (Diseño propio).

Para confirmar que la elección del *dataset* es la acertada o válida se obtendrá mediante un análisis estadístico un resumen de la información real de la que se dispone.

7. Desarrollo, pruebas y despliegue del sistema

Durante esta segunda etapa del proyecto, comenzamos en primer lugar con la fase de desarrollo. Se parte por realizar un análisis, tratamiento y limpieza sobre la fuente de datos (*dataset*) escogida en el capítulo anterior. Veremos si cumple algunos criterios básicos como para considerarla válida para el sistema y para el entrenamiento de las redes neuronales.

Las redes deberán ser capaces de generar las etiquetas sobre el producto que aparezca en la imagen recibida. Esta tarea de la fase será completada bajo las directrices propias de la metodología CRISP-DM.

Para hacer un uso práctico de la solución se elaborarán dos nuevos componentes: un servidor y un cliente móvil. El servidor desarrollado en Python permitirá el control de las redes neuronales y de sus resultados, así como del envío y recepción de los datos mediante una conexión con el cliente, detallado en la parte de integración de componentes. Por el otro lado el cliente funcionará sobre Android. El cliente será quien podrá solicitar recomendaciones al servidor mediante el establecimiento de una conexión y el envío de una imagen cifrada.

Cuando todos los componentes estén listos e integrados se dará paso a la realización de pruebas que permitan comprobar su calidad y funcionamiento así como la calidad de los resultados en términos estadísticos. Si todos los resultados son satisfactorios se procederá al despliegue mediante la generación de un APK de la aplicación y la puesta en marcha sin cese del servidor.

7.1. Preparación y estudio del conjuntos de datos

En el estudio del estado del arte se incluyó una selección de las herramientas y lenguajes de programación. En el caso del análisis del *dataset* y el diseño de las redes se elaborará desde Python mediante el IDE Jupyter Notebook provisto por la suite Anaconda. Comenzaremos por arrancarlo mediante el terminal y crear un nuevo *Notebook* de trabajo.

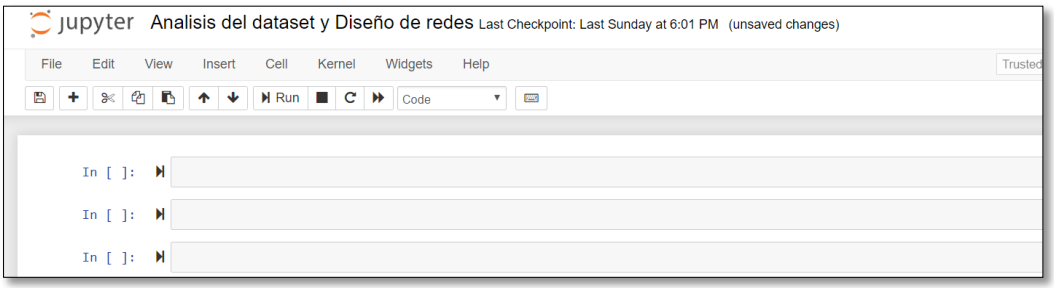


Figura 29. Visualización del entorno de trabajo en un Notebook del IDE Jupyter (Diseño propio).

Cada celda del *Notebook* se dedicará a trozos de código en Python con fines distintos, pero relacionados entre sí de forma secuencial.

En primer lugar procederemos a importar las librerías imprescindibles para cualquier proyecto de ciencia de datos (pandas, numpy, matplotlib…). Importarlas permitirá el manejo de ficheros, trabajar con *dataframes* (información estructurada por columnas), realizar operaciones estadísticas, mostrar resultados de forma gráfica etc.

```
import pandas as pd
import scipy as sp
import os
from os import path
import numpy as np
import csv

#carga de librería para manejo de los directorios del SO
import shutil

#Carga de libreria para la gestión de errores
#from pandas.parser import CParserError

#Librerías para realizar muestras por pantalla de gráficos e imagenes
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib import image
import seaborn as sn

#Librerias para las imagenes
from PIL import Image
```

Figura 30. Listado de librerías importadas para trabajar en el Notebook (Diseño propio).

Importadas las librerías, es momento de cargar el documento "styles.csv", en él se encuentra toda la información relativa a cada una de las imágenes descargadas en el archivo comprimido *Fashion Product Images (Small).* Si la lectura es correcta es posible visualizar el contenido del documento tal y como se aprecia en la siguiente figura.

```
In [5]: #Observamos que se ha cargado correctamente y la información disponible
        df.head()
```

Out[5]:

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName	Unnamed: 10	Unnamed: 11
1	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011	Casual	Turtle Check Men Navy Blue Shirt	NaN	NaN
2	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012	Casual	Peter England Men Party Blue Jeans	NaN	NaN
3	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016	Casual	Titan Women Silver Watch	NaN	NaN
4	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011	Casual	Manchester United Men Solid Black Track Pants	NaN	NaN
5	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012	Casual	Puma Men Grey T-shirt	NaN	NaN

Figura 31. Cabecera del dataframe cargado a partir del fichero de datos de las imágenes (Diseño propio).

Comprobaremos la calidad real los datos procediendo a primero a comprobar la presencia de valores nulos en las distintas columnas disponibles. Este paso ayudará a la restringir la selección de las imágenes a utilizar y a seleccionar las columnas que nos resulten más interesantes para la generación modelos.

```
In [7]: #Observamos la existencia de nulos e información incompleta
        print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44446 entries, 1 to 44446
Data columns (total 12 columns):
id                  44446 non-null object
gender              44446 non-null object
masterCategory      44446 non-null object
subCategory         44446 non-null object
articleType         44446 non-null object
baseColour          44431 non-null object
season              44425 non-null object
year                44445 non-null object
usage               44129 non-null object
productDisplayName  44439 non-null object
Unnamed: 10         0 non-null float64
Unnamed: 11         2 non-null object
```

Figura 32. Información resumida sobre el dataframe (Diseño propio).

Comienza la parte de **Transformación** (ETL) del dato. Con objeto de poder realizar las recomendación se ha seleccionado como fuente de información las columnas destinadas al **género**, el tipo de **artículo** y el **color**. Estos serán los atributos que marcarán las redes neuronales a diseñar. Por su parte el identificador se ha destinado como índice para los elementos del nuevo *dataframe*, además no ha sido necesario eliminar registros ya que estas tres columnas se encuentran completamente informadas.

```
In [10]:
#Seleccionamos las columnas que nos interesan para los modelos y colocamos como index el id
df_1 = df_1[['id','gender','articleType','baseColour']]
df_1.set_index(df_1['id'], inplace = True)
df_1.pop('id')
#Vemos como ha quedado el dataframe
df_1.head()
```

Out[10]:

id	gender	articleType	baseColour
15970	Men	Shirts	Navy Blue
39386	Men	Jeans	Blue
59263	Women	Watches	Silver
21379	Men	Track Pants	Black

Figura 33. Información resumida sobre el dataframe (Diseño propio).

En términos estadísticos estudiaremos la información disponible para cada columna, pudiendo comprobar cómo es su distribución por valores. Generamos un gráfico de barras para agrupar en cada barra el número de imágenes de cada género (ver figura 34). Se colorea el porcentaje de presentación de verde si supera el corte y rojo si no, el corte lo marca si al menos tiene un porcentaje representativo superior a un 5% sobre el total, ya que hay pocos "generos".

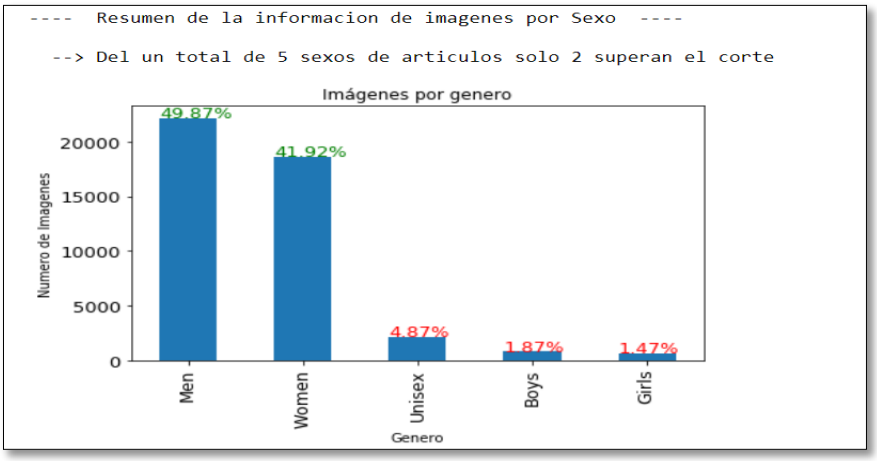


Figura 34. Información resumida sobre el dataframe (Diseño propio).

Se observa que la población de imágenes está repartida de forma muy homogénea entre imágenes de hombres y de mujeres, no tanto como para niños, niñas o elementos unisex. De esta forma el **género** lo limitaremos a **masculino** y **femenino**.

Si repetimos este mismo proceso para las categorías de productos lo haremos según el sexo, de esta forma **escogeremos** las **categorías** más **representativas** de cada **género**. En el caso del productos cuya categoría se asocie a los hombres y utilizando un percentil 60 veremos un gráfico como el que se aprecia en la siguiente figura.

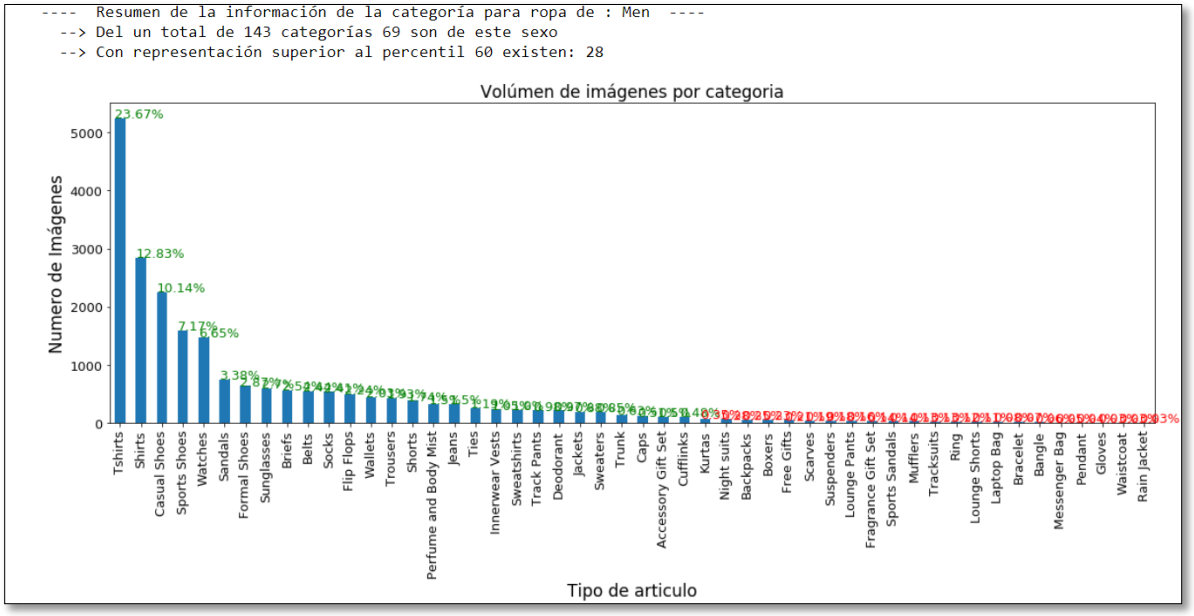


Figura 35. Gráfico simplificado de la información resumida para categorías para hombres (Diseño propio).

Previamente a la selección completa de las categorías asociadas a hombres que pasan el corte, **reduciremos** las **categorías** eliminando las que no se asocian a las artículos de moda como los perfumes, desodorantes, accesorios de baño etc. Nuevamente se vuelve a ejecutar en este caso sobre las categorías de productos de mujeres con el mismo percentil anterior, véase en la siguiente figura los resultados.

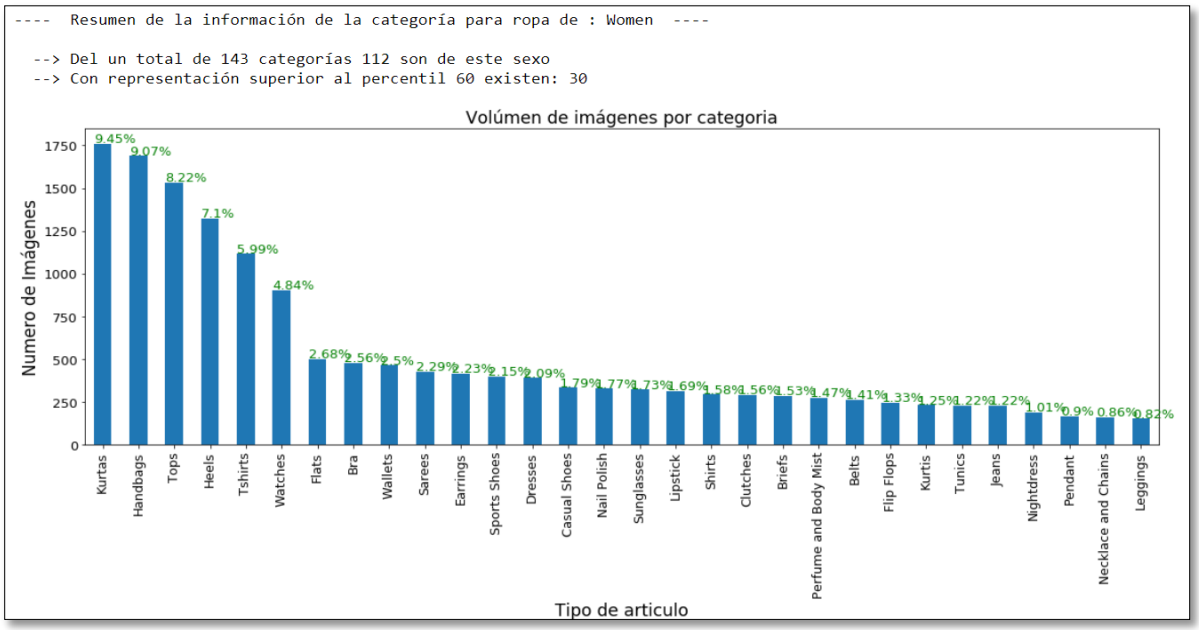


Figura 36. Gráfico simplificado de la información resumida para categorías para mujeres (Diseño propio).

Al igual que en las categorías de hombre **reduciremos** las **categorías** eliminando las que no se asocian a las artículos de moda como los pintalabios, perfumes…

Como último atributo a analizar tenemos los colores de los artículos, mostramos en un "gráfico de tarta" los colores más representativos según el mismo percentil (Ver figura 37). No modificaremos la selección ya que no tenemos colores que puedan causar confusión entre ellos.

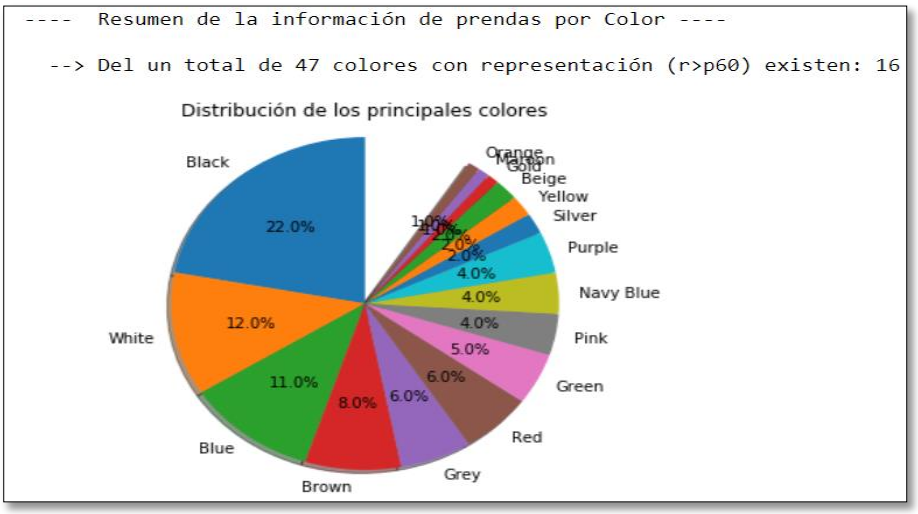


Figura 37. Gráfico simplificado de la representatividad de cada color en el dataset (Diseño propio).

Redefiniremos también los valores de cada registro para los atributos de color y categoría sustituyendo los cada espacio por un '_' , esto evitará problemas en las redes neuronales. Por último para dar por finalizado el tratamiento guardamos en un fichero Excel el *dataframe* obtenido al que le hemos quitado los registros de los 4 ficheros que no se encuentren en el directorio donde se ubican las imágenes tras haber realizado una búsqueda en Python.

7.2. Construcción de las redes neuronales

La elección Python como lenguaje de programación no fue decisión infundada, ni tomada únicamente en la posibilidad de manejar grandes cantidades de datos de forma sencilla con pocas líneas de código. Python dispone de multitud de APIs (Interfaces de programación de aplicaciones), una de las más destacadas para la programación de redes neuronales es Keras.

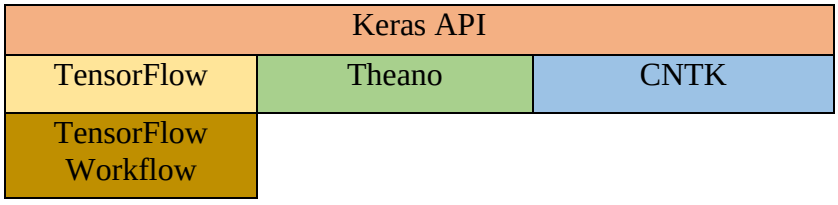


Figura 38. Esquema por capas de la API Keras (Diseño propio).

Keras permite a un alto nivel trabajar sobre TensorFlow, CNTK o Theano. Por defecto no está incluido en la instalación que realizamos con Anaconda por lo que se debe instalar mediante el terminal que ofrece Anaconda o mediante su interfaz gráfica (Ver la guía de instalación para más detalles). Completada su instalación se importarán diferentes paquetes para diseñar, entrenar y ejecutar las redes neuronales.

```
import keras.backend as K
from keras.models import Sequential , load_model
from keras.layers import Dense,Dropout,BatchNormalization,Conv2D,MaxPool2D,Dense, Flatten
from keras.utils import to_categorical
from keras.optimizers import SGD,Adam
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
from keras.callbacks import EarlyStopping
from sklearn.metrics import classification_report, confusion_matrix
```

Figura 39. Listado de librerías importadas para trabajar con Keras (Diseño propio).

Importamos además dos librerías de Sklearn para generar los conjuntos de datos que utilizaremos para el entrenamiento y la validación, elaborando para ello una función. Junto a ella desarrollaremos una lista más extensa de funciones para la generación, entrenamiento y prueba de las redes neuronales, funciones descritas en la siguiente tabla.

Función	Finalidad
`def conjuntos_dataset(columna, dataFrame):`	Mediante el *dataframe* y un nombre de una columna del mismo se devuelve un conjunto de entrenamiento y de prueba, junto a los resultados para cada conjunto.
`def creacion_directorios(columna, dataFrame):`	Mediante el *dataframe* y un nombre de una columna del mismo se genera una carpeta para dicha columna. Se crea una carpeta de pruebas y otra de entrenamiento con subcarpetas internas que incluyen las imágenes ordenadas según su pertenecía a la columna.
`def generador_modelo (num_salidas,X_train, X_test, train_dir, test_dir, epochs=1, batch_size=32 ):`	Devolvemos como salida una red entrenada y un test. Para obtenerlos se indica el nº de salidas de la red, los datasets de entrenamiento y prueba , junto a sus directorios. Podemos escoger también los *epochs* (repeticiones del entrenamiento completo) y el tamaño del *batch (imágenes a procesar para actualizar los parámetros)* para no utilizar los establecidos por defecto.
`def load(filename, img_width =60, img_height=80):`	Devuelve una imagen tras cargarla y adaptarla a resolución que debe tener para utilizarla en la red.

`def etiqueta_genero(path):`
`def etiqueta_articulo_mujer(path):`
`def etiqueta_articulo_hombre(path):`
	Cada función genera las etiquetas indicadas en su nombre mediante la ruta de la imagen a utilizar.

`def guardar_modelo (modelo, nombre):`	Guardamos el modelo en la carpeta de modelos con el nombre recibido como argumento.

Tabla 11. Nombres y argumentos de las funciones creadas y su finalidad (Diseño propio).

El proceso de generación de las redes se realiza mediante la función **generar_modelo**, detallada en la tabla anterior. Internamente la función sigue una plantilla de red configurable, es posible cambiar mediante un parámetro el número de salidas que puede tener la red. Del número de posibilidades dentro de cada categoría a predecir dependerá por tanto el número de salidas (capa *softmax*).

La imagen de entrada a la red deberá poseer unas dimensiones 80 pixeles de alto y 60 pixeles de ancho, dimensiones que coinciden con las de todas las imágenes del *dataset* descargado. A mayor resolución mayores son las probabilidades de que el funcionamiento de la red sea mejor, aunque esto supondría sacrificar en rendimiento para la fase de entrenamiento haciéndola requerir de mayor tiempo de cómputo para realizar las operaciones.

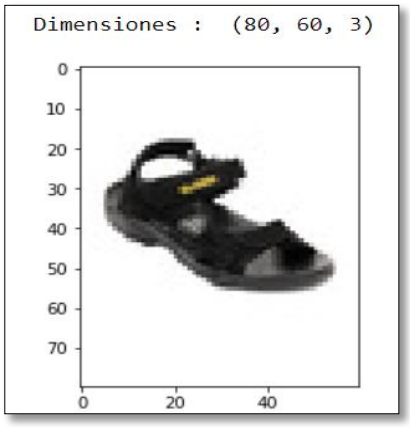


Figura 40. Dimensiones que comparten todas las imágenes del dataset small (Diseño propio).

Mencionar que la arquitectura de la red se ha detallado de forma gráfica en los anexos, gracias a una herramienta de diseño web. De igual forma en Python tras generar una red es posible obtener un resumen de su arquitectura mediante la función *summary()*.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 78, 58, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 39, 29, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 38, 28, 32)        4128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 19, 14, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 17, 12, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 6, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1536)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1536)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               393472
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 514
=================================================================
Total params: 408,258
Trainable params: 408,258
Non-trainable params: 0
```

Figura 41. Salida de la función summary() sobre la red para determinar el género (Diseño propio).

La plantilla de la red neuronal coincide con la idea de una red neuronal convolucional (CNN). Sus 6 primeras capas componen las capas de extracción de características de la imagen, todas ellas comparten el número de nodos que es 32, sin embargo los nodos de cada capa reciben y generan matrices con dimensiones distintas debido a las operaciones que realizan sobre la matriz recibida de la capas anterior. De forma alterna se hace uso de una convolución y tras ello un *max_pooling* con una máscara *de 2x2*, hasta reducir considerablemente el número de pixeles de la matriz antes de la fase de clasificación

Tras las capas anteriores comienzan las capas orientadas a realizar la clasificación de la imagen. La primera realiza un *flattening* o aplanado de la totalidad de pixeles, tras ello la capa de *dropout* permite de forma aleatoria hacer que no se utilicen la totalidad de valores de los 1536 pixeles. De esta forma la red no se hace dependiente de los valores de unos determinados píxeles y puede realizar un aprendizaje más profundo. En último lugar se aplican capas de densidad para reducir poco a poco el número de pixeles a una última capa con activación software y cuyo número de salidas viene determinado por el argumento recibido en la función.

Destacar que la gran mayoría de imágenes se encuentran correctamente centradas, con buena iluminación y sobre un fondo blanco. Esto puede impedir que la clasificación de la red para las imágenes en condiciones diferentes a esas, no generen resultados satisfactorios. Para evitarlo, el entrenamiento se realizará sobre las propias imágenes a las que se les aplicará previamente alguno de los cambios indicado en un *ImageDataGenerator*. Este generador permite cambiar la orientación, el brillo, girar las imágenes etc.

Opciones del generador	Resultados ofrecidos

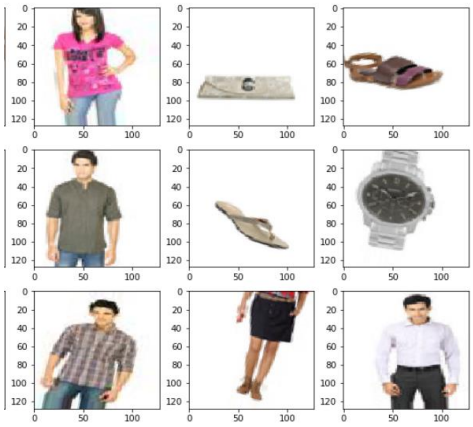
`train_datagen = ImageDataGenerator(`
`  rescale=1. / 255,`
`  shear_range=0.2,`
`  zoom_range=0.2,`
`  brightness_range=[0.75, 1.25],`
`  horizontal flip=True)`
	

Tabla 12. Resultados obtenidos tras aplicar el ImageDataGenerator sobre las imagenes (Diseño propio).

Conocido el funcionamiento del generador de modelos, deberemos generar un total de **4 modelos distintos** mediante un entrenamiento de 3 epochs. Modelos para : género, categorías para hombre, categorías para mujer y colores.

```
#Generamos un modelo acorde a las salidas esperadas (2 por generos Hombre Mujer,)
modelo_genero, test_genero = generador_modelo(2, X_train, X_test, train_data_dir, test_data_dir, epochs=3)
```

```
Found 21170 images belonging to 2 classes.
Found 9074 images belonging to 2 classes.
Epoch 1/3
  255/21170 [..............................] - ETA: 1:19:45 - loss: 0.5824 - acc: 0.6944
```

Figura 42. Generación de un modelo de red que averigüe el género de un producto (Diseño propio).

7.3. Evaluación de los resultados de las redes

Cada vez que termina el entrenamiento de un modelo es conveniente comprobar la calidad de sus resultados de forma estadística y a grandes rasgos. Para comenzar podemos realizar simples pruebas con algunas imágenes, a ser posible se utilizarán imágenes con las que la red no haya entrenado.

```
#Simulacion de prueba con una imagen
nombre_imagen="images/1165.jpg"
imagen = load(nombre_imagen)

df_aux = pd.DataFrame(columns=sorted(list(df_1['gender'].unique())))
df_aux = df_aux.append(pd.Series(np.round(modelo_genero.predict(imagen)[0].ravel(),2)*100,
                                 index=df_aux.columns ),
                       ignore_index=True)
df_aux.head()
```

	Men	Women
0	0.0	100.0

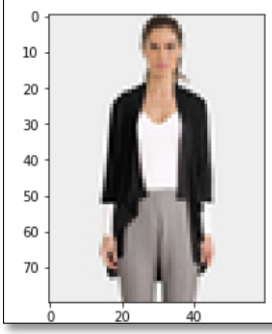


Figura 43. Prueba rápida y simple de un modelo mediante una única imagen (Diseño propio).

El resultado de la red del género nos otorga un porcentaje según su pertenencia a cada posible salida de la red. según sea en este caso hombre o mujer. La suma de todos los porcentajes debe por tanto ser 100. En la anterior figura el resultado es correcto y sin margen de error, no obstante un par de pruebas de este tipo no aseguran aún que la red vaya a funcionar siempre o casi siempre bien.

Cada modelo pasará por las mismas pruebas. Las pruebas se realizarán con sus respectivos conjuntos de prueba (los resultados para todos los modelos pueden consultarse en el anexo 10.3). Estos conjuntos disponen de un alto volumen de imágenes, en base a los resultados que se obtengan podremos tener una conclusión más acertada sobre los modelos.

Comentaremos una de ellas y se ofrecerá una valoración global tras haberlas realizado sobre todos los modelos. Las pruebas consisten en proporcionar el conjunto de imágenes de prueba a la red y comparar los resultados obtenidos con los que realmente deberían tener las imágenes. Existen diversas formas gráficas y diferentes valores estadísticos fácilmente calculables en Python para analizar la calidad de un modelo:

- ***Loss*** : Este valor estadístico es la suma arrastrada de los errores en los resultados generados por el modelo, esto depende de la función utilizada para el cálculo. A medida que el modelo cometa más errores mayor será este valor (Menor es mejor).

- ***Accuracy**:* La exactitud de un modelo indica en forma de porcentaje, qué parte del total de valores tiene correctamente asignado su valor. Si la exactitud es del 60% querrá decir que solo 6 de cada 10 imágenes son correctamente etiquetas.

- ***Precision**:* La precisión es el ratio de predicciones correctas sobre el total de predicciones positivas.

- ***Recall :*** La sensitividad es el ratio de predicciones correctas sobre el total de las observaciones de la clase o valor estudiado.

- ***F1-Score:*** Consiste en operar la precisión con la sensitividad mediante una operación poderada.

- ***Support:*** Número de elementos asociados a la clase o valor estudiado.

- ***Confusion Matrix:*** Permite comparar en una matriz de forma contabilizada las asignaciones realizadas de los elementos con a las distintas clases y compararlo con la que realmente debería tener asignada.

		Predicción	
		Clase A	***Clase B***
Observación	***Clase A***	***Verdadero Positivo***	***Falso Negativo***
	Clase B	***Falso Positivo***	***Verdadero Positivo***

Tabla 13. Esquema de la matriz de confusión (Diseño propio).

La figura 44 muestra los resultados estadísticos que presenta la red del género sobre el conjunto de prueba. El valor de la función de pérdida es bueno al igual que la exactitud, a tan solo 1 de cada 10 artículos no se le asignaría correctamente el género. La matriz de confusión presenta unos buenos resultados, aun así se puede decir que la red tiende a clasificar como más a los productos de mujeres como de hombre.

```
---- Red Neuronal para el Genero ----

Loss:  0.22270759548488223
Accuracy:  0.9079004552848555

Confusion Matrix

[[4789  355]
 [ 456 3372]]

Classification Report

              precision    recall  f1-score   support

         Men       0.91      0.93      0.92      5144
       Women       0.90      0.88      0.89      3828

    accuracy                           0.91      8972
   macro avg       0.91      0.91      0.91      8972
weighted avg       0.91      0.91      0.91      8972
```

Figura 44. Información estadística de los resultados tras las pruebas con el modelo del Género (Diseño propio).

Para el resto de modelos las pruebas muestran unos valores muy buenos, aunque mejorables. Destacar el caso de la red de identificación del color, los resultados de sus valores estadísticos y de su exactitud demuestran que con los entrenamientos realizados, la red aún no ha aprendido a diferenciar correctamente los colores de una prenda.

Para lograr un mejor funcionamiento que reduzca así los errores observables en la matriz de confusión entre otras medidas podemos: utilizar otra estructura de red, observar si los productos etiquetados disponen de colores mixtos en las imágenes impidiendo así aprender correctamente a la red, entrenar un mayor número de veces… Esto mejorará notablemente los resultados que pueda ofrecer la red neuronal de los colores.

7.4. Servidor funcional en Python

Preparar un servidor funcional en Python consiste en destinar un equipo informático, en este caso un ordenador, al suministro de un servicio basado en peticiones a una serie de clientes o usuarios. La arquitectura con la que se opera es de cliente-servidor, los usuarios serán quienes desde de la aplicación móvil a desarrollar realicen las peticiones y por otro lado el servidor ofrece como servicio la recomendación de productos de moda para los usuarios que lo soliciten.

El servidor funcionará sobre un sistema operativo, Windows 10 en este caso, además necesitará de todos los programas y requisitos hardware que garanticen su correcto funcionamiento.

Por parte del *Software* requeriremos nuevamente de Anaconda con los paquetes que permiten utilizar Keras instalado y de otros que se instalarán también para la gestión de las conexiones mediante sockets. En cuanto al *Hardware* se requiere como imprescindible una tarjeta de red que permita la conexión a un *router* u otro dispositivo que conecte al equipo a una red local. La CPU una vez ya generados los modelos, apenas realiza un mínimo esfuerzo para ejecutar los modelos.

El directorio donde el servidor ejecutará el programa desarrollado en Python dispondrá necesariamente de los siguientes elementos:

Figura 45. Elementos del directorio donde se realice la ejecución del programa en Python (Diseño propio).

1. Carpeta con las imágenes del *dataset*, a ser posible las descargadas con mayor resolución, para enviar al cliente una vez generadas las recomendaciones.

2. Una carpeta con todos los modelos generados para poder utilizarlos.

3. Un fichero Python con todas las librerías y funciones generadas a cargar por nuestro de ejecución del servidor.

4. Un fichero de configuración con el puerto sobre el que se realizará la conexión.

5. El fichero Python que una vez ejecutado dará pondrá en funcionamiento al servidor.

6. El fichero CSV con la información sobre los artículos de los que se dispone para obtener de este los artículos a recomendar.

7. Un fichero de texto con un diccionario de las traducciones de todos los términos, de esta forma los resultados obtenidos se puedan enviar en Castellano.

8. Ante cualquier problema se realizará una copia local y fácilmente accesible del código del servidor.

Como se había indicado en el capítulo anterior también será necesario almacenar una copia del *dataset* junto a cada modelo para garantizar la reproductibilidad de los mismos. Una vez se disponga de los elementos anteriores procederemos a programar el fichero del servidor.

Para comenzar, el programa carga todas las funciones y librerías necesarias para las recomendaciones, carga de modelos, transformaciones de datos…. alojadas en el fichero "funcsLibs.py"(3). Además en este fichero se incluirán las nuevas funciones que se desarrollen.

El programa que activa el servidor hará uso del fichero de configuración (4) para la elección del puerto. A continuación se abre un puerto de conexión al equipo para el establecimiento de la conexión con un cliente, se establece una clave para realizar encriptaciones/desencriptaciones para el método AES (detallado posteriormente) y tras ello se genera un diccionario mediante el fichero de traducciones (7).

```
#---------------------- Configuracion del Servidor y Carga de Ficheros --------------------------

%run funcsLibs.py #Cargamos las librerias necesarias al igual que las funciones que utiliza el servidor

#Establecimiento de la IP y puerto del equipo que actuará como servidor
confg = open("Server_config.txt","r+")
HOST='192.168.1.106'
PORT = int(confg.readline())
confg.close()

#Activamos la escucha de datos para la IP y Puerto anterior para un numero alto de usos
listensocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
numberOfConnections=999
listensocket.bind((HOST, PORT))
listensocket.listen(numberOfConnections)

#Establecemos una clave y generamos una variable para utilizar el cifrado AES en modo ECB
key = "7[z^eN]Y<pj8BF}Y"
aes = AES.new(key, AES.MODE_ECB)

#Creamos un diccionario mediante un fichero con las traducciones de los terminos
traduccionTerminos = crearDicFichero("traducciones.txt")
```

Figura 46. Código inicial del fichero de programa Python que activa el servidor (Diseño propio).

Esto se ejecuta una sola vez ya que el resto del tiempo el servidor se encontrará en bucle, a la espera de peticiones mediante el establecimiento de conexiones vía socket. El servidor abrirá el socket y quedará a la espera de una nueva conexión. Una vez esta se establezca comenzarán los intercambios de datos (detallados en la integración).

Mediante la conexión establecida se puede enviar y recibir de imágenes y datos. Tras recibir la imagen encriptada desde el cliente, la imagen se desencripta y se procesa mediante las redes neuronales para después ser borrada. Las redes diseñadas generarán las etiquetas (genero, sexo y color), gracias a ellas y a la función diseñada *ficheros_articulos (etiquetas,df)* obtendremos las 3 imágenes de productos recomendados que serán enviadas al cliente.

El proceso de recomendación se basa en estas dos reglas:

1. Si se disponen de al menos 3 imágenes con las mismas etiquetas que la recibida, se escogen 3 imágenes aleatoriamente de entre las disponibles.

2. Si se dispone de menos de 3 imágenes con las mismas etiquetas, las imágenes restantes hasta esas completar esas 3 se escogen de forma distinta. Se hace de forma aleatoria pero se obliga a que coincidan las etiquetas del género, el artículo y no la del color.

Una vez procesada la imagen, se envía la información traducida del artículo detectado y las imágenes obtenidas se procede a cerrar la conexión con el cliente. El servidor informa de que se queda a la espera de una nueva conexión (figura 47), para volver a facilitar el servicio al que está destinado, ofrecer recomendaciones de moda. Esto será posible tras la integración.

```
---------- Esperando nueva Conexion ------------
Conexión recibida....
Recibiendo 206576 bytes de imagen
OK - Imagen recibida correctamente
OK - Etiquetas enviadas
Imagenes : [28686, 4086, 11781]
Enviando imagen Nº0 ...
OK - Imagen nº0 recibida
Enviando imagen Nº1 ...
OK - Imagen nº1 recibida
Enviando imagen Nº2 ...
OK - Imagen nº2 recibida
Conexion Cerrada...

---------- Esperando nueva Conexion ------------
```

Figura 47. Secuencia de mensajes emitidos durante el funcionamiento del servidor (Diseño propio).

7.5. Desarrollo de aplicación en Android Studio

Finalizado el desarrollo de las redes y el servidor, es momento de crear el cliente. El cliente se modela como una aplicación para el sistema operativo Android y será desarrollada en Android Studio. La aplicación será por tanto el modo en el que el usuario podrá hacer uso del sistema de información y por tanto de la obtención de las recomendaciones.

Para comenzar creamos un nuevo proyecto en Android Studio, este proyecto divide en subcarpetas los recursos según su finalidad (ver figura 48). Por un lado el *Manifests* es el fichero donde se establecen los permisos que se requieren del dispositivo o se indica el nombre de la aplicación. La carpeta para *Java* aloja los controladores y otras clases que se deseen utilizar. Por otro lado en la carpeta de *Resources* se destina a los ficheros de imagen, las vistas y los ficheros donde se detallan colores, estilos y los textos.

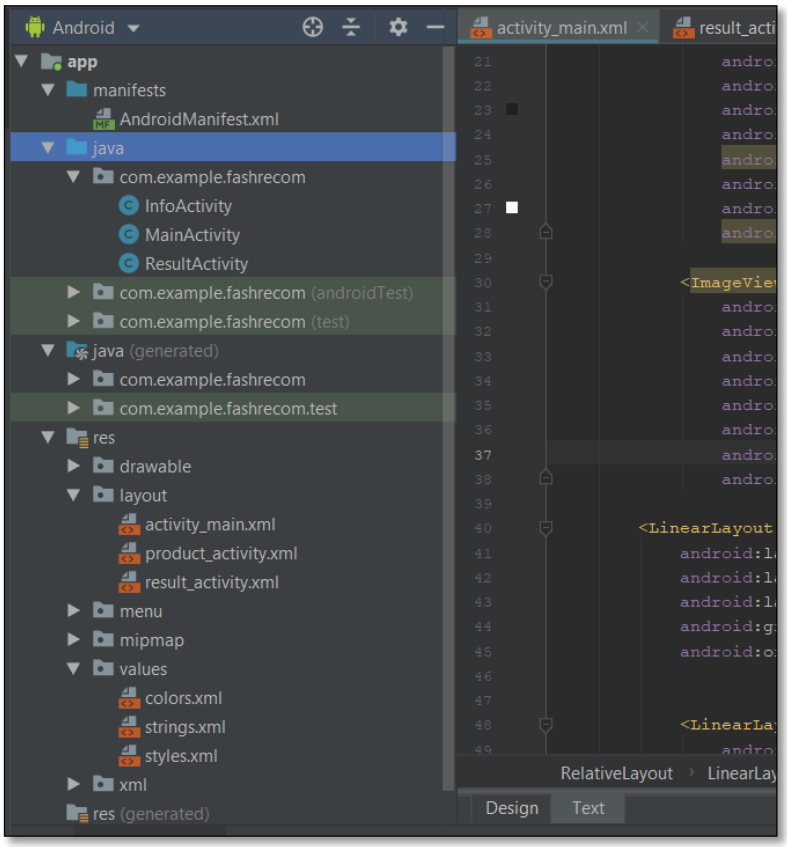


Figura 48. Secuencia de mensajes emitidos durante el funcionamiento del servidor (Diseño propio).

Durante su desarrollo se seguirá el patrón MVC y dado que se dispone de los bocetos elaborados en el capítulo 6, la vista intentará parecerse lo máximo posible a estos bocetos. Este es el primer elemento que nos encargaremos de desarrollar.

Las vistas o pantallas en Android se denominan *Activities*, cada vista se edita desde su propio fichero con extensión XML (*Extensible Markup Language*). Disponemos de dos vistas: la vista principal de bienvenida a la aplicación y la vista de con las etiquetas del artículo y las recomendaciones.

Al igual que el boceto la vista principal se muestra organiza los elementos superiores y centrales a modo de tutorial de la aplicación, en su parte inferior se disponen dos botones uno de acceso a la cámara y otro de acceso a la galería, por último se coloca una imagen a modo de espacio publicitario. La segunda vista permitirá mostrar la información relativa a la imagen que se suba desde la vista anterior y predispone 3 elementos de imagen, en los que se mostrarán las imágenes recibidas desde el servidor.

Véase la siguiente figura con la interfaz final conseguida para la aplicación.

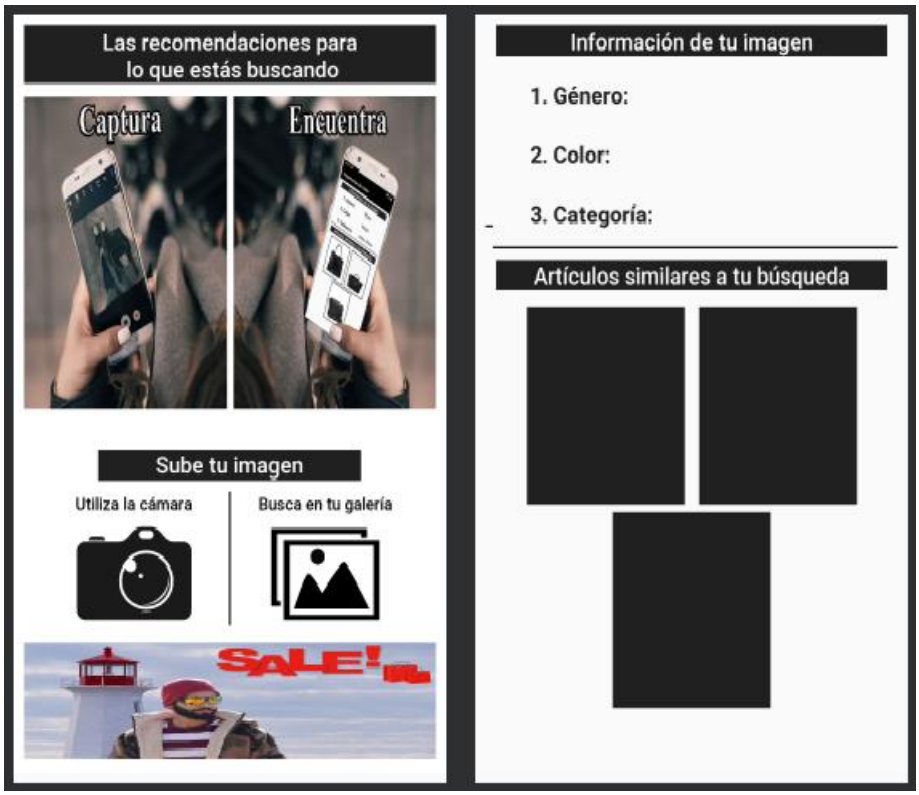


Figura 49. Resultados de las vistas creadas para la interfaz de la aplicación Android (Diseño propio).

Creadas las vistas, es el momento de desarrollar el controlador. Este juega un papel fundamental en la aplicación pues es el elemento que gestiona las interacciones del usuario con la aplicación, de modificar los elementos de la vista y de conectarse con el modelo (Servidor) para el envío/recepción de los datos.

El controlar requiere de dos clases Java, una para cada una de las vistas. El primero (*MainActivity.java*) se encarga de controlar las acciones que realice el usuario para subir la imagen ya lo haga mediante el botón de la cámara o la galería, establecerá la comunicación con el servidor y realizará con este el intercambio de datos (envío de imagen, recepción de etiquetas y de imágenes de recomendación), una vez finalice su cometido dará paso al siguiente controlador.

El controlador de la vista de las recomendaciones cambia la vista de la aplicación a la propia de las recomendaciones recibe como argumentos, por parte del anterior controlador, las etiquetas que corresponden a la imagen subida. Además se encarga de establecer las imágenes recibidas mediante la ruta de almacenamiento destinada a las mismas.

```
public class ResultActivity extends AppCompatActivity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.result_activity);

        //Sacamos las etiquetas recibidas durante la creacion del intent
        Intent intent = getIntent();
        String etiquetas = intent.getStringExtra( name: "etiquetas");
```

Figura 50. Activación de la vista y obtención de las etiquetas en el segundo controlador (Diseño propio).

7.6. Integración y pruebas del sistema

La combinación o unión de las distintas aplicaciones o subsistemas generados (cliente y servidor) dará lugar al sistema completo, esto es lo que se define como integración. Este proceso busca adaptar el funcionamiento de los componentes del sistema para que puedan trabajar de forma funcional entre ellos.

Existen por lo general algunas dificultades para realizar la integración si se utilizan diferentes tecnologías. En este caso la integración deberá permitir conectar una aplicación Android con un programa Python ejecutado en un ordenador con Windows 10. Dado que no se dispone de un servidor alojado en la nube integraremos ambas aplicaciones para trabajar en una red local.

La solución escogida ha sido utilizar **sockets**. Desde un fichero de configuración en la aplicación móvil se indica la dirección IP y el puerto al que se debe conectar, por el lado del servidor el fichero indica el puerto de apertura que se abrirá para recibir conexiones de clientes.

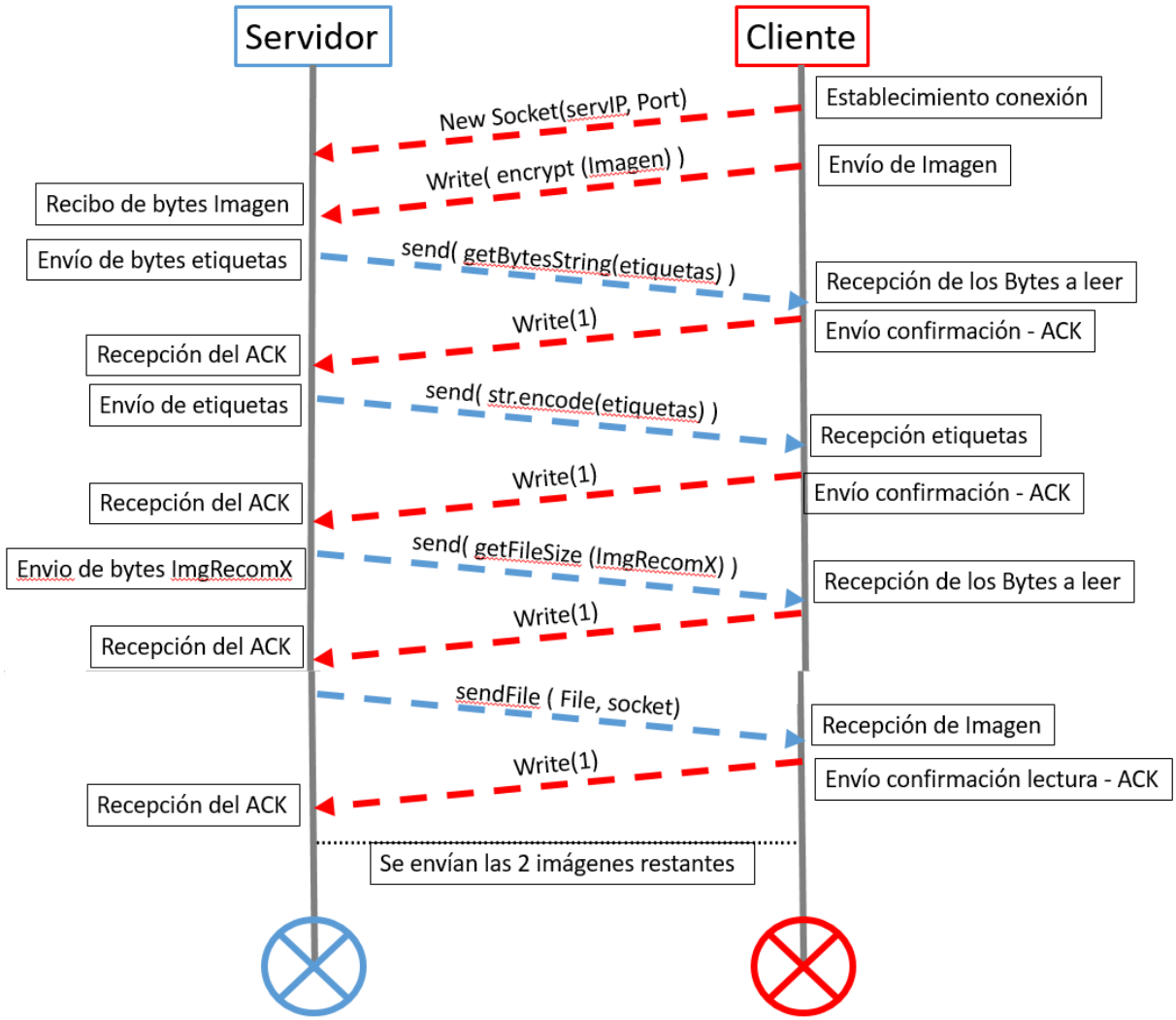


Figura 51. Flujo de mensajes entre los componentes del sistema (Diseño propio).

Para seguir el flujo de mensajes anterior, se han de tener en cuenta algunas limitaciones en la implementación de las librerías de los sockets para cada lenguaje. Por parte de Python la recepción/envío por sockets acepta únicamente bytes, sin embargo desde Java podemos escoger entre algunos tipos de datos (*String, Int o Byte*) para realizar la lectura o envío.

Garantizar una correcta integración en ambos subsistemas supone que debamos trabajar con bytes en ambos lenguajes. Aprovechando el envío de la información en bytes y con objeto de garantizar privacidad al uso que hacen los usuarios del sistema de recomendaciones la imagen que envíe desde el cliente se enviará encriptada.

En términos de seguridad existen multitud de algoritmos de cifrado. Haremos uso de AES-128 (*Advanced Encryption Standard*) el sucesor del cifrado estándar DES. Este algoritmo se basa en sustituciones, permutaciones y transformaciones operadas sobre bloques de 16 bytes o lo que es lo mismo 128 bits. Se simplificará el proceso de estableciendo inicialmente de una clave, tras generar una clave aleatoria esta se ha establecido para el cliente y el servidor.

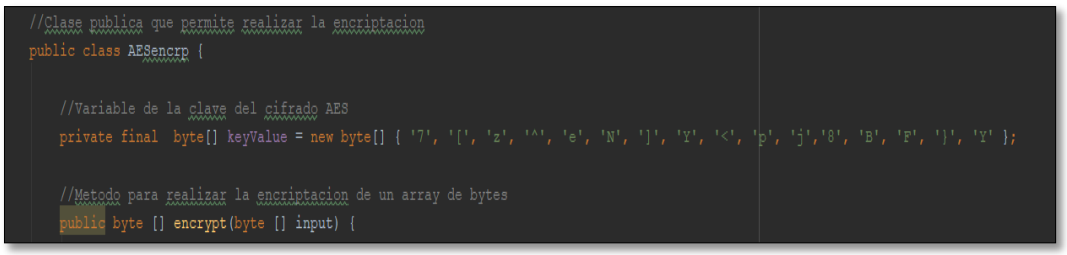


Figura 52. Clase de Java que almacena una variable con la clave (Diseño propio).

Una vez finalizada la integración se podrá observar la secuencia de mensajes de la figura 47 por el lado del servidor, así como las imágenes y etiquetas en la aplicación móvil. Para confirmar que las recomendaciones pueden considerarse como buenas y que el funcionamiento del cliente/servidor también lo es, se detallarán a continuación una serie de pruebas de funcionamiento.

Las pruebas a realizar serán pruebas unitarias y pruebas de simulación real para comprobar el funcionamiento del sistema. Debido a que no disponemos de campos de introducción de texto o validación se realizará una prueba unitaria para comprobar que es posible establecer una conexión con el servidor, tal y como se aprecia en la siguiente figura.

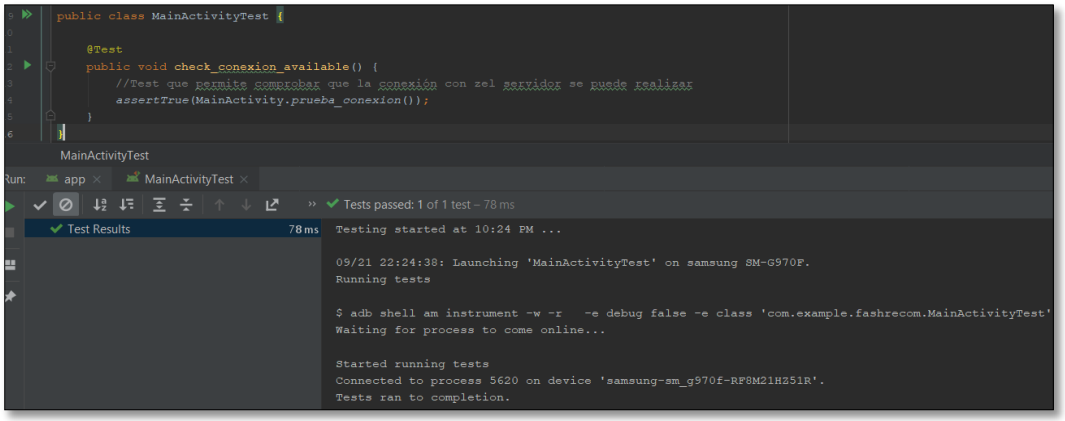


Figura 53. Resultado positivo tras ejecutar la prueba de conexión (Diseño propio).

Las pruebas de simulación consisten en escoger entre uno de los dos métodos de selección de la imagen, ya sea mediante una fotografía realizada desde la cámara o una imagen seleccionada desde la galería. Se producirá el intercambio de datos entre cliente y servidor, de forma que el controlador de la aplicación prepare los datos recibidos para que los visualicemos.

La siguiente tabla muestra una prueba de simulación completa. Los resultados como se puede apreciar corresponden con lo esperado para el objeto fotografiado tanto en género, como en color y categoría. Podemos considerar que esta prueba ha sido superada con éxito. El resto de pruebas de simulación se encuentran en los apéndices (véase el apéndice 10.4)

Opción escogida	Imagen utilizada/capturada	Resultados obtenidos
		

Tabla 29. Pruebas y resultados de la simulación número 1 (Diseño propio).

7.7. Preparación del despliegue

Concluido el desarrollo y validada la calidad del sistema tras las pruebas realizadas es momento de desplegar el sistema. No realizaremos una transición entre versiones del sistema sino un primer despliegue. El despliegue del sistema consiste en poner en funcionamiento el servidor y en disposición de los usuarios la posibilidad de hacer uso del cliente de Android.

El servidor debe disponer del directorio con los ficheros y carpetas mencionadas en el apartado 7.3 de este capítulo, tras esto simplemente se debe poner en ejecución el fichero que gestiona las conexiones, las recomendaciones y el envío de datos.

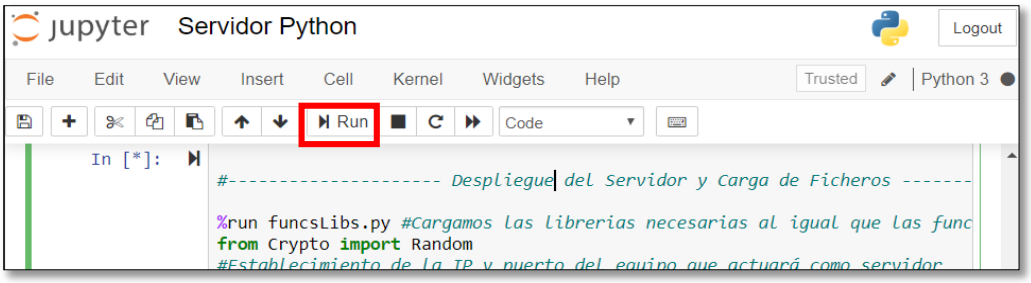


Figura 54. Despliegue del servidor (Diseño propio).

Desplegar una aplicación en el sistema operativo Android es un proceso que es posible completar de una estas dos formas:

1. **Despliegue físico**: El despliegue físico requiere de un ordenador con Android Studio instalado y de tener el propio proyecto de la aplicación abierto. El despliegue se realiza desde un simple botón una vez se conecta en modo desarrollador un dispositivo con Android al equipo.

2. **Despliegue digital**: Se genera único archivo de instalación desde Android Studio. Este fichero se puede facilitar a los usuarios de forma digital mediante plataformas de distribución de aplicaciones o mediante un enlace en el caso de que suba a una plataforma de alojamiento de datos.

Optaremos por la segunda, generaremos un fichero de instalación o APK (*Android Package*) desde el proyecto, de esta forma el despliegue será mucho más sencillo de completar.

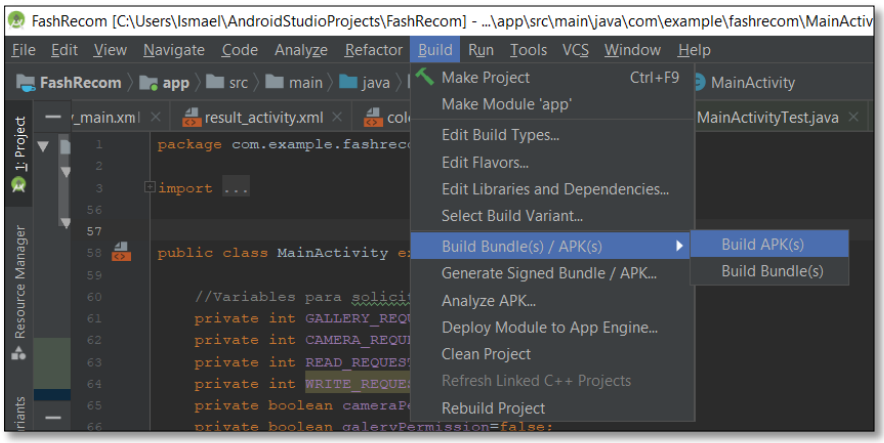


Figura 55. Generación del fichero APK de la aplicación (Diseño propio).

8. Conclusiones

La propuesta de este trabajo consistía en desarrollar un sistema de recomendaciones de productos de moda basado en el uso de imágenes. Para desarrollar este sistema no se partía de ningún material ya desarrollado, tan solo de una fuente de imágenes. Finalmente es posible afirmar que el resultado de la implementación y la integración de las diferentes partes que componen el sistema desarrollado ha sido un éxito. Se ha logrado obtener un sistema funcional, operativo en tiempo real y a su vez práctico.

Parte de la curiosidad personal que me ha motivado a trabajar en un proyecto con imágenes, la despertaron asignaturas cursadas como lo fueron "Visión por Computador" o "Procesamiento de Imágenes y Vídeo". La motivación por hacerlo con grandes cantidades de datos "Diseño y Almacenes de Datos" y la de conocer la información que esconden los datos "Métodos estadísticos para la computación". Este trabajo ha requerido además de una formación más profunda en Python y en las redes neuronales, incluso de formación complementaria en Android. De igual forma el resultado permite demostrar como la formación que recibimos debe seguir siendo diversa, pues este proyecto ha requerido como base el conocimiento obtenido en las asignaturas citadas. Asignaturas cuyos ámbitos de estudio son aparentemente muy distintos.

Haber podido completar el proyecto ha sido posible gracias a dos hechos fundamentales: el primero ha sido haber elaborado una correcta planificación temporal y el segundo poder disponer de una fuente de datos que se adecuaba al problema.

En la planificación temporal de cualquier proyecto, no se debería escatimar en incluir días suficientes para realizar una correcta formación que permita adaptarse, conocer y probar la tecnología con la que se desea trabajar. Aunque de forma personal había trabajado con redes neuronales en Python, no ocurría lo mismo con Android, una tecnología que siempre había querido explorar. Sin el periodo de formación incluido en la planificación, comenzar a desarrollar la aplicación hubiese implicado retrasos continuos en el resto de tareas del proyecto.

Como otro hecho fundamental, destacar que en muchos proyectos de aprendizaje profundo como este en particular, una parte esencial antes de comenzar a desarrollar o pensar en entrenar las redes neuronales y que puede comprometer seriamente calidad del resultado final, es no disponer de un buen conjunto de datos. Entendiendo un buen conjunto como un conjunto que presenta estas dos propiedades:

- Ser heterogéneo: Las diversidad en las imágenes disponibles permite que la red generada sea capaz de aprender cuáles son los verdaderos rasgos que diferencian el valor que debe tomar el atributo.

- Estar debidamente etiquetado: Si la calidad del etiquetado de la información es buena podemos dar libertad a que la red comience a entrenar.

Existen algunos otros *datasets* de artículos de moda en internet. Algunos de ellos con una mayor cantidad de imágenes o con imágenes de usuarios reales no solo de modelos, ni de los productos sobre un fondo que permita distinguirlos con claridad. Aparentemente podríamos inducir al pensamiento erróneo de que las redes podrían presentar mejores resultados con este tipo de conjuntos. No obstante, el hecho de no cumplir ambas propiedades anteriores lo hubiese

impedido. Al ser los valores de la mayoría de etiquetas incorrectos o incompletos, por muy buena que fuese la red, el resultado del aprendizaje realizado aplicado a imágenes reales hubiese con seguridad peor que el actual.

El sistema desarrollado ha demostrado no solo ser capaz de ofrecer respuestas a recomendaciones en tiempos inferiores a algunos de los sistemas actuales sino que incluso en muchas ocasiones es más acertado en las recomendaciones ofrecidas. Este tipo de búsquedas mediante imágenes son realmente cómodas de realizar. Si se desarrollasen más modelos de redes neuronales para complementar a las actuales y detectar así otros rasgos de las prendas como lo son la estacionalidad, el estampado, la textura etc. se podrían facilitar recomendaciones aún más precisas.

Otra posible e interesante ampliación para este trabajo consistiría en poder realizar la detección de la prenda en la imagen, antes de que las redes neuronales que detecten los atributos la procesen. Con esto se persigue, poder recortar automáticamente el producto de la imagen y analizar únicamente la zona en la que aparezca el producto. Esto supondría una gran mejoría en la calidad de las recomendaciones ya que los modelos analizarían la parte en la que se encuentra ese producto y no derivarían el resultado a un error por el resto de elementos que aparecen en la imagen o por el resto de colores de los elementos en la imagen.

Puedo concluir afirmando sin temor a equivocarme que el futuro de las búsquedas de productos pasará por hacer uso de las imágenes, rompiendo así el estigma de utilizar las tradicionales búsquedas por texto.

9. Referencias bibliográficas

Aston, B. (2019). 161 Inspiradoras Frases Para Los Gerentes De Proyectos. Recuperado el 17 de Junio del 2019 de: thedigitalprojectmanager.com/es/gestion-proyectos-citas-inspiracion/

Badiru, A. B., Rusnock, C. F., Valencia, V. V. (2018). *Project Management for Research: A Guide for Graduate Students*. CRC Press. ISBN: 9781315360102.

Balado, E. S. (2005). *La Nueva Era Del Comercio/the New Era of Commerce: El Comercio Electronico, Las Tic's Al Servicio De La Gestion Empresarial.* Ideaspropias Editorial SL. ISBN: 978-8493454722

Bello, C. (s.f.). *250 Frases de Empresarios exitosos*. Recuperado el 2 de Julio de 2019 de: books.google.es/books?id=ulWQDwAAQBAJ&lpg=PP1&hl=es&pg=PT57#v=onepage&q&f=false

Cardoso, A., Daolio, F., Vargas, S. (2018). Product Characterisation towards Personalisation. Learning Attributes from Unstructured Data to Recommend Fashion Products. En *Proceedings of the 24th ACM SIGKDD* International *Conference on Knowledge Discovery & Data Mining*, pp. 80-89. Recuperado el 15 de Junio de 2019 de: arxiv.org/pdf/1803.07679.pdf

Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep learning*. The MIT Press. ISBN: 9780262035613.

González J. , Javier. (2016). *Image Processing* [Material docente].Universidad de Málaga.

Google Trends (2019). Interés a lo largo del tiempo sobre "Convolutional Neural Networks" Recuperado el 6 de Julio del 2019 de: trends.google.com/trends/explore?date=today%205-y&q=convolutional%20neural%20networks

Gringer (2007). *SVG version of Overfitting.png by Dake. Created from scratch using Inkscape v0.45.* Recuperado el 20 de Junio del 2019 de: commons.wikimedia.org/wiki/File:Overfitting_svg.svg

Hilera R., J, Martínez J., V. (1995). *Redes neuronales artificiales. Fundamentos, modelos y aplicaciones*. Ra-Ma. ISBN: 9788478971558

Hipocrates (s.f). Recuperado el 22 de Junio del 2019 de: https://gruptrobat.com/cursos-y-talleres/neurociencia/

LeCun, Y. et al.(1998). *Gradient-Based Learning Applied to Document Recognition*. Recuperado el 23 de Junio del 2019 de: yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Lévy Mangin, J.P., Flórez López, R., Fernández-Fernández, J.M. (2008*). Las redes neuronales artificiales*. NetBiblo. ISBN: 9788497452465.

Mayranna (2013). Perceptron. Recuperado el 17 de Junio del 2019 de: commons.wikimedia.org/wiki/File:Perceptron_moj.png

Miller, D. (2018). *How is E-Commerce Using AI to Make Recoomendations?*. Recuperado el 23 de Junio del 2019 de : exponea.com/blog/e-commerce-using-ai-make-recommendations/

Photographer (2006). *Notación caso de uso.* Recuperado el 10 de Junio del 2019 de : commons.wikimedia.org/wiki/File:Notacion_Caso_de_Uso.png

Piatetsky, G. (2014). CRISP-DM, still the top methodology for analytics, data mining, or data science projects. Recuperado el 4 de junio del 2019 de: www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.htm

Portilla, J., Francesco, M. (2018). *Deep Learning with Python and Keras* [Material del curso]

Prabhu (2018). Understanding of Convolutional Neural Network (CNN) - Deep Learning. Recuperado el 27 de Julio del 2019 de : https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Prieto, A. et al. (2016). Neural networks: An overview of early research, current frameworks and new challenges. *Neurocomputing*, (214), pp. 242-268. Recuperado de: www.sciencedirect.com/science/article/pii/S0925231216305550?via%3Dihub

Real Academia Española (2014). Definición del término "comercio" en diccionario de la lengua española (23.ª ed.). Recuperado el 20 de Junio de 2019 de: dle.rae.es/?id=9vYPFME

Rodriguez, O. (2010). Metodología para el desarrollo de proyectos en minería de datos CRISP-DM en Gallardo, J.A. (2009). Metodología para la definición de requisitos en proyectos de data mining (Tesis doctoral). Universidad Politécnica de Madrid, España. Recuperado el 19 de Junio del 2019 de: www.oldemarrodriguez.com/yahoo_site_admin/assets/docs/Documento_CRISP-DM.2385037.pdf

Salcedo, L. (2018). *Introducción a las Redes Neuronales - Parte #1: Elementos básicos de una Red Neuronal*. Recuperado el 23 de Junio del 2019 de : www.pythondiario.com/2018/07/introduccion-las-redes-neuronales-parte_12.html

Saltz, J., Shamshurin, I., Crowston, K. (2017). *Comparing Data Science Project Management Methodologies via a Controlled Experiment*. En *Proceedings of the 50th Hawaii International Conference on System Sciences*. Recuperado el 27 de Junio del 2019 de: core.ac.uk/download/pdf/77239583.pdf

Seoane, E. B. (2005). *La nueva era del comercio: El comercio electrónico. Las TIC al servicio de la gestión empresarial*. Vigo, España: Ideaspropias Editorial.

The History of Machine Learning (s.f.). Recuperado el 4 de Junio de 2019 de: https://www.bbc.com/timelines/zypd97h

Train a MobileNet Wihthing 60 lines of Code (2019). Recuperado el 17 de Junio del 2019 de : tensorflowlitemobile.com/2019/01/14/train-a-mobilenet-within-60-lines-of-code/

Vincent C., Kevin C. (2002). An Introduction to Neural Networks [Material Docente]. Universidad de Manitoba, Canada. Recuperado el 10 de Julio del 2019 de: www2.econ.iastate.edu/tesfatsi/NeuralNetworks.CheungCannonNotes.pdf

Ujjwalkarn (2016). An Intuitive Explanation of Convolutional Neural Networks. Recuperado el 12 de Julio del 2019 de: ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Herramientas, Software y Datos

Alex Lenail (2019). NN-SVG [Herramienta Web]. Recuperado el día 15 de Julio del 2019 de: alexlenail.me/NN-SVG/LeNet.html

Anaconda (2019). Anaconda Distribution (3.7) [Suite de Software]. Recuperado el día 15 de Junio del 2019 de: www.anaconda.com/distribution/

GanttProject Team (2019). GanttProject (2.8.10) [Software]. Recuperado el día 1 de Junio del 2019 de: www.ganttproject.biz/

Google (2019). Android Studio (3.5) [Software Desarrollo]. Recuperado el día 21 de Julio del 2019 de: developer.android.com/studio#downloads

No Magic, Inc (2019). MagicDraw (19.0) [Software]. Recuperado el día 11 de Junio del 2019 de: www.magicdraw.com/download/magicdraw

Param Aggarwal (2019). Fashion Product Images (Small) [Dataset]. Recuperado el día 20 de Junio del 2019 de: https://www.kaggle.com/paramaggarwal/fashion-product-images-small/downloads/fashion-product-images-small.zip/1

Param Aggarwal (2019). Fashion Product Images Dataset [Dataset]. Recuperado el día 20 de Junio del 2019 de: https://www.kaggle.com/paramaggarwal/fashion-product-images-dataset

S.C Evercoder Software S.R.L (2019). Moqups [Herramienta Web]. Recuperado el día 13 de Junio del 2019 de: moqups.com/favicon.png

10. Anexos

10.1. Listado de requisitos

Nombre	**RF-01:** Subir nuevo conjunto de datos
Versión	1.0
Actor	Administrador
Descripción	• Se debe permitir el almacenamiento de nuevos conjuntos de datos en el sistema. Estos se pueden destinar a los resultados de las recomendaciones o al entrenamientos/validación de las redes neuronales.

Tabla 14. Requisito funcional número uno del sistema (Diseño propio).

Nombre	**RF-02:** Borrar conjunto de datos
Versión	1.0
Actor	Administrador
Descripción	• El administrador debe disponer de permisos de borrado sobre las imágenes destinadas al entrenamientos y validación de las redes neuronales, así como para las destinadas a las recomendaciones.

Tabla 15. Requisito funcional número dos del sistema (Diseño propio).

Nombre	**RF-03:** Subir modelos
Versión	1.0
Actor	Administrador
Descripción	• El administrador debe disponer de permisos de para realizar la subida de nuevos modelos ya entrenados a la carpeta donde se almacenan los modelos de producción.

Tabla 16. Requisito funcional número tres del sistema (Diseño propio).

Nombre	**RF-04:** Actualizar modelos
Versión	1.0
Actor	Administrador
Descripción	• El administrador debe disponer de permisos para actualizar los modelos existentes. Podrá sustituir modelos existentes

	mediante modelos ya generados o actualizar los existentes en términos de arquitectura o mediante la realización de nuevos entrenamientos.

Tabla 17. Requisito funcional número cuatro del sistema (Diseño propio).

Nombre	**RF-05:** Modificar servidor
Versión	1.0
Actor	Administrador
Descripción	• El administrador puede modificar si lo desea, el fichero del funcionamiento del servidor. Permitiendo cambios en los puertos, tipos de cifrado, mensajes emitidos…

Tabla 18. Requisito funcional número cinco del sistema (Diseño propio).

Nombre	**RF-06:** Solicitar recomendación
Versión	1.0
Actor	Usuario
Descripción	• Desde la aplicación móvil el usuario será capaz de subir una imagen. La foto será enviada al servidor para que allí se generen las etiquetas y recomendaciones, utilizando para ello las redes neuronales disponibles.

Tabla 19. Requisito funcional número seis del sistema (Diseño propio).

Nombre	**RF-07:** Generar recomendación
Versión	1.0
Actor	Usuario
Descripción	• A petición del usuario en la aplicación móvil el servidor puede recibir solicitudes de conexión para generar recomendaciones de moda. • Estas recomendaciones comienzan con la recepción de una imagen cifrada, la cual se descifra y se pasa a las redes neuronales para generar las etiquetas del producto. • Mediante las etiquetas se procede a la búsqueda de productos con las misma etiquetas para que estas se envíen de vuelta a la aplicación.

Tabla 20. Requisito funcional número siete del sistema (Diseño propio).

Nombre	**RF-08:** Ejecutar modelos
Versión	1.0
Actor	Usuario
Descripción	• A petición del usuario que solicite una recomendación, el sistema ejecutará los modelos usando la foto recibida con objeto de poder obtener las etiquetas del mismo.

Tabla 21. Requisito funcional número ocho del sistema (Diseño propio).

Nombre	**RF-09:** Modificar *Front-End*
Versión	1.0
Actor	Administrador
Descripción	• El administrador dispone de permisos para modificar la interfaz visual (UI) con objeto de poder cambiar cumplir con patrones de diseño, adaptarla a todos los dispositivos, añadir botones para nuevas funcionalidades etc.

Tabla 22. Requisito funcional número nueve del sistema (Diseño propio).

Nombre	**RF-10:** Modificar *Back-End*
Versión	1.0
Actor	Usuario
Descripción	• El administrador dispone de permisos para modificar la lógica del controlador de la aplicación móvil. Se le permite por lo tanto mejorar la seguridad, realizar mejoras de rendimiento etc

Tabla 23. Requisito funcional número diez del sistema (Diseño propio).

Nombre	**RNF-01:** Garantizar seguridad durante la recomendación
Versión	1.0
Descripción	• La imagen facilitada por el usuario para obtener la recomendación debe enviarse cifrada al servidor, garantizando así la confidencialidad e integridad de la misma.

Tabla 24. Requisito no funcional número uno del sistema (Diseño propio).

Nombre	**RNF-02:** Controlar la conexión entre aplicación y servidor
Versión	1.0
Descripción	• La conexión entre ambas partes se debe limitar desde el inicio de la solicitud de recomendación hasta el envío de la confirmación de llega de la última foto por parte de la aplicación móvil.

Tabla 25. Requisito no funcional número dos del sistema (Diseño propio).

Nombre	**RNF-03:** Utilizar una interfaz de usuario clara y sencilla
Versión	1.0
Descripción	• La interfaz de la aplicación debe explicar y facilitar al usuario una rápida adaptación para un correcto uso.

Tabla 26. Requisito no funcional número tres del sistema (Diseño propio).

Nombre	**RNF-03:** Tiempos de respuesta breves
Versión	1.0
Descripción	• La respuesta en segundos que genere el servidor tras la recepción de la solicitud de recomendación por parte de la aplicación debe ser muy baja.

Tabla 27. Requisito no funcional número tres del sistema (Diseño propio).

10.2. Arquitectura de la plantilla de red neuronal

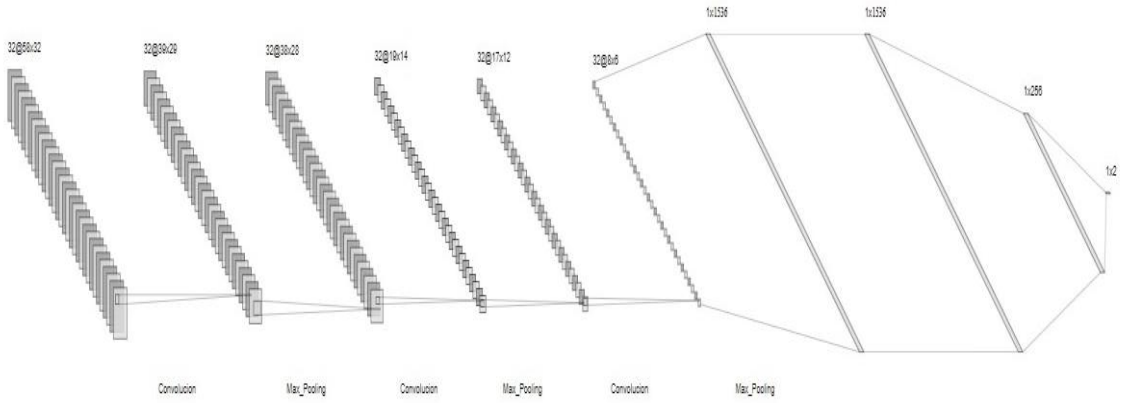
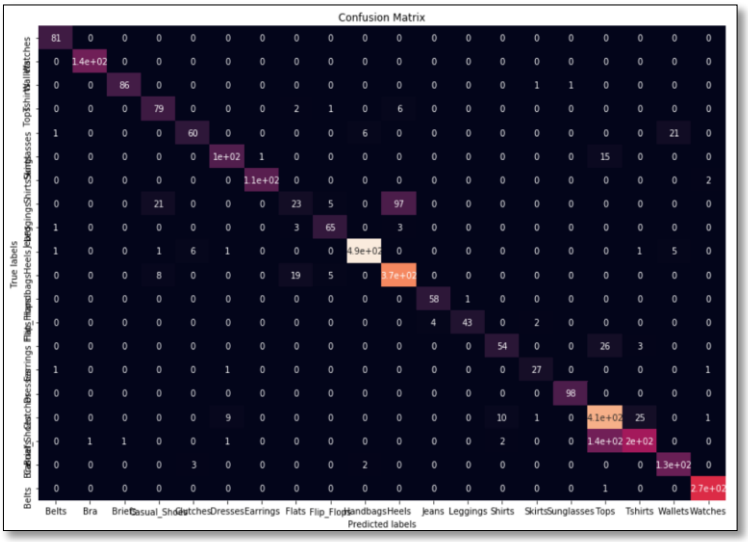


Figura 56. Arquitectura de la plantilla de red neuronal diseñada desde la primera capa (Diseño propio).

10.3. Resultados estadísticos de las redes neuronales

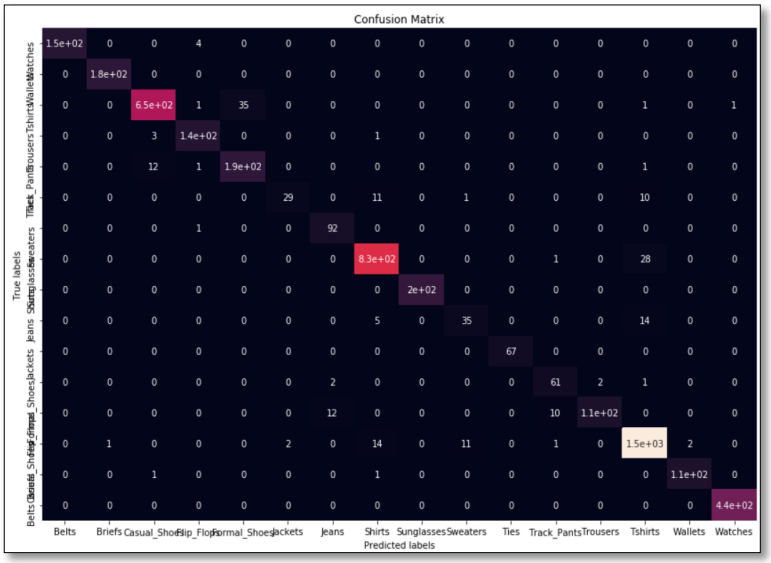
Modelo	Resultados
modelo_genero	---- Red Neuronal para el Genero ---- Loss: 0.22270759548488223 Accuracy: 0.9079004552848555 Confusion Matrix [[4789 355] [456 3372]] Classification Report precision recall f1-score support Men 0.91 0.93 0.92 5144 Women 0.90 0.88 0.89 3828 accuracy 0.91 8972 macro avg 0.91 0.91 0.91 8972 weighted avg 0.91 0.91 0.91 8972

modelo_articulos_mujer




```
---- Red Neuronal para los articulos de Mujeres ----

Loss:  0.34802346056638445
Accuracy:  0.8610743289087973

Classification Report

                precision    recall  f1-score   support

         Belts       0.95      1.00      0.98        81
           Bra       0.99      1.00      1.00       140
        Briefs       0.99      0.98      0.98        88
  Casual_Shoes       0.72      0.90      0.80        88
      Clutches       0.87      0.68      0.76        88
       Dresses       0.90      0.87      0.88       120
      Earrings       0.99      0.98      0.99       114
         Flats       0.49      0.16      0.24       146
    Flip_Flops       0.86      0.90      0.88        72
      Handbags       0.98      0.97      0.98       502
         Heels       0.78      0.92      0.84       398
         Jeans       0.94      0.98      0.96        59
      Leggings       0.98      0.88      0.92        49
        Shirts       0.82      0.65      0.72        83
        Skirts       0.87      0.90      0.89        30
    Sunglasses       0.99      1.00      0.99        98
          Tops       0.69      0.90      0.78       457
       Tshirts       0.87      0.58      0.69       346
       Wallets       0.84      0.96      0.90       138
       Watches       0.99      1.00      0.99       271

      accuracy                           0.86      3368
     macro avg       0.87      0.86      0.86      3368
  weighted avg       0.86      0.86      0.85      3368
```

modelo_articulos_hombre




```
---- Red Neuronal para los articulos de Hombres ----

Loss:  0.1306225761303621
Accuracy:  0.9617480540313376

Classification Report

              precision    recall  f1-score   support

       Belts       1.00      0.97      0.99       157
      Briefs       0.99      1.00      1.00       177
Casual_Shoes       0.98      0.94      0.96       684
  Flip_Flops       0.95      0.97      0.96       149
Formal_Shoes       0.84      0.93      0.88       200
     Jackets       0.94      0.57      0.71        51
       Jeans       0.87      0.99      0.92        93
      Shirts       0.96      0.97      0.96       859
  Sunglasses       1.00      1.00      1.00       199
    Sweaters       0.74      0.65      0.69        54
        Ties       1.00      1.00      1.00        67
 Track_Pants       0.84      0.92      0.88        66
    Trousers       0.98      0.83      0.90       132
     Tshirts       0.96      0.98      0.97      1524
     Wallets       0.98      0.98      0.98       114
     Watches       1.00      1.00      1.00       436

    accuracy                           0.96      4962
   macro avg       0.94      0.92      0.93      4962
weighted avg       0.96      0.96      0.96      4962
```

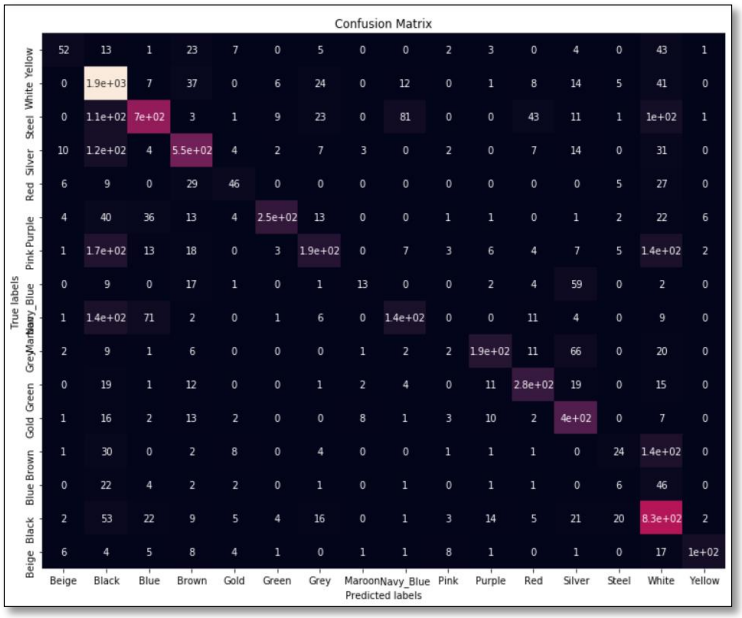
modelo_color	 ---- Red Neuronal para el color del articulo ---- Loss: 1.2594751228859868 Accuracy: 0.6217878395838188 Classification Report precision recall f1-score support Beige 0.41 0.25 0.31 154 Black 0.64 0.90 0.75 2035 Blue 0.73 0.67 0.70 1087 Brown 0.61 0.75 0.68 754 Gold 0.30 0.35 0.33 122 Green 0.85 0.51 0.64 394 Grey 0.47 0.07 0.12 568 Maroon 0.31 0.08 0.13 108 Navy_Blue 0.52 0.28 0.37 380 Pink 0.73 0.32 0.45 311 Purple 0.95 0.43 0.60 365 Red 0.62 0.83 0.71 463 Silver 0.24 0.03 0.05 212 Steel 0.00 0.00 0.00 86 White 0.50 0.74 0.60 1007 Yellow 0.72 0.68 0.70 161 accuracy 0.62 8207 macro avg 0.54 0.43 0.45 8207 weighted avg 0.61 0.62 0.58 8207

Tabla 28. Resultados obtenidos tras el simulación con el conjunto de prueba (Diseño propio).

10.4. Resultados de pruebas de simulación del sistema

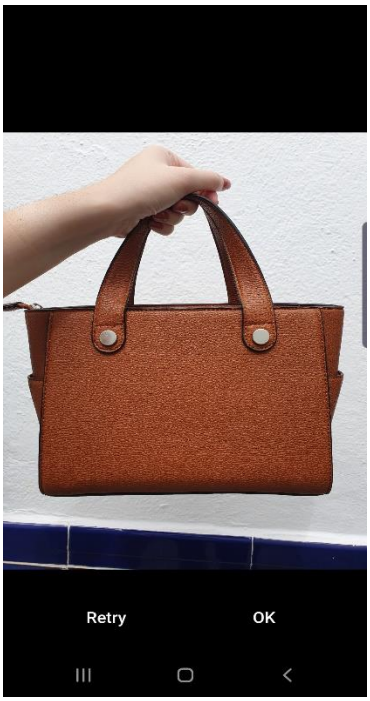
Opción escogida	Imagen utilizada/capturada	Resultados obtenidos
		

Tabla 29. Pruebas y resultados de la simulación número 1 (Diseño propio).

Opción escogida	Imagen utilizada/capturada	Resultados obtenidos
		

Tabla 30. Pruebas y resultados de la simulación número 2 (Diseño propio).

Opción escogida	Imagen utilizada/capturada	Resultados obtenidos
		

Tabla 31. Pruebas y resultados de la simulación número 3 (Diseño propio).

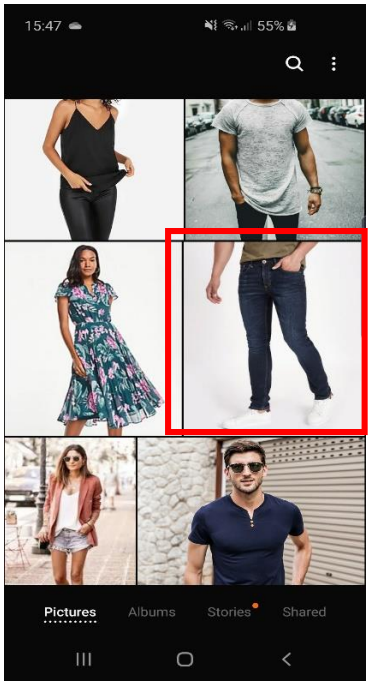
Opción escogida	Imagen utilizada/capturada	Resultados obtenidos
		

Tabla 32. Pruebas y resultados de la simulación número 4 (Diseño propio).

10.5. Manual de instalación

La puesta a prueba del sistema en un nuevo escenario requiere en primer lugar de la instalación los dos siguientes elementos:

- **Anaconda**: Esta suite instala Python en el equipo así como otros IDEs. En especial requeriremos de Jupyter Notebook para ejecutar el servidor y de la instalación previa de algunos paquetes.

- **Aplicación móvil**: Desde un dispositivo móvil que opere con el S.O. Android en una versión Android 4.4 o superior, realizaremos la instalación de la aplicación.

Para comenzar la **instalación de Anaconda** es necesario acudir al enlace web indicado en las referencias. El enlace permite acceder al panel de descargas de su página web. Una vez allí se debe descargar la versión de Anaconda con Python 3.7 acorde al S.O sobre el que se instalará.

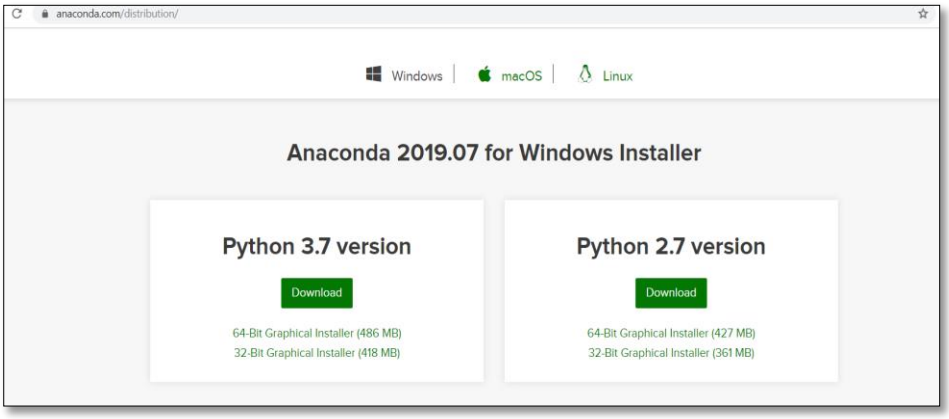


Figura 57. Entorno web para realizar la descarga de Anaconda (Diseño propio).

Completada la descarga del fichero, ya sea mediante la ejecución del terminal o mediante un doble clic del fichero descargado comenzamos a instalar Anaconda.

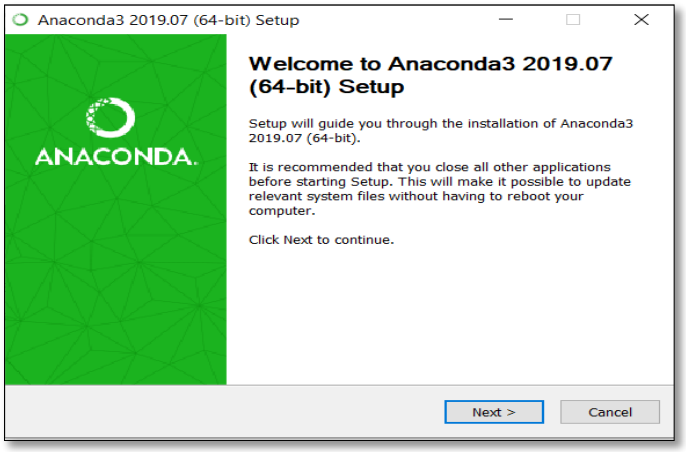


Figura 58. Ventana de instalación de Anaconda en el S.O. Windows (Diseño propio).

Se deberá leer los términos y condiciones de uso de la licencia de Anaconda. En caso de estar de acuerdo y para continuar con la instalación hay que aceptarlos.

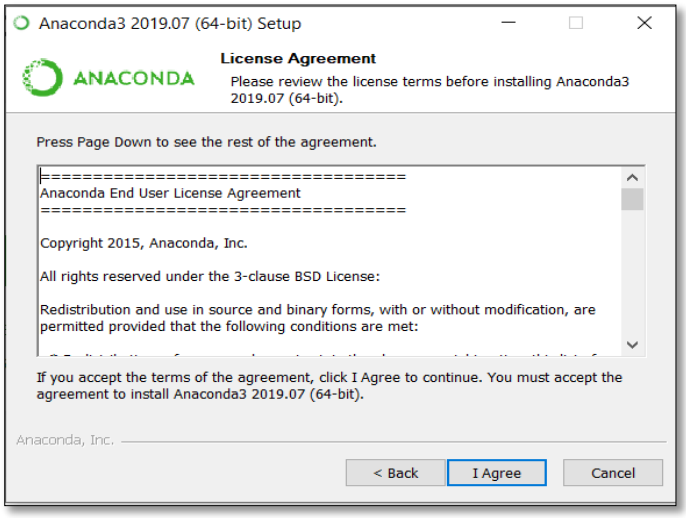


Figura 59. Ventana de términos y condiciones de la instalación de Anaconda (Diseño propio).

Es necesario indicar un directorio nuevo o utilizar el indicado por defecto como lugar de instalación de Anaconda. Una vez completada la elección del directorio se procederá a realizar la instalación.

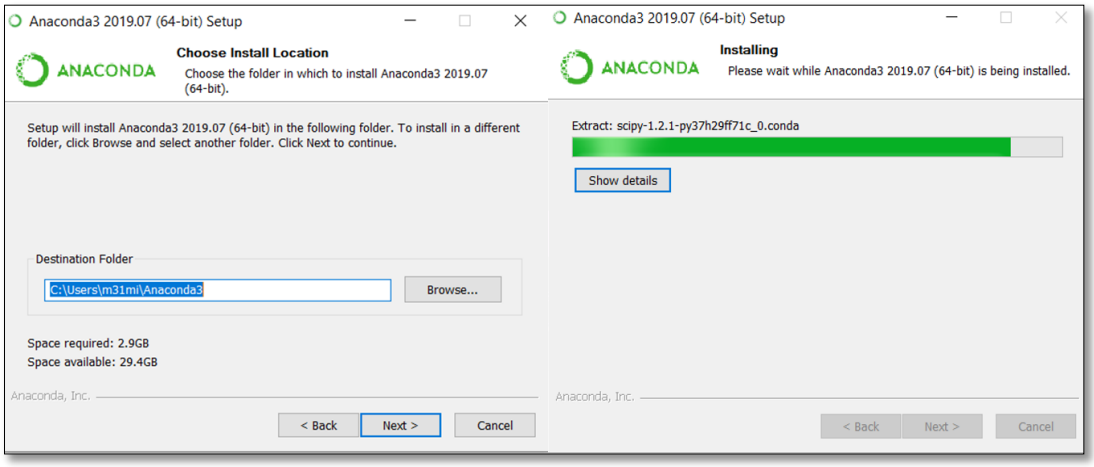


Figura 60. Elección del directorio de instalación y progreso de instalación (Diseño propio).

Completada la instalación, abriremos desde el panel de aplicaciones Anaconda Prompt para instalar los paquetes necesarios que permitan ejecutar correctamente el servidor.

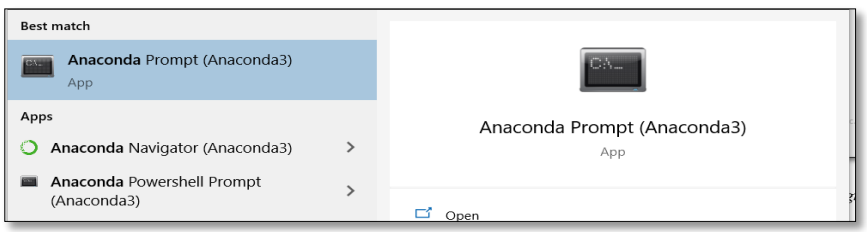


Figura 61. Símbolo del sistema para Anaconda Prompt (Diseño propio).

Abierta la consola de Anaconda, se debe ejecutar uno por uno los comandos de instalación que aparecen en el margen izquierdo de la figura inferior.

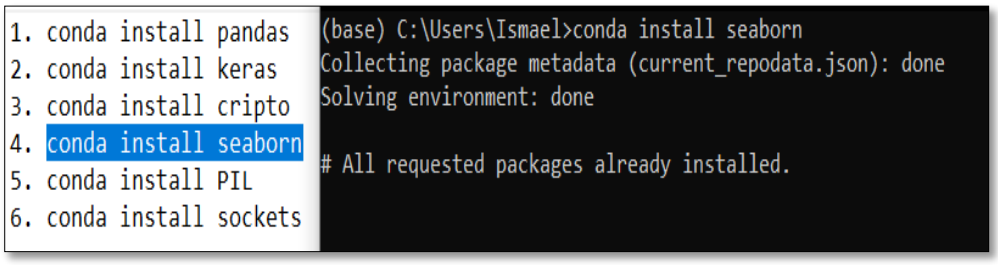


Figura 62. Proceso de instalación de un paquete en Anaconda desde Anaconda Prompt (Diseño propio).

Una vez finalizada la instalación de los paquetes, se debe preparar el directorio de la figura 45 antes de poner en funcionamiento el servidor, ya que de lo contrario no será posible. Utilizando para ello el conjunto de datos descargable e indicado en las referencias. Tras haberlo hecho arrancamos Jupyter Notebook desde Anaconda Prompt.

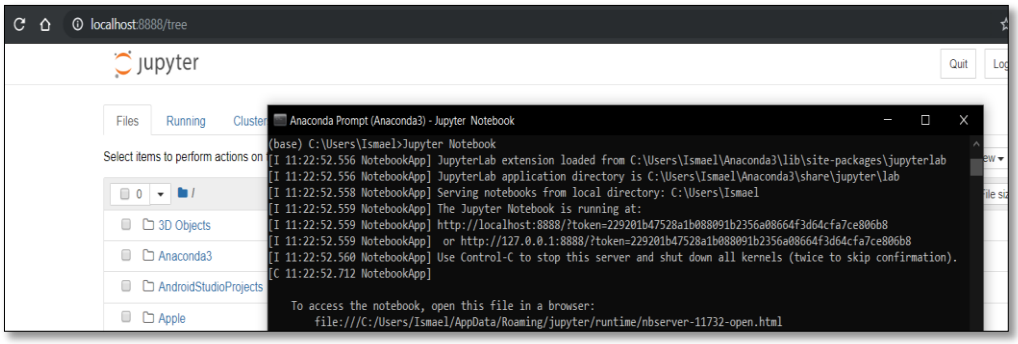


Figura 63. Apertura de Jupyter Notebook desde Anaconda Prompt (Diseño propio).

A continuación navegaremos en Jupyter hasta encontrar el directorio en el que se encuentre el fichero llamado “Servidor Python”. Antes de ejecutar el servidor, es necesario asegurarse de que en el fichero de Python “Servidor Python” y en el fichero de texto “Server_config” se hace uso de la dirección IP del sistema y de un puerto disponible.

Para comprobar la dirección IP del sistema, abrimos un terminal y utilizamos el comando *ipconfig*. Tras ello copiamos la dirección IPv4 que tenemos actualmente según el medio de conexión que estemos utilizando.

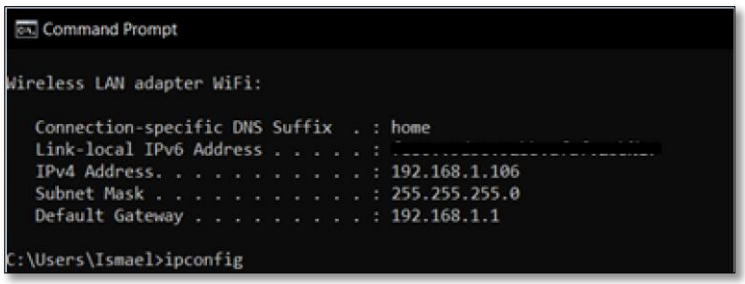


Figura 64. Comprobación de la dirección IP actual desde un terminal (Diseño propio).

Cambiaremos la dirección IP en el fichero Python “Servidor Python” e indicaremos un puerto disponible en el fichero “Server_config”. Por último tan solo será necesario pulsar sobre el botón “*Run*” una vez abierto el fichero del servidor como para ponerlo en marcha.

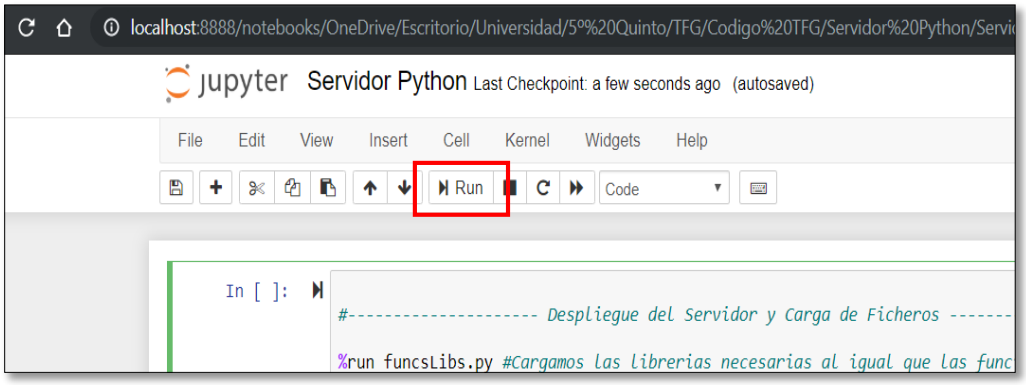


Figura 65. Ejecución del fichero que activa el servidor desde Jupyter Notebook (Diseño propio).

Para realizar la **instalación del cliente en Android**, es necesario transferir al dispositivo el fichero de instalación de la aplicación móvil con extensión “.apk”. En la gran mayoría de dispositivos con este S.O es posible contar con un administrador de ficheros, habrá que utilizarlo para buscar el fichero de instalación.

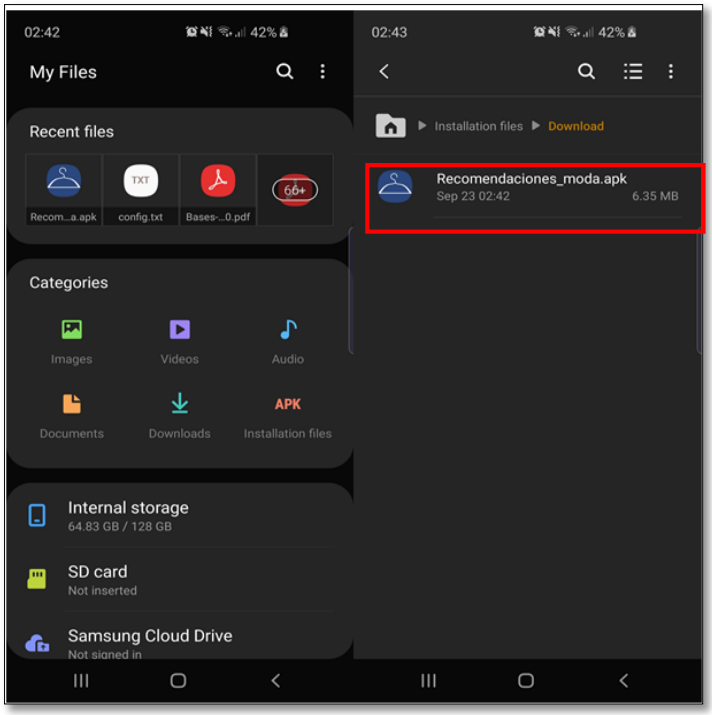


Figura 66. Búsqueda del fichero de instalación de la aplicación con extensión .apk (Diseño propio).

Encontrado el fichero de instalación será procede a abrirlo. Una vez abierto pulsamos sobre el botón inferior derecho “*install*”, este nos permite confirmar si realmente deseamos instalar la aplicación o si pulsamos sobre “*cancel*” para cancelar la instalación. Una vez que aceptemos, la instalación dará comienzo.

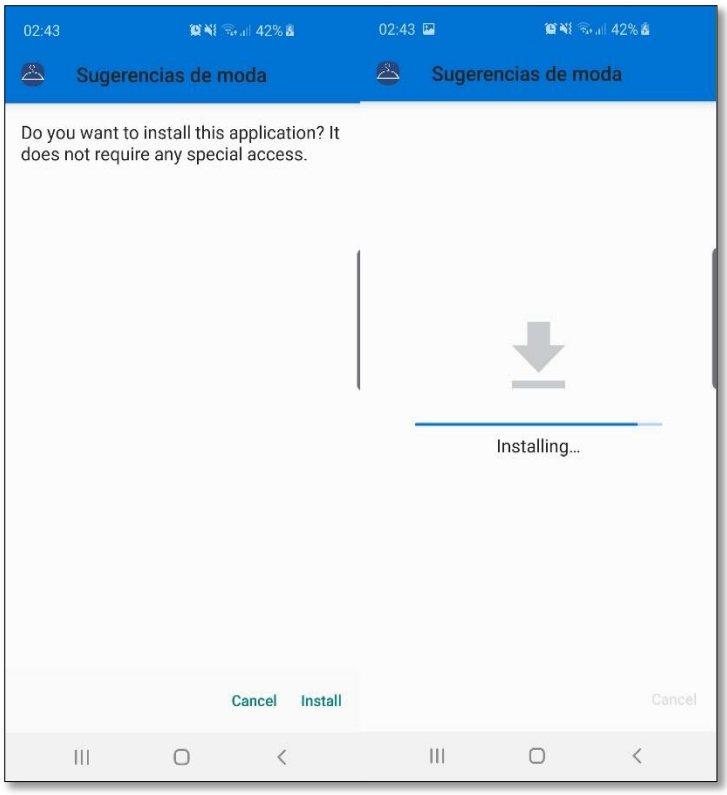


Figura 67. Confirmación y proceso de instalación de la aplicación móvil (Diseño propio).

Finalizada la instalación, en el menú de aplicaciones Android contaremos con un icono de acceso a la aplicación como el de la siguiente figura.

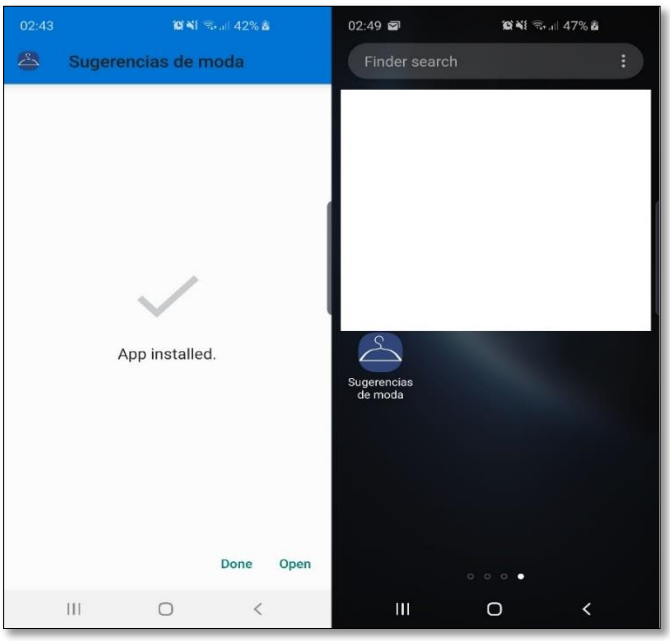


Figura 68. Confirmación de instalación completada e icono de la aplicación móvil (Diseño propio).

Antes de abrirla, configuraremos el fichero “config.txt” que se sitúa en el directorio de instalación de la aplicación “/Android/data/com.example.fashrecom”. Encontrado el fichero de configuración modificamos la primera línea con la misma dirección IPv4 que la utilizada en el

servidor y en la segunda línea, indicamos el mismo puerto utilizado que en el servidor. Esto permitirá que la conexión se pueda realizar correctamente.

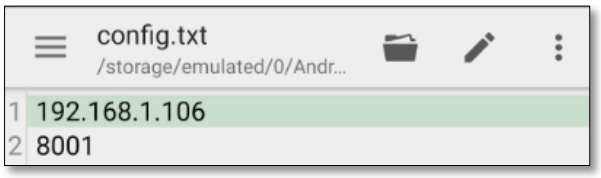


Figura 69. Confirmación y proceso de instalación de la aplicación móvil (Diseño propio).

Salvados los cambios realizados sobre el fichero de configuración tan solo quedará abrir la aplicación desde el panel de aplicaciones. Una vez abierta ya podremos hacer uso de ella.

Figura 70. Interfaz de la aplicación móvil tras su apertura (Diseño propio).