Desarrollo de Técnicas de Machine Learning Operations (MLOPS)

Memoria Trabajo Fin de Grado Ingeniería Informática

Septiembre 2023

Autora:

Ana Gómez Marcos

Tutores:

Angélica González Arrieta

Roberto López González

Certificado de los tutores

Dña. Angélica González Arrieta, profesorado del Departamento de Informática y Automática de la Universidad de Salamanca.

D. Roberto López González, fundador y CEO de la empresa Artelnics.

CERTIFICAN:

Que el trabajo titulado “Desarrollo de Técnicas de Machine Learning Operations (MLOPS)” ha sido realizado por Dña. Ana Gómez Marcos, con DNI 70958281-S, para la asignatura “Trabajo de Fin de Grado” de la titulación “Grado en Ingeniería Informática de la Universidad de Salamanca”.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 5 de Septiembre de 2023

Firmado por ***7144** ROBERTO LOPEZ (R: ****3658*) el día 07/09/2023 con un certificado emitido por AC Representación

GONZALEZ ARRIETA MARIA ANGELICA - 06555917C

Firmado digitalmente por GONZALEZ ARRIETA MARIA ANGELICA - 06555917C
Fecha: 2023.09.07 08:16:32 +02'00'

Firmado por GOMEZ MARCOS ANA - ***5828 el día 07/09/2023 con un certificado emitido por AC FNMT Usuarios**

Resumen

El machine learning (ML) es un subconjunto de la Inteligencia Artificial (IA), se centra en la capacidad de las máquinas para recibir un conjunto de datos y aprender por sí mismos, adaptando los algoritmos usados a medida que aprenden más sobre la información que es procesada. Este aprendizaje puede ser usado en muy diversos ámbitos, en este caso nos centraremos en sus usos en el ámbito de la investigación.

A pesar de su gran potencia aún sigue costando acomodar su uso a situaciones determinadas y por eso la idea del proyecto es precisamente esa, desarrollar y adaptar un modelo de machine learning para poder aplicarlo a una situación específica, de manera que siga los principios de Machine Learning Operations (MLOps) en el sentido de que involucra la implementación y despliegue de un modelo de Machine Learning en un entorno de laboratorio.

El objetivo es entrenar y mejorar el modelo que se va a desarrollar, usando una rama del machine learning (ML) llamada deep learning (DL) o aprendizaje profundo, para que pueda distinguir de forma acertada los elementos requeridos en imágenes realizadas por microscopio, y a su vez beneficiarse de las mejores prácticas de MLOps para garantizar un despliegue rápido y eficiente, todo esto con la finalidad de poder ayudar a reducir los tiempos que se invierten en este sentido en el campo de la investigación, donde los tiempos son un elemento fundamental y así poder realizar los cálculos necesarios sobre las imágenes analizadas, permitiendo a los investigadores dedicar más esfuerzo a otras actividades más pertinentes.

En particular, se enfocará en la detección de células, las cuales desempeñan un papel crucial en el diagnóstico y estudio de diversas patologías, como el cáncer, por ejemplo. Esto se debe a que identificar las células marca el punto de partida para numerosos análisis. Las características de estas células, como su densidad o brillo en una imagen, aportan datos sumamente importantes en una variedad de estudios. Esto resalta la potencia de esta disciplina en el sector, al mismo tiempo que se presentan los procedimientos para generar un modelo de machine learning.

Para demostrar la utilidad y aplicabilidad de estas complejas técnicas de aprendizaje automático al público en general, se ha concebido la idea de crear una interfaz interactiva. Esta interfaz proporcionará a los usuarios una experiencia visualmente más atractiva y comprensible que les permitirá experimentar de primera mano cómo funciona el modelo de aprendizaje automático que hemos desarrollado.

Uno de los objetivos principales es eliminar las barreras entre las técnicas de aprendizaje automático y el público en general, haciendo que conceptos a menudo complejos sean accesibles, emocionantes y sobre todo útiles. Se desea mostrar cómo el machine learning, así como el uso de la práctica de Machine Learning Operations, no son sólo una herramienta poderosa en manos de expertos, sino también una tecnología que puede tener un impacto directo en la mejora de la investigación.

Palabras clave: Machine Learning, Inteligencia Artificial, Red Neuronal, Algoritmo, Redes Convolucionales, U-Net, Machine Learning Operations (MLOps).

Abstract

Machine Learning (ML) is a subset of Artificial Intelligence (AI) that focuses on machines' ability to receive a dataset and learn autonomously, adapting the algorithms used as they gain more knowledge about the processed information. This learning can be applied in various fields, but in this case, we will concentrate on its applications in the realm of research.

Despite its immense potential, accommodating its use for specific situations still poses challenges. That's precisely the project's idea: to develop and tailor a machine learning model to apply it to a specific scenario while adhering to the principles of Machine Learning Operations (MLOps). This involves implementing and deploying a Machine Learning model in a laboratory environment.

The goal is to train and enhance the model being developed, utilizing a branch of machine learning called deep learning (DL). This ensures that it can accurately distinguish the required elements in microscope images and, at the same time, benefit from MLOps best practices to ensure a fast and efficient deployment. The ultimate aim is to reduce the time invested in this regard in the field of research, where time is a crucial factor, enabling researchers to focus more effort on other relevant activities.

Specifically, the focus will be on cell detection, which plays a crucial role in the diagnosis and study of various pathologies, such as cancer, for instance. Identifying cells marks the starting point for numerous analyses. The characteristics of these cells, such as density or brightness in an image, provide highly valuable data in various studies. This underscores the power of this discipline in the sector while presenting the procedures for generating a machine learning model.

To demonstrate the utility and applicability of these complex machine learning techniques to the general public, the idea of creating an interactive interface has been conceived. This interface will provide users with a visually appealing and comprehensible experience, allowing them to firsthand experience how the developed machine learning model operates.

One of the main objectives is to eliminate barriers between machine learning techniques and the general public. This involves making often complex concepts accessible, exciting, and, most importantly, useful. The aim is to show that machine learning, as well as the practice of Machine Learning Operations, is not just a powerful tool in the hands of experts but also a technology that can have a direct impact on improving research.

Keywords: Machine Learning, Artificial Intelligence, Neural Network, Algorithm, Convolutional Networks, U-Net, Machine Learning Operations (MLOps).

Tabla de contenido

Lista de figuras

Lista de tablas

1. Introducción

El 'machine learning' (ML) o aprendizaje automático [1] es una rama de la Inteligencia Artificial [2], que, a través de algoritmos, permite que las máquinas sean capaces de aprender sin estar obligados a programarlas específicamente para ello, de forma autónoma. Esta habilidad se ha vuelto actualmente indispensable en diversas disciplinas, presente en un sinfín de aplicaciones. Específicamente se ha vuelto de gran interés en disciplinas relacionadas con la investigación, donde centraremos el proyecto, ya que cobra especial interés el tiempo empleado en realizar determinadas tareas.

Dentro del Machine Learning encontramos lo que se denomina como Machine Learning Operations, abreviado como MLOps, que es una práctica que combina conceptos de desarrollo del software y operaciones para gestionar y automatizar el ciclo de vida de los modelos de Machine Learning de forma eficiente. Todo este proceso incluye tareas como desarrollo de modelos, entrenamiento, prueba, implementación, monitoreo y mantenimiento, garantizando que los modelos desarrollados sean confiables, escalables y se puedan desplegar de manera consistente en entornos de producción.

Para acercar estas técnicas vanguardistas al público en general, se ha considerado la realización de una interfaz interactiva que hace que el mundo del aprendizaje automático sea accesible y emocionante y a su vez, sea fácilmente integrable en el flujo de trabajo del laboratorio, respaldando la idea de facilitar la integración de modelos en operaciones científicas. Esta interfaz permite visualizar el funcionamiento del modelo, requisito importante en cualquier carrera técnica.

Actualmente, la combinación de algoritmos de aprendizaje automático con los modelos de redes neuronales está ayudando a los sistemas informáticos a mejorar su rendimiento, permitiendo abordar problemas complejos que necesitan del análisis de una cuantiosa cantidad de datos que, de normal, requerirían de mucho más tiempo y recursos.

En la disciplina del machine learning nos encontramos con que su desarrollo no es tan simple, se basa en un tratamiento y conocimiento trascendente de los datos, una transformación de estos para poder ser usados por los modelos, la elección del modelo que se ajuste más al problema definido y el entendimiento de las métricas de calidad. Sin todo este discernimiento, podría generarse un modelo que errase y llevase a deducir resultados desacertados o falsos con, en muchos casos, consecuencias nefastas en ámbitos como en el que vamos a aplicarla.

El objetivo del proyecto es demostrar la gran capacidad de análisis que nos pueden proporcionar las técnicas de Machine Learning en el ámbito de la investigación, en específico el uso de deep learning en el análisis de imágenes, automatizando un proceso que muchas veces se realiza manualmente debido a la falta de recursos, ya que no existen demasiadas aplicaciones que proporcionen la solución que buscamos. Mediante la implantación de este sistema se busca reducir los tiempos, permitiendo a los investigadores invertirlo en otras actividades de mayor relevancia, de esta manera se alinea con los principios y objetivos de MLOps, especialmente en términos de implementación, monitoreo y gestión del ciclo de vida del modelo en el contexto de la ciencia y el laboratorio.

El aprendizaje profundo o deep learning (DL) [3] como hemos mencionado anteriormente se puede considerar la nueva evolución del machine learning, utiliza el mismo concepto que el machine learning pero usa otros algoritmos distintos, es decir, a través de redes neuronales interconectadas, esta técnica imita la estructura y el funcionamiento del cerebro humano. De esta manera nos permite disminuir la intervención humana a la hora de lograr los resultados esperados.

Se denominan redes neuronales [4] debido a que su funcionamiento está inspirado en la forma en la que funciona el cerebro humano, utiliza "neuronas" interconectadas en una estructura de capas, creando un sistema que es posible de adaptar permitiendo así a las computadoras aprender de sus errores y mejorar, modelando las relaciones entre los datos de entrada y salida que no son lineales y que son complejos.

La interfaz interactiva generada brindará la oportunidad a cualquier usuario de hacer uso del modelo que se va a desarrollar. No solo se verán las técnicas en acción, sino que también acercará su relevancia al público en general y aplicabilidad en el mundo real. Al acercar el aprendizaje automático a través de esta experiencia visual, estamos allanando el camino para una comprensión más profunda y una adopción más amplia de estas innovadoras técnicas en el público en general, todo esto mientras creamos soluciones tangibles para abordar desafíos reales en campos críticos como la investigación del cáncer.

Este documento detalla el proceso de desarrollo del proyecto. La elaboración seguirá las pautas de la plataforma Diaweb [5] y abordará los siguientes aspectos:

- Objetivos del Proyecto: Se expondrán los objetivos que se buscan lograr con la ejecución de este proyecto.
- Técnicas y Herramientas: Se presentarán las metodologías y las herramientas empleadas para llevar a cabo el proyecto.
- Aspectos Relevantes: Se destacarán los puntos más notables que surgieron durante el desarrollo del sistema.
- Conclusiones y Futuras Líneas de Trabajo: Se proporcionarán las conclusiones del proyecto y se sugerirá cómo se puede continuar desarrollando el sistema en el futuro.

La información contenida en este informe se complementa con:

- Anexo I - Plan del proyecto software: recoge la planificación del proyecto para valorar su viabilidad.
- Anexo II - Especificación de requisitos software: proporciona una descripción completa de todos los requisitos de software definidos para el sistema.
- Anexo III - Proceso de diseño centrado en el usuario: Aborda las necesidades, limitaciones, comportamientos y características de los usuarios en el proceso de diseño.
- Anexo IV - Manual del programador: Presenta los aspectos relacionados con el código fuente de la aplicación.
- Anexo V - Manual de usuario: ofrece un tutorial esencial para el manejo adecuado de la aplicación.

2. Machine Learning

Históricamente la Inteligencia Artificial no es del todo lineal, se han producido periodos de gran abundancia en su desarrollo pero también ha sufrido los llamados "inviernos de la IA".

Para poder entender un poco la evolución histórica de la Inteligencia Artificial [6] y de uno de sus subcampos, el Machine Learning (ML) hay que mencionar los momentos más importantes que dieron paso a cómo lo conocemos hoy.

Entre 1600 y 1850 debemos mencionar a tres importantes figuras de la prehistoria de la Inteligencia Artificial:

- Wilhelm Schickard (1623), destacado profesor, matemático, teólogo y cartógrafo alemán, inventó varias máquinas pero su aportación principal fue la primera calculadora mecánica (artefacto que le permitía realizar operaciones aritméticas de forma completamente mecánica) y una máquina para aprender la gramática del hebreo.
- Charles Babbage (1822) construyó la calculadora mecánica, capaz de realizar cálculos en tablas de funciones numéricas por el método de diferencias, además de diseñar la máquina analítica para ejecutar programas de tabulación o computación.
- Ada Lovelace (1830), matemática británica y primera programadora cuya aportación más importante fue el concepto de la máquina universal. Creó un artefacto que en teoría se podía programar y reprogramar para realizar diversas tareas sin limitarse al cálculo matemático como el procesamiento de símbolos, palabras e incluso música.

Entre 1900 y 1950 se produjeron tres eventos que marcaron el inicio de la era computacional.

- En 1911 se funda IBM, una compañía que desarrollaba máquinas para contar tarjetas perforadas, llevándolos a ser líderes en soluciones de software, hardware y servicios que han marcado el avance tecnológico de esta época.
- En 1936 se crea la máquina de Turing, por Alan Turing, padre de la Inteligencia Artificial. Creó un modelo computacional capaz de almacenar y procesar información virtualmente, marcando la historia de la informática y siendo considerado este punto como el origen de las computadoras, teléfonos, tabletas y otras tecnologías.
- 1943 se desarrolla la primera computadora digital funcional ENIAC (Electronic Numerical Integrator And Computer)

Por último tenemos ya lo que podemos considerar como el comienzo de la Inteligencia Artificial, que se da en el periodo entre 1950 y los 2000.

- En 1950 se realiza la prueba de Turing, desarrollada por Alan Turing, con el objetivo de determinar si la Inteligencia Artificial puede imitar las respuestas humanas. Consiste en realizar una conversación entre una computadora y una persona pero sin determinar quién es la máquina, la computadora pasa con éxito la prueba de Turing si no se puede distinguir al humano de la máquina.
- En 1956 se realiza la primera conferencia en Darmouth College sobre Inteligencia Artificial, punto de partida para llamar a la Inteligencia Artificial con ese término y

dónde se determinó que en 25 años los ordenadores harían todo el trabajo que el ser humano realizaba por ese entonces.

- Entre 1974 y 1980 se produjo el primer invierno de la Inteligencia Artificial, este dió lugar debido a la poca memoria y la rapidez en los procesos, muy inferior a los recursos de los que disponemos hoy en día.
- Entre 1970 y 1980 se dió paso a los sistemas expertos, los cuales tuvieron gran popularidad. Usaban conocimientos de expertos para crear un programa en el que el usuario realiza una pregunta con el fin de obtener una respuesta y se cataloga como útil o no dependiendo del resultado. A pesar de su sencillez se volvieron muy útiles aunque en la actualidad ha disminuido su uso.
- En 1980 tuvieron éxito los procesadores de lenguaje natural, haciendo posible el entendimiento del lenguaje humano a las computadoras y máquinas. Se desarrollaron para traducir el ruso al inglés pero en 1980 se aplicaron diferentes algoritmos y tecnologías computacionales que proporcionaron un sinfín de usos alternativos.
- Entre 1987 y 1193 se produjo el segundo invierno de IA. Se desarrolló el primer sistema comercial de Inteligencia Artificial XCON, se creó el lenguaje de programación LISP y se convirtió en el denominador común entre desarrolladores de Inteligencia Artificial pero en 1987 el mercado colapsó debido a que la tecnología de las PCs opacaba a las costosas máquinas LISP.
- En 1990 se creó e investigó los agentes inteligentes, también llamados bots o asistentes virtuales digitales, capaces de interpretar y procesar la información que recibían del entorno y actuar con base a los datos que recogían y analizaban.

Posteriormente se dió lugar a la evolución de la Inteligencia Artificial tal y como la conocemos hoy, con asistentes virtuales, androides, técnicas de Machine Learning aplicadas a procesadores de lenguaje natural, algoritmos de Inteligencia Artificial Autónoma, etc.

En la historia del Machine Learning se producen como tal una serie de momentos que marcaron una diferencia en este ámbito, entre los cuales tenemos:

- En 1952 Arthur Samuel desarrolla el primer programa que juega a las damas chinas, dando una demostración temprana de los conceptos fundamentales de la Inteligencia Artificial.
- En 1957 Frank Rosenblatt diseña el Perceptrón que permite clasificar, explicar y modelar habilidades de reconocimiento de patrones en imágenes (primer ordenador que se especializaba en crear redes neuronales).
- En 1963 se construye MENACE (Máquina Motor Educable de Ceros y Cruces), una computadora hecha de 304 cajas de fósforos diseñadas donde se construyó uno de los primeros programas con la capacidad de aprender a jugar a diversos juegos.
- En 1967 se desarrolla el algoritmo del vecino más cercano, uno de los algoritmos de clasificación más básicos y esenciales del Machine Learning. Clasificador de aprendizaje supervisado que usaba como medio la proximidad para reconocer patrones, minería de datos y detección de intrusos a un punto de datos individual para clasificar el interés de los datos que lo rodean.
- En 1970 Seppo Linnainmaa publica el modelo inverso de diferenciación automática, conocido posteriormente como propagación hacia atrás y usado para entrenar redes neuronales artificiales.
- En 1979 Hans Motavec crea el primer vehículo autónomo, que constaba de dos ruedas y una cámara de televisión con movilidad lateral.
- En 1986 Terry Sejnowski crea NETtalk, una red neuronal artificial que aprende a pronunciar palabras como los niños.
- En 1997 Jürgen Schmidhuber y Sepp Hochreiter crean la primera inteligencia de reconocimiento del discurso mediante una técnica llamada LSTM, que tenía la capacidad de recopilar datos como imágenes, palabras y sonidos donde un algoritmo se encarga de interpretar y almacenar la información obtenida para realizar acciones sobre los datos.
- En 2014 Facebook desarrolla DeepFace, un algoritmo que permite reconocer individuos en fotos, permitiendo, con una exactitud del 97,25% identificar a las personas que aparecían en cada imagen.

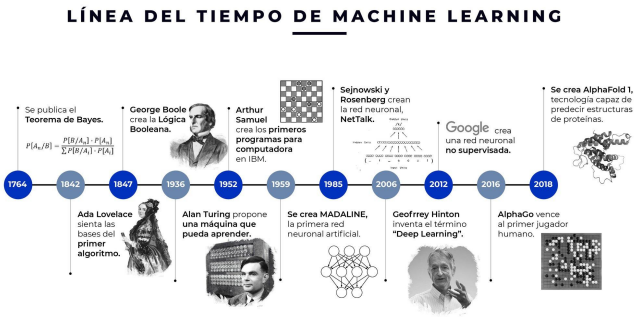

Figura 1: Línea del tiempo de Machine Learning

Nota: Fuente [6]

Grupos de algoritmos de Machine Learning

En este apartado voy a centrarme en los dos grupos de algoritmos principales que se usan en Machine Learning (ML) [7]:

- Aprendizaje Supervisado: son algoritmos que cuentan con un aprendizaje previo, es decir, aprenden de datos con elementos etiquetados que posibilitan tomar decisiones o hacer predicciones, necesitando de intervención humana.
    - Mientras el 'operador' conoce las respuestas al problema, el algoritmo identifica los patrones en los datos, aprende de las observaciones y de esta manera va siendo corregido por el operador hasta que alcanza el nivel suficiente de precisión y rendimiento.
    - En este caso los algoritmos pueden ser de clasificación (clasificación de objetos dentro de clases) o de regresión (predicción de un valor numérico).
- Aprendizaje No Supervisado: a diferencia de los anteriores, estos algoritmos no cuentan con un conocimiento previo sino que tienen el objetivo de encontrar patrones en el conjunto de datos y así permitir ordenarlos de alguna manera.
    - Aquí se deja que el algoritmo sea el que interprete grandes conjuntos de datos los dirija. A medida que evalúa más datos, aumenta su capacidad de toma de decisiones sobre los mismos y se refina.

Deep Learning

También llamado aprendizaje profundo [8], está actualmente en auge debido a su capacidad de aproximarse cada vez más a la potencia de la percepción humana. En el enfoque del Deep Learning se utilizan estructuras lógicas cuya distribución se asemeja al sistema nervioso de los mamíferos, esto es debido a que usa capas de unidades de proceso, es decir, neuronas artificiales, que son las encargadas de realizar el proceso de detección de características en los objetos que perciben. Proporciona una mejora a considerar si la comparamos con algoritmos más tradicionales, permitiendo clasificar, reconocer, detectar y describir, es decir, entender.

Estos métodos de Deep Learning (DL) y su capacidad de mejora y adaptación constante a cambios en el patrón de información presenta una gran oportunidad para poder introducir un comportamiento más dinámico a la analítica.

A pesar de todas las ventajas mencionadas anteriormente, es verdad que se necesita mucho poder de cómputo para poder resolver problemas de Deep Learning (DL), esto es debido a que su complejidad aumenta conforme aumenta el número de capas y que son necesarios grandes volúmenes de datos para poder entrenar las redes.

3. Objetivos del proyecto

En este apartado se detallarán los objetivos a perseguir con la realización del proyecto, tanto personales como del sistema.

Objetivos del sistema

Como objetivo principal tenemos el desarrollo de un modelo de Machine Learning capaz de identificar objetos en imágenes de laboratorio, en específico de los núcleos de las células, y sacar datos de ellos para su posterior utilización en el ámbito de la investigación, funcionalidad que se tiene pensado implementar en un futuro. Todo esto mientras se aplican las técnicas de Machine Learning Operations (MLOps) para garantizar acelerar los procesos científicos en este contexto de manera efectiva.

Dicho objetivo se conseguirá mediante la realización de los siguientes subobjetivos:

- Implementar PyTorch para construir una U-Net, modificar el modelo y construir los bloques necesarios para su correcto funcionamiento con la finalidad de poder completarlo realizando las modificaciones necesarias, todo esto mediante herramientas que faciliten su desarrollo.
- Entrenar el modelo con las imágenes necesarias previamente tratadas y preparadas para que sea capaz de detectar los núcleos de las células como he mencionado anteriormente.
- Desarrollar el modelo en un entorno en el que sea sencilla su modificación para aquellos con ciertos conocimientos en informática, de manera que garantice la posibilidad de volver a entrenar y actualizar el modelo con datos adicionales y poder mejorar su rendimiento con el tiempo, permitiendo la capacidad de adaptarlo a cambios en la distribución de datos y en las necesidades de rendimiento de la aplicación.

Durante la ejecución del proyecto, surgió otro objetivo importante: crear una interfaz que pueda aprovechar el modelo desarrollado. Esta adición se basa en la idea de proporcionar una herramienta visual que no solo exhiba el funcionamiento interno, sino que también simplifique su accesibilidad para un público amplio además de permitir su fácil integración en el flujo de trabajo del laboratorio, respaldando la idea de facilitar la integración de modelos en operaciones científicas.

Dicho objetivo se conseguirá mediante la realización de los siguiente subobjetivos:

- Diseñar una interfaz intuitiva, con una experiencia de usuario amigable, permitiendo a los usuarios interactuar de manera intuitiva con el modelo y comprender sus resultados sin dificultad, permitiendo que la interfaz promueva la conciencia, es decir, que no solo resuelva el problema sino que también cree conciencia sobre la importancia de las técnicas de machine learning en la investigación.
- Incorporar el modelo realizado con anterioridad en el sistema, es decir, para poder ser usado por la interfaz, de forma que sea funcional.

Objetivos personales

Como objetivo personal se busca superar con éxito la tarea de realizar un proyecto de grandes envergaduras, mediante el cual se puedan plasmar ideas tan complicadas y actuales como es la Inteligencia Artificial de una manera sencilla y asequible, para que cualquier usuario interesado pueda entender la capacidad real que aporta en el mundo de la computación. Esto supone un gran reto personal que, ligado al afán de superación, permite obtener una mayor seguridad en uno mismo.

También se busca demostrar las infinitas posibilidades y todos los usos que tiene la Inteligencia Artificial en nuestro día a día, permitiendo acercar más a los usuarios a temas que quizás no se encuentran al alcance de cualquiera, convirtiéndolos en materias desconocidas y herméticas. Esto mientras contribuye a facilitar tareas de investigación.

Además de permitir la adaptación a un proyecto completamente distinto a todos los realizados personalmente hasta ahora, todo mientras se aplican diversos conocimientos obtenidos durante toda la carrera.

4. Conceptos teóricos

- Inteligencia Artificial (IA):

La Inteligencia Artificial es la habilidad que tiene una máquina para presentar las mismas capacidades que un ser humano, como el razonamiento, el aprendizaje, la creatividad y la capacidad de planificar. Permite que los sistemas tecnológicos puedan percibir su entorno, se relacionen con él, resuelvan problemas y actúen con un fin determinado. Todo esto ya que la máquina es capaz de recibir datos, procesarlos y responder ante ellos adaptando su comportamiento en cierta medida.

- Machine Learning (ML):

El Machine Learning es una rama de la inteligencia artificial que hace posible el aprendizaje autónomo de las máquinas, sin necesidad de ser programadas expresamente para ello.

- Deep Learning:

El Deep Learning o aprendizaje profundo, parte del Machine Learning para, a partir de una gran cantidad de datos y numerosas capas de procesamiento con algoritmos, es capaz de conseguir que un ordenador termine aprendiendo por cuenta propia y realizando tareas semejantes a las que realizan los seres humanos, así como la identificación de imágenes, el reconocimiento del habla, etc.

- Neurona Artificial:

Las Neuronas Artificiales son unidades de cálculo que pretenden simular el comportamiento de una neurona perteneciente a un cerebro humano. La interconexión de varias de estas unidades forma lo que se conoce como redes neuronales artificiales.

Su funcionamiento se basa en un conjunto de conexiones de entrada que permiten recibir los estímulos externos (Valores Xn) con los que se realiza un cálculo interno que, en términos generales se trata de una suma ponderada de los valores de las entradas con su peso correspondiente (Wn), posibilitando que cada conexión que llega a la neurona tenga un valor que permita definir con qué intensidad afecta a la neurona cada variable de entrada y posteriormente pasando por una función de activación generando de esta manera un valor de salida (Y). A parte de las entradas también encontramos un valor añadido llamado sesgo o bias en inglés. Como podemos observar en la Figura 2.

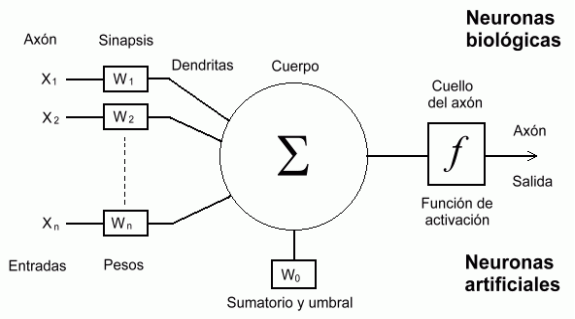

Figura 2: Estructura de la neurona artificial

Nota: Requena, A., Quintanilla, R., Bolarín, J.M., Vázquez, A., Bastida A., Zúñiga, J., et al. Nuevas Tecnologías y Contaminación de Atmósferas, para PYMEs (VI-3-4, Pág. 3). Universidad de Murcia, Spain. Recuperado de https://www.um.es/LEQ/Atmosferas/Ch-VI-3/F63s4p3.htm

- Redes Neuronales Artificiales:

Se trata de un modelo computacional que permite simular el comportamiento del cerebro humano permitiendo dotar a las máquinas de la capacidad de aprender de una manera similar a como se comporta nuestro cerebro.

- Capas:

Niveles que forman la estructura de una red neuronal. Dentro de la red existen varios tipos; las capas de entrada que se encargan de recibir los datos, las de salida que proporcionarán un resultado al usuario y las capas ocultas que no poseen una conexión directa con el entorno conteniendo unidades no observables:

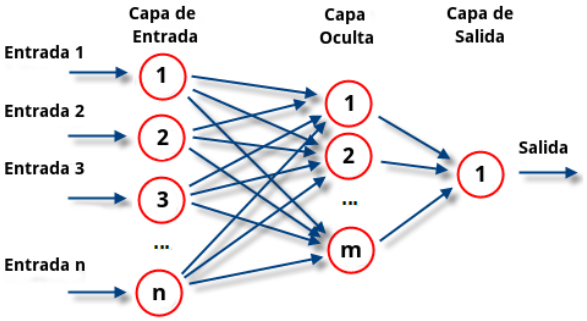

Figura 3: Capas de una red neuronal

- CNN (Convolutional Neural Network):

Dentro del Deep Learning, una red neuronal convolucional (CNN) es un tipo de red neuronal artificial que se usa aplicada al análisis de imágenes. Usa un tipo de operación matemática llamada convolución en lugar de la multiplicación de matrices general, en al menos una de sus capas.

- U-Net:

Es un modelo de red neuronal dedicado a tareas de visión artificial (Computer vision), más concretamente a problemas de segmentación semántica. Su nombre se debe a que su estructura se presenta con la forma de una U como podemos ver en la Figura 4.

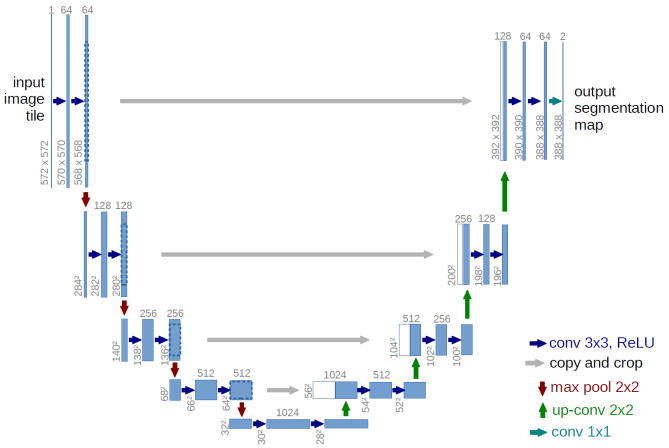

Figura 4: Estructura de la U-Net

- Segmentación Semántica:

Se trata de un algoritmo de Deep Learning que asocia una etiqueta o categoría a cada píxel presente en una imagen. Se usa con el fin de reconocer un conjunto de píxeles que conforman distintas categorías.

- Segmentación de instancias:

Va un paso más allá de la Segmentación Semántica, clasifica cada instancia de una misma clase por separado.

- TensorFlow:

Librería de código libre para Machine Learning (ML). Desarrollada por Google para satisfacer las necesidades a partir de redes neuronales artificiales. Permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos.

- *Front-end* y *Back-end*:
    - Frontend es la parte de una aplicación que se encarga de interactuar con el usuario al que va dirigido. Todo aquello visual previamente configurado por el programador, de manera que obtiene una dinámica que interactúa con el cliente.
    - Backend es la parte del servidor, a la cual no puede acceder el usuario y está destinada a realizar diversos cálculos, operaciones y acciones con el fin de comunicar al frontend que a su vez comunica con el usuario.

- Época *(epochs):*

Iteración completa a través del conjunto de datos de entrenamiento. En cada época el modelo pasa por cada lote de datos proporcionado en el conjunto de entrenamiento, calcula las predicciones y pérdidas, actualizando a su vez los parámetros del modelo usando un algoritmo de optimización elegido.

Después de cada época, el modelo habrá visto y ajustado su rendimiento en todo el conjunto de datos de entrenamiento. La idea principal es que a medida que avanzan las épocas, el modelo debe mejorar su capacidad de generalizar y realizar las predicciones más precisas en datos que no ha procesado ni visto con anterioridad.

5. Técnicas y herramientas

A continuación comentaré las herramientas utilizadas para desarrollar el modelo, las herramientas metodológicas y librerías necesarias para realizar el proyecto.

Entorno de desarrollo y herramientas

Visual Studio Code

Visual Studio es un conjunto de herramientas desarrollado por Microsoft para crear software. Proporciona un espacio donde los programadores pueden escribir, editar y depurar su código de manera efectiva. Además, ofrece características de resaltado de sintaxis, depuración y soporte para varios lenguajes y plataformas, lo que lo convierte en una herramienta esencial en el proceso de desarrollo de aplicaciones y software.

Se ha utilizado para desarrollar todo lo relacionado con la interfaz del proyecto.

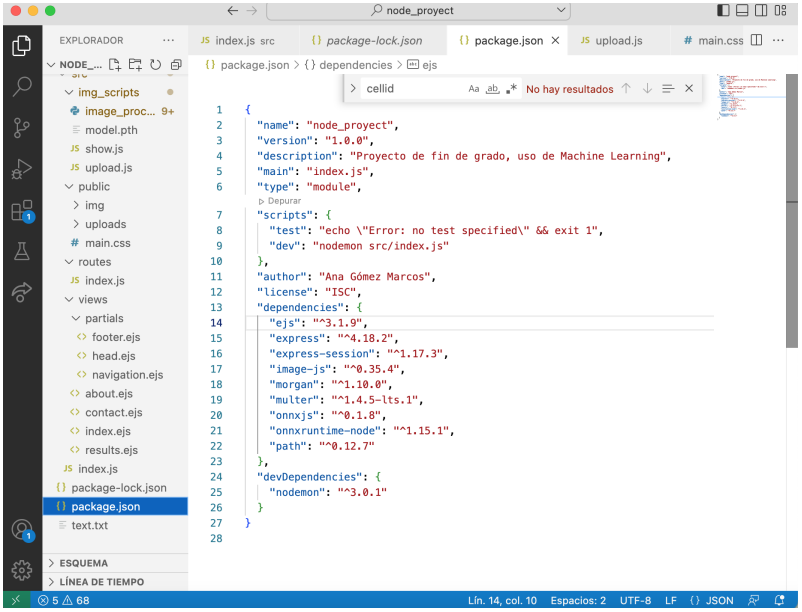

Figura 5: Visual Studio Code

Diagrams.net - Draw.io

Diagrams.net (antes Draw.io) es una herramienta en línea para crear diagramas visuales fácilmente. Permite arrastrar y soltar elementos para construir diagramas como diagramas de flujo y organigramas. Es útil para comunicar ideas de manera visual y colaborar en tiempo real. Ideal para representar conceptos y procesos de forma clara y profesional.

Se ha utilizado para realizar todos los diagramas de casos de uso y actores. Esta elección es debida a que se ha utilizado con anterioridad.

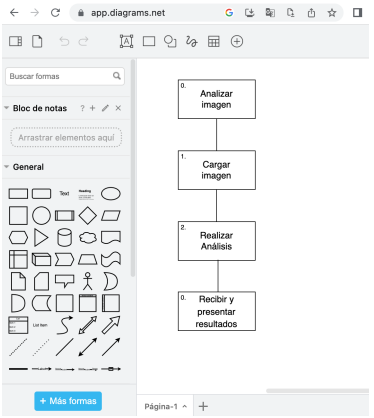

Figura 6: Draw.io

Google Colaboratory

Producto de Google Research. Permite a los usuarios escribir y ejecutar código arbitrario de Python en el navegador. Técnicamente, es un servicio de cuaderno alojado en Jupyter que no requiere configuración y que ofrece acceso a recursos como GPUs necesarias para tareas de aprendizaje automático y análisis de datos.

Se va a usar para tareas que requieran de mayores recursos que un ordenador no pueda aportar ya que es una interfaz en la nube, permitiendo que la memoria y el espacio no ocupen en nuestro ordenador sino que los asigna Google. Además de ser usado para poder monitorear y mantener el modelo, reentrenarlo y actualizarlo de manera regular permitiendo incorporar nuevos datos y recalibración de hiperparámetros.

Figura 7: Google Colaboratory

Bootstrap

Framework de diseño front-end de código abierto, el cual nos proporciona las herramientas suficientes para generar interfaces web de manera rápida y sencilla.

Proporciona una colección de componentes y estilos predefinidos, como botones, formularios, barras de navegación y más que han sido utilizados en la creación de la aplicación web de este proyecto.

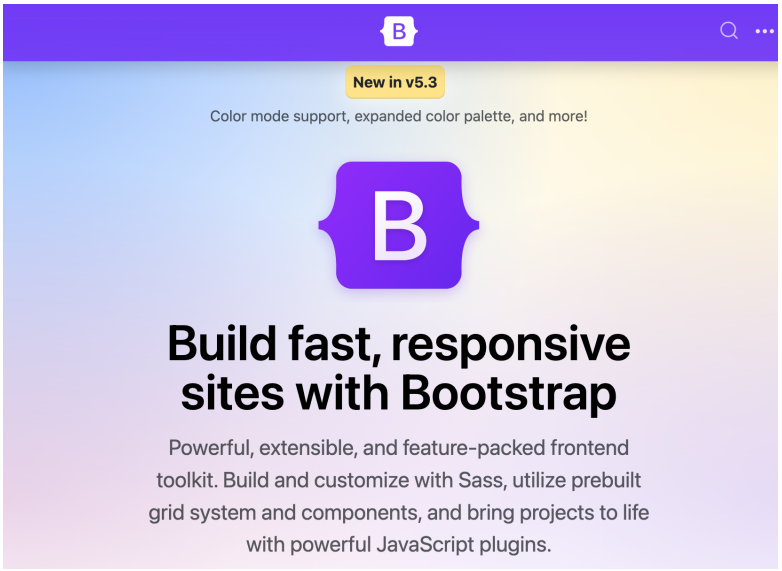

Figura 8: Bootstrap

Código de programación

Python

Lenguaje de programación de alto nivel y con código abierto. Tiene una filosofía que hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, es decir, soporta de forma parcial la orientación a objetos, programación imperativa y en menor medida, la programación funcional.

Durante el desarrollo del proyecto se ha usado para la creación del modelo de Inteligencia artificial y posteriormente, para generar el script que usa el modelo dentro de la interfaz.

HTML y CSS

Lenguajes de programación para la creación del *front-end* de la aplicación.

Mediante HTML se estructura la página web con sus contenidos, los cuales se modifican visualmente mediante CSS.

Se usará en el proyecto para generar una interfaz sencilla que facilite el uso del modelo a usuarios que no estén familiarizados con los cuadernos de Google.

JavaScript

Lenguaje de programación que funciona en los navegadores de forma nativa (no es necesario compilar). Se ha usado como complemento a HTML y CSS para crear la estructura de la aplicación web.

Herramientas CASE

Asana

Software que permite administrar proyectos ya sea generando planes, asignando recursos a diversas tareas, realizar un seguimiento del desarrollo del proyecto, coordinar presupuestos y distribuir las cargas de trabajo, todo esto mediante tableros.

Bibliotecas Python

A continuación, se presentan las bibliotecas más relevantes para la implementación de un modelo de inteligencia artificial:

TensorFlow

Librería de código libre para Machine Learning (ML). Desarrollada por Google para generar redes neuronales artificiales, permitiendo también entrenarlas de forma que sean capaces de detectar patrones y razonamientos usados por los humanos.

Es multiplataforma, pudiendo trabajar con GPUs, CPUs e incluso con las unidades de procesamiento de tensores (TPUs).

Keras

Biblioteca de código abierto (con licencia MIT) para Python, basada principalmente en el trabajo de François Chollet, desarrollador de Google. El objetivo principal de esta biblioteca es acelerar la creación de redes neuronales, para ello, funciona como una interfaz de uso intuitivo (API) que permite acceder a varios frameworks de aprendizaje automático y desarrollarlos.

Scikit-Learn

Librería gratuita para Python. Cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad. Presenta compatibilidad con otras librerías (NumPy, SciPy y matplotlib por ejemplo).

Herramienta básica para empezar a programar y estructurar los sistemas de análisis de datos y modelado estadístico.

Una de las ventajas que presenta es la variedad de módulos y algoritmos que facilitan el aprendizaje y trabajo.

PyTorch

Marco de trabajo de código abierto usado principalmente en aprendizaje profundo y procesamiento de tensores. Desarrollado principalmente por el equipo de investigación de inteligencia artificial de Facebook y actualmente es una de las bibliotecas más ampliamente usadas para poder construir y entrenar modelos de aprendizaje profundo.

A diferencia de otros marcos de trabajo de aprendizaje profundo, permite usar tensores dinámicos, pudiendo cambiar de tamaño y forma durante la ejecución del programa, facilitando el desarrollo de modelos flexibles y adaptables, permite el cálculo automático de gradientes (autograd), una amplia variedad de módulos y clases para construir y entrenar redes neuronales, optimizadores predefinidos para ajustar los parámetros de la red durante el entrenamiento, módulos de redes neuronales predefinidos, flexibilidad sobre el flujo del programa, permite la ejecución en GPU y la distribución de cómputos en múltiples dispositivos y además contiene una comunidad activa con una gama enorme de recursos y documentación.

Datasets

Un dataset se define como una colección de datos o imágenes cuya finalidad es el entrenamiento de una red neuronal, convirtiéndose en un elemento fundamental para el correcto funcionamiento de la Inteligencia Artificial.

Para poder entrenar cualquier modelo de aprendizaje automático necesitamos los siguientes tipos de conjuntos de datos que se usan en diferentes etapas de creación del modelo: entrenamiento, validación y conjuntos de prueba.

- Train (entrenamiento)

  Sirve para ajustar inicialmente el modelo, es un conjunto de ejemplos usados para poder ajustar los parámetros que usará el modelo. Normalmente es el conjunto más grande de los tres.

- Validation (validación)

  El modelo ya ajustado se usa para predecir las respuestas de las observaciones en un segundo conjunto de datos, el conjunto de datos de validación, el cual proporciona una evaluación imparcial del ajuste del modelo en el conjunto de datos de entrenamiento mientras se ajustan los hiperparámetros del modelo.

- Test (prueba)

  Conjunto de datos usados para proporcionar una evaluación de forma imparcial del ajuste del modelo final en el conjunto de datos de entrenamiento. No se deben haber usado con anterioridad.

En el caso de este proyecto se ha decidido usar un dataset pequeño, de 90 imágenes, de las cuales hemos usado 60 para entrenamiento y validación y 30 para pruebas.

El dataset utilizado en este trabajo fue obtenido de la página web de EMBL-EBI BioStudies con el número de acceso S-BSST265. Este dataset fue creado el 15 de abril de 2020 y modificado por última vez el 8 de marzo de 2023. Los autores del dataset son Sabine Taschner-Mandl, Inge M. Ambros, Peter F. Ambros, Klaus Beiske, Allan Hanbury, Wolfgang Doerr, Tamara Weiss, Maria Berneder, Magdalena Ambros y Eva Bozsaky. Puede acceder al dataset en la siguiente URL: https://www.ebi.ac.uk/biostudies/bioimages/studies/S-BSST265/

Generalmente para el entrenamiento de una red neuronal se utilizan unos porcentajes determinados para cada conjunto de datos: un 80% para entrenamiento, 10% para el conjunto de datos de validación y otro 10% para el conjunto de datos de prueba. Estos porcentajes no son fijos, se pueden modificar conforme se quiera dar un mayor hincapié en el entrenamiento (ajuste de parámetros), en la validación (ajuste de hiperparámetros) o de test, respetando siempre que el conjunto de elementos de entrenamiento sea el mayor de todos como podemos observar en la Figura 9.

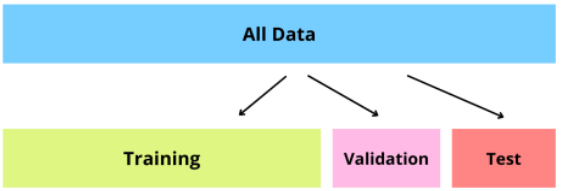

Figura 9: Disposición conjunto de datos

6. Aspectos relevantes del desarrollo del proyecto

En este apartado se presentan los aspectos más importantes de las distintas fases que conllevan el desarrollo del proyecto.

Metodología

Como cualquier proyecto de investigación se necesita de una metodología flexible que permita enfrentarse a las dificultades que puedan presentarse a lo largo del desarrollo del proyecto. En este caso se ha elegido la metodología Kanban [9], que permite flexibilidad, simplicidad y visualización de tareas y procesos a la hora de realizar las tareas de investigación.

Metodología Kanban

Metodología originaria de Japón, se desarrolló en las plantas de producción de Toyota para optimizar la productividad. Su nombre se traduce a "tarjeta de señal" ya que se sustenta en la utilización de tarjetas visuales para realizar la producción.

Se estructura mediante tarjetas, las cuales son tareas del proyecto, permitiendo así la elección de tareas a cada uno de los integrantes del equipo de desarrollo, actualizando de manera transparente los avances al resto de miembros.

Proporciona un gran valor a la evolución contínua y optimización de las labores de trabajo y se caracteriza por:

- Visualización del flujo de trabajo: permite ver las tareas en una tabla lo que supone que todos los integrantes del equipo estén más atento al trabajo que se está realizando y el avance del mismo.
- Limitación del trabajo en curso: es decir, permite equilibrar el trabajo, impidiendo que se produzcan demasiadas tareas en paralelo para evitar desarreglos.
- Políticas explícitas: definición de directrices y normas para la gestión del trabajo y la determinación de los tiempos y las formas en las que se actualiza una tarea.
- Gestión del flujo de trabajo: permite supervisar, medir y analizar el flujo de trabajo en todas las etapas permitiendo así la gestión de cambios y variaciones que puedan surgir.

Como en el proyecto existen multitud de tareas diferentes, la característica flexible y dinámica de este tipo de metodología nos permite agrupar todos los aspectos necesarios para que no suponga un impedimento a la hora de realizar el proyecto.

Para realizar el proyecto vamos a usar la herramienta llamada Asana que nos posibilita organizar de forma sencilla el proyecto mediante distintos tipos de vistas permitiéndonos observar las tareas, fechas o gráficas.

Tablero

Como se ha mencionado anteriormente, podemos usar distintas vistas para la visualización de las tareas, siendo la más común mediante tableros.

En Asana la las tareas se dividen en las siguientes categorías (Figura 10):

- Tareas pendientes: grupo de tareas que aún no han sido asignadas/aceptadas por los integrantes o que no han sido inicializadas.
- En curso: grupo de tareas en las que actualmente se está trabajando.
- Trabajo terminado: grupo de tareas ya finalizadas por los integrantes del grupo.

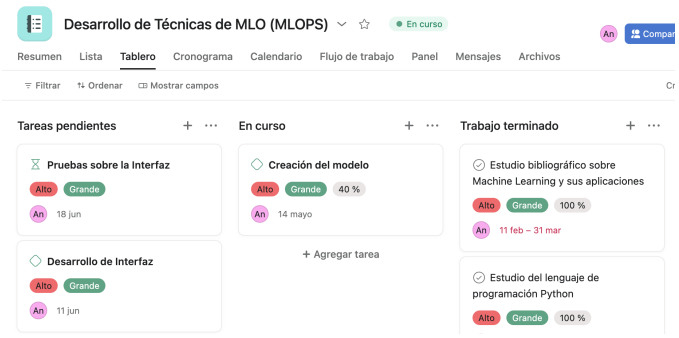

Figura 10: Disposición tareas en Asana

Además nos permite determinar las características de cada una de las tareas, siendo estas:

- Responsable: indica la persona del grupo responsable de la realización de la tarea.
- Fecha de Entrega: permite determinar la fecha en la que se debe entregar la tarea determinada.
- Proyectos: permite determinar a qué proyecto o proyectos pertenece la tarea.
- Dependencias: indica con qué tareas comparte dependencia la tarea determinada.
- Prioridad: que puede tomar valores de Bajo, Medio, Alto pudiendo ser modificadas en caso de que sea necesario.
- Nivel de dedicación: puede tomar valores de Pequeño, medio, grande, indicando el nivel de dedicación necesario para la susodicha.
- Progreso: indica el porcentaje de progreso alcanzado con la tarea.
- Descripción: breve descripción de la tarea.

En la siguiente Figura 11 podemos observar las características anteriormente mencionadas:

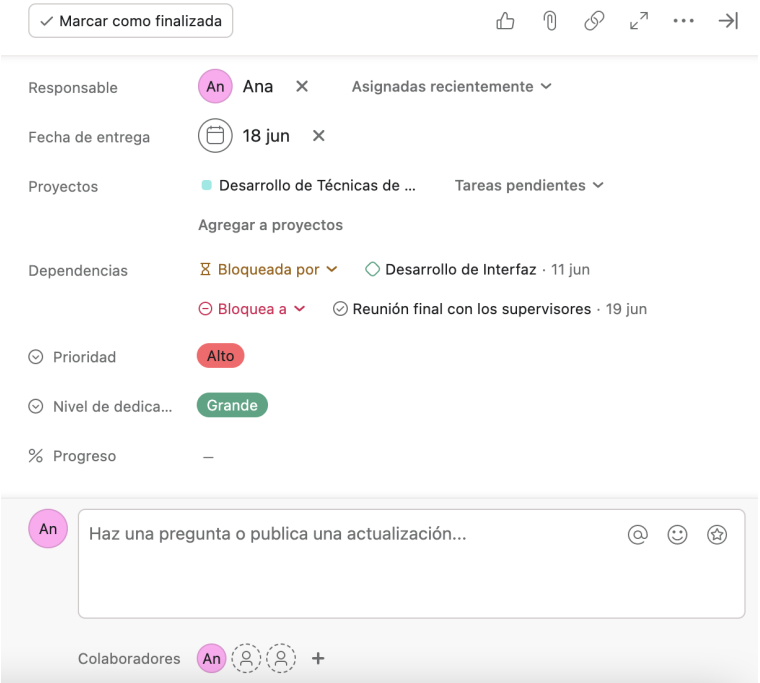

Figura 11: Características de las tareas en Asana

Las principales ventajas de Kanban es que nos proporciona las herramientas necesarias para evitar procesos innecesarios, es decir, aquellos que reducen la eficiencia o maximizan el tiempo de trabajo, nos permite concienciarnos sobre el desarrollo, ya que proporciona un control y una comunicación sobre el estado del producto de forma sencilla permitiendo al equipo estar actualizado de forma constante, nos aporta una motivación extra ya que el equipo se ve más único, con mayores capacidades para plantear y resolver problemas a medida que van saliendo, facilita una flexibilidad impresionante a la hora de enfrentar los cambios que puedan producirse en el producto y que se pueden localizar en breves periodos de tiempo ya que las tareas no son de mucha duración y además, nos permite una mayor eficacia ya que optimiza los tiempos, mejorando la calidad en menor tiempo.

Planificación temporal del proyecto

Debido a que es un proyecto de fin de grado, está realizado de forma individual, con lo cual para el desarrollo de este se utilizará un único recurso (el alumno). Esto no se da de forma general, ya que en un proyecto normal existe más de un recurso y cada uno de ellos seleccionan las tareas a completar que se encuentran en el tablero.

La diferencia principal en Kanban con otras metodologías es que se fundamenta en mostrar el progreso del proyecto mediante el desarrollo de las tareas en el tablero, no en las etapas. Por ello, se ha dividido la evaluación del proyecto en ciclos y se han definido todas las tareas que se van a realizar, organizándose dentro de secciones más generales que formarán las tareas principales (las cuales posteriormente marcarán los hitos) con las subtareas relacionadas. Cada tarjeta contendrá la persona responsable de la tarea, la fecha de finalización, el nivel de prioridad, el nivel de dedicación, el progreso y las dependencias con otras tareas, tal y como podemos observar en la Figura 12.

Para una información con más detalle, como la explicación de cada una de las fases, se puede consultar el *Anexo I: plan de Proyecto Software*, adjunto a esta memoria.

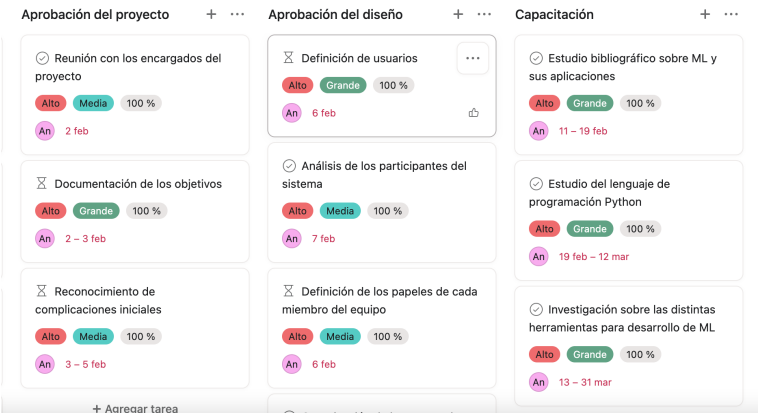

Figura 12: Planificación del proyecto en Asana

Inicio

Se tratará de la definición de todas las tareas a realizar en el proyecto, los objetivos que se quieren alcanzar y los medios para lograrlo. Debido a que no se tiene un amplio conocimiento sobre este área de estudio y es un requisito importante a la hora de seguir el proyecto, he incluido como tareas iniciales que permitan comenzar a desarrollar el proyecto las siguientes:

- Aprobación del proyecto

Lo primero necesario es aprobar la idea por parte de todos los miembros que componen el proyecto, esto es debido a que el título como tal es bastante genérico y se ha considerado necesario especificar más la línea de investigación sobre el tema, por lo cual era necesario plantear una tarea en la cual se informará a los supervisores del proyecto del campo de investigación seleccionado. Esta tarea principal se ha subdividido en varias tareas para su consecución como podemos ver en la Figura 13:

- Reunión con los encargados del proyecto.
- Documentación de los objetivos.
- Reconocimiento de las complicaciones iniciales.

- Aprobación del diseño

Tras la elección del campo de investigación se considera necesario determinar a qué usuarios va a ir dirigido, quién se va a encargar de las diversas tareas y varias características más que se verán determinadas por las siguientes subtareas:

- Definición de usuarios.
- Análisis de los participantes del sistema.
- Definición de los papeles de cada miembro del equipo.
- Organización de las pautas de trabajo a seguir.
- Definición de los tiempos.
- Estimación del gasto.
- Selección del lenguaje de programación a usar.
- Selección del Software a usar.

- Capacitación

Debido a que el tema elegido no es un tema con el que el alumno esté familiarizado es necesario una tarea que permita investigar sobre el tema a tratar para poder conseguir los objetivos necesarios, en este caso es necesario el estudio del campo de investigación, del lenguaje a usar debido a que no ha sido utilizado durante los estudios del alumno y de las herramientas necesarias para su construcción, determinado entonces por las siguientes subtareas:

- Estudio bibliográfico sobre ML y sus aplicaciones.
- Estudio del lenguaje de programación Python.
- Investigación sobre las distintas herramientas para el desarrollo de ML.

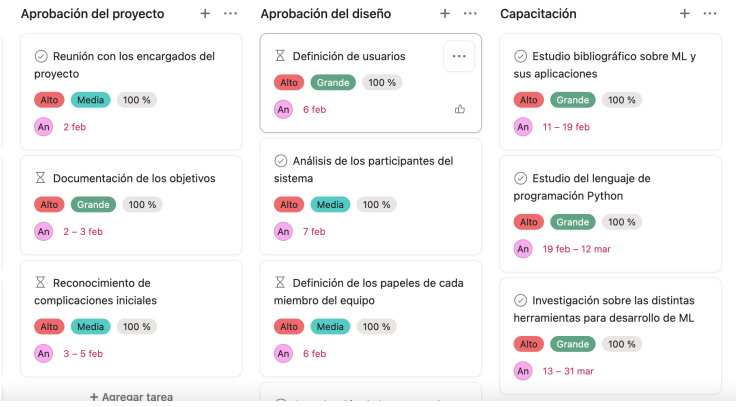

Figura 13: Fase "Inicio" en la planificación temporal

Etapa intermedia

Caracterizada por la realización de todas las tareas relacionadas con el desarrollo del modelo como tal, se compone de las siguientes tareas principales como podemos ver en las *Figuras 14 y 15*:

- Creación del modelo: Tras determinar todos los objetivos a conseguir y capacitar al alumno para la consecución de las tareas, es necesario comenzar una etapa para comenzar el desarrollo como tal del modelo de Machine Learning (ML), formada por las siguientes subtareas:
    - Reunión con los supervisores del proyecto.
    - Revisión de los objetivos aceptados con anterioridad.
    - Determinar aspectos importantes del desarrollo.
    - Selección de la red neuronal a usar.
    - Selección y preparación de los datos a usar.

- Pruebas: Una vez se ha comenzado a desarrollar el modelo, es necesario comprobar su funcionamiento mediante pruebas, que se dividen en las siguientes subtareas:
    - Pruebas sobre lo desarrollado.
    - Recomendaciones teóricas.

- Recolección de los comentarios: tras lo anterior es necesario que para seguir con el proyecto exista una tarea para comprobar que todo se está llevando a cabo de manera eficaz, por lo tanto se ha dividido esta tarea en las siguientes subtareas:
    - Reunión con los supervisores.
    - Confirmar los resultados obtenidos.
    - Confirmar los objetivos alcanzados.
    - Evaluar el progreso realizado.

- Corrección de errores: Tras el desarrollo del modelo se ha considerado necesario una etapa de evaluación de los errores que se hayan podido encontrar y por tanto la búsqueda de soluciones para cada uno de ellos, se divide en las siguientes subtareas:
    - Depuración del modelo.
    - Evaluación de hiperparámetros y algoritmos usados.
    - Mejora de resultados.
    - Revisión de datos utilizados.
    - Reunión con los supervisores.

- Desarrollo de una interfaz: Tras la creación del modelo se ha considerado desarrollar una interfaz sencilla que facilite el uso a los usuarios a los que va dirigida, teniendo en cuenta que éstos tienen ciertos conocimientos en programación y en el uso de cuadernos de Jupyter y Google Colab no sería necesario pero sí que proporciona más seguridad a la hora de ser usado, por tanto se ha dividido en las siguientes subtareas:
    - Diseño de la aplicación.
    - Funcionalidades de la aplicación.
    - Recomendaciones teóricas.
    - Recomendaciones prácticas.

- Pruebas sobre la interfaz: una vez desarrollada la interfaz es necesario realizar pruebas sobre ella para comprobar que se va a dar un uso correcto de ella, por tanto se ha subdividido en las siguientes tareas:
    - Pruebas sobre el diseño de la interfaz.
    - Revisión de funcionalidad.
    - Mejoras sobre la aplicación.

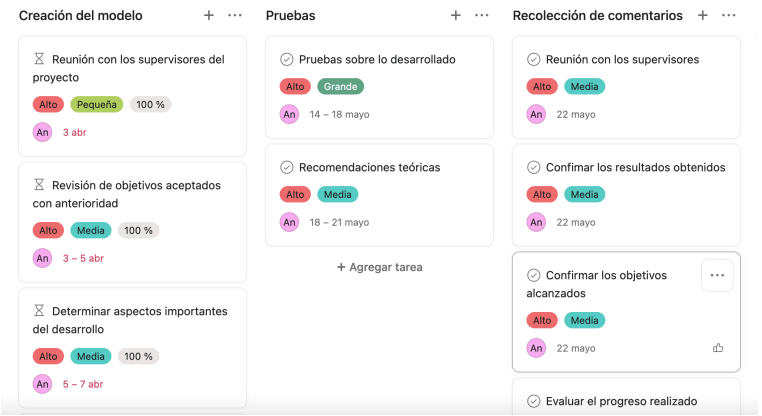

Figura 14: Fase Intermedia en la planificación temporal-1

Figura 15: Fase Intermedia en la planificación temporal-2

Revisión y conclusiones finales

Por último como conclusión final del proyecto, se ha estimado que se deben realizar unas subtareas que indicaré a continuación para completar el proyecto y además mencionar que su duración suma un total de 21,2 semanas aproximadamente:

- Aprobación final: se trata de una serie de subtareas para completar todo el trabajo realizado y determinar si está listo para su entrega, como podemos observar en la Figura 16. Se divide en:
    - Reunión final con los supervisores.
    - Puesta en marcha de la aplicación.
    - Implementación de las mejoras.
    - Elaboración de la documentación final.
    - Implementación de las mejoras.
    - Últimas revisiones y entrega.

Figura 16: Fase Final en la planificación temporal

Dentro de las opciones que nos permite la aplicación de Asana, también podemos ver las tareas a realizar como un cronograma, como se muestra en la *Figura 17* donde también nos muestra sus dependencias y las tareas críticas, nos permite ver mediante un panel las tareas que hemos finalizado, las que no y las totales o las próximas a realizar, mediante gráficas también como podemos ver en la *Figura 18* e incluso nos permite también ver las tareas mediante una lista de forma que se vea mucho más clara la organización del proyecto y comprobar que el transcurso del proyecto avanza de la manera planificada inicialmente, como podemos observar en la *Figura 19*:

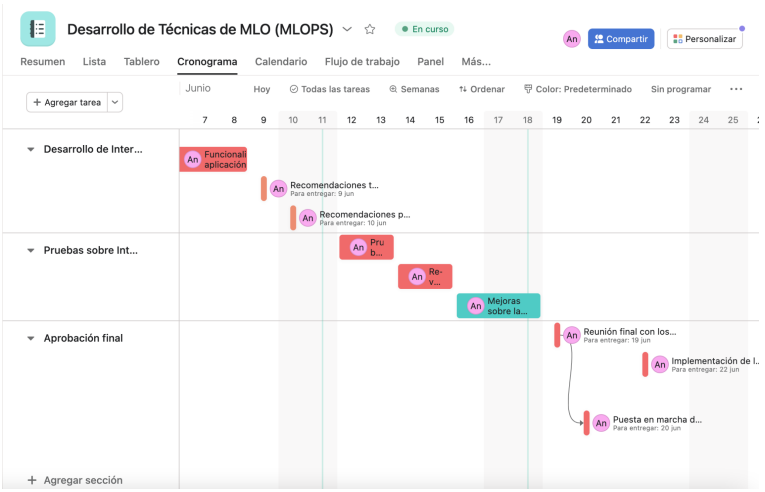

Figura 17: Cronograma

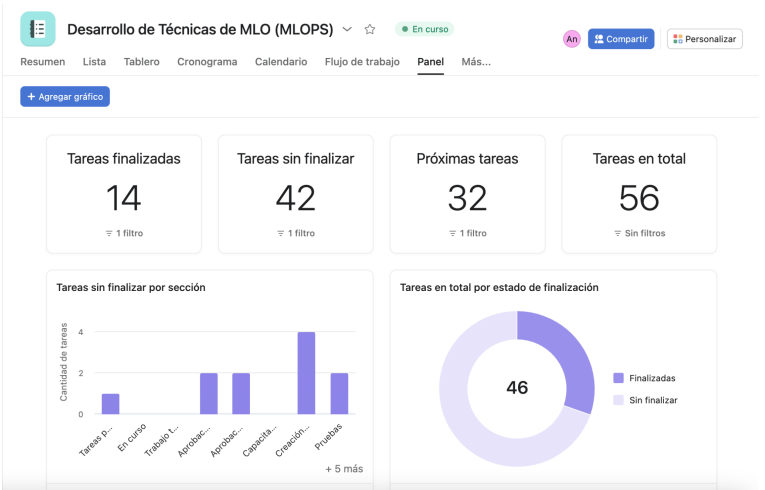

Figura 18: Panel de tareas

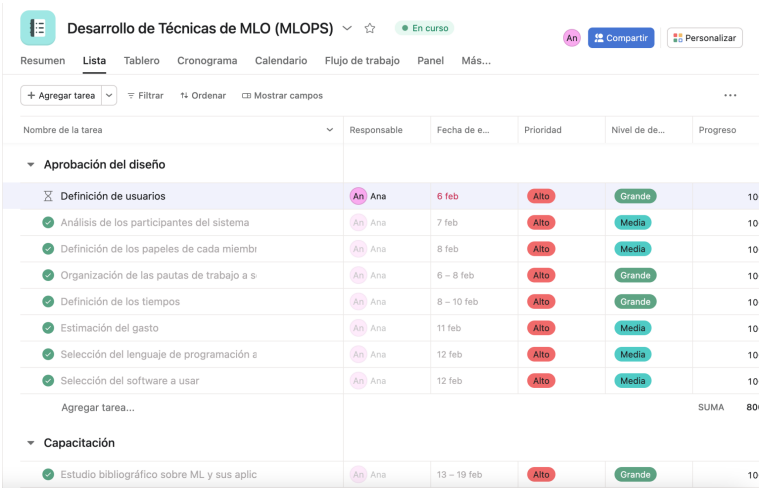

Figura 19: Lista de Tareas

Como resumen general de la planificación del proyecto, que podemos ver en el *Anexo I - Planificación Temporal* con mayor detalle. a continuación se presentan las tareas que representan hitos que marcarán las fechas estimadas siguiendo la planificación anterior:

Tabla 1: Hitos del proyecto

Hito	Fecha
Aprobación del Proyecto	5 de Febrero 2023
Aprobación del Diseño	10 de Febrero de 2023
Capacitación	31 de Marzo de 2023
Creación del modelo	14 de Mayo de 2023
Pruebas	21 de Mayo de 2023
Recolección de comentarios de los miembros	22 de Mayo de 2023
Corrección de errores	31 de Mayo de 2023
Desarrollo de la Interfaz	11 de Junio de 2023
Pruebas sobre la Interfaz	18 de Junio de 2023
Aprobación Final	30 de Junio de 2023

Aunque la mayoría de las tareas se llevaron a cabo correctamente, surgió un cambio de última hora, una situación frecuente en muchos proyectos. Esto requirió ajustes en las fechas de desarrollo de la interfaz, lo que llevó a posponer el proyecto por algunos meses. A pesar de estos cambios, las modificaciones no generaron complicaciones significativas en la finalización del proyecto. Todas las modificaciones se encuentran explicadas en el *Anexo I - Planificación Temporal.*

Estimación de costes

A la hora de realizar una estimación de costes se debe de tener en cuenta no solo el coste del material sino también el de los recursos humanos (personas) de los que disponemos, por tanto a continuación se presentan por separado ambas categorías:

- Estimación de costes de los materiales

El gasto que tendremos en cuenta es el de los materiales hardware utilizados, en este caso el del ordenador del que disponemos ya que utilizaremos un cuaderno para entrenar y lanzar nuestro modelo ya sea de forma local (en ciertas ocasiones) o como haremos generalmente, en la nube, por lo tanto no suponen coste para el proyecto.

- Estimación de costes de los recursos humanos

Se considera necesario una primera definición de los recursos de los que se poseen durante el transcurso del proyecto. Los roles se dividirán entre los jefes de proyecto que se encargarán de revisar el trabajo a realizar y el alumno, el cual tomará el resto de responsabilidades las cuales serán de analista y programador, esto es debido a que como proyecto de fin de grado es necesario considerarlo un proyecto con un fuerte componente de individualidad.

Vamos a considerar que se trata de una jornada laboral completa a la cual le incluimos los fines de semana debido a que como hemos mencionado anteriormente, se trata de un trabajo de fin de grado y se dispone de tiempo extra ya que se cuenta con pocas asignaturas que pudieran restar tiempo efectivo al trabajo, con una duración del proyecto de 21,2 semanas, que se amplia a las 29,14 semanas tras las modificaciones realizadas. La estimación total del alumno sería de 208 días incluyendo como hemos dicho anteriormente fines de semana, con lo cual eso hace un total de 1456 horas como analista y programador, los dos supervisores del TFG como jefes de proyecto se les asignará aproximadamente una hora semanal para reuniones, resultando en un total de 29 horas.

Finalmente, teniendo en cuenta que esto es simplemente una aproximación y que el alumno (analista-programador) no va a recibir ningún tipo de compensación económica, tras la evaluación anterior podemos resumir los avances en la siguiente tabla:

Tabla 2: Estimación de costes humanos

	Número	Salario	Horas	Coste
Jefe de Proyecto	2	15	29	435€
Analista-Programador	1	0	1456	0€

7. Implementación

Para la realización de la implementación se puede diferenciar entre la que se va a llevar a cabo para el modelo de la red neuronal y la que se va a llevar a cabo para la aplicación web que vamos a desarrollar.

Todo el código se encuentra en el siguiente repositorio: https://github.com/anagm98/model_cell_id.git

Red neuronal artificial

Como hemos definido con anterioridad, una red neuronal [10] ofrece los medios para modelar de manera eficiente y efectiva problemas grandes y complejos, permitiendo mediante unos datos hallar patrones o relaciones de una manera inductiva mediante algoritmos de aprendizaje basados en los datos proporcionados. Son un método para resolver problemas ya sea de forma individual o mediante una combinación con otros métodos que necesitan que el balance datos/conocimiento se se incline hacia los datos, pudiendo reajustar los pesos de sus interconexiones.

Su estructura está basada en el cerebro humano, en particular en el sistema nervioso, compuesto por redes de neuronas biológicas que poseen de forma individual bajas capacidades de procesamiento, sin embargo, al conectarse entre ellas permiten mucha mayor capacidad cognitiva.

Neurona artificial

Elemento fundamental y básico de procesamiento en una red neuronal artificial, constituyen dispositivos simples de cálculo cuyo funcionamiento se basa en un vector de entrada procedente o bien del exterior o bien de estímulos recibidos de otras neuronas, proporcionando una respuesta o una salida. Existen varios tipos de neuronas artificiales: de entrada, de salida y ocultas que son las que se encargan de recibir estímulos y emitir salidas dentro del sistema, sin tener contacto con el exterior, es donde se lleva a cabo el procesamiento básico de la información (generando la representación interna de ésta). Cada neurona artificial se caracteriza por los siguientes elementos:

- **Estado de activación inicial** (anterior a la recepción de estímulos).
- **Estímulos de entrada a la neurona con unos pesos asociados.**
- **Función de propagación**, que determina la entrada total a la neurona.
- **Función de activación o transferencia**, que combina las entradas a la neurona con el estado de activación inicial para generar un nuevo valor de activación. Sin estas, las redes neuronales solo podrían aprender relaciones lineales, es decir, el modelo funcionaría como un modelo de regresión lineal y no podríamos obtener relaciones no lineales entre los datos de entrada y los de salida. De forma general encontramos los siguientes tipos de funciones:
    - Función de activación ReLU: aplica una transformación no lineal muy simple, activa la neurona solo si el input está por encima de cero. Mientras el valor de entrada sea menor que 0 el valor de salida se mantiene a 0, en el momento en

el que el valor de entrada aumenta, el valor de salida aumenta de forma lineal con el de entrada. Es la función usada por preferencia.
    - Sigmoide: transforma valores en el rango de (-inf, +inf) a valores en el rango (0,1). Se usa actualmente en modelos de clasificación binaria ya que su salida puede interpretarse como probabilidades.
    - Tangente hiperbólica (Tanh): se comporta similar a la función sigmoide pero su salida está acotada en el rango de (-1, 1).
    - Leaky ReLU: transforma los valores introducidos multiplicando los negativos por un coeficiente rectificativo y dejando los positivos según entran.
    - Softmax: transforma las salidas a una representación en forma de probabilidades, de tal manera que el sumatorio de todas las probabilidades de las salidas de 1 (representación en forma de probabilidades).
- **Función de salida**, que transforma el estado final de la activación en la señal de salida.
- **Señal de salida**, que se transmite a otras neuronas artificiales. Se considera un conjunto finito de salidas, mientras que las tareas de ajuste de funciones suelen precisar salidas continuas en un determinado intervalo. El tipo de salida deseada determinará la función de transferencia y salida que debe implementarse en las neuronas que se encuentren en la última capa de la red.
- **Regla de aprendizaje**, que determina la forma de actualización de los pesos en la red.

Como tal, la neurona puede resolver problemas lineales, pero a la hora de resolver problemas no lineales encontramos limitaciones, por lo tanto, para solventar este problema se realizan agrupaciones de neuronas artificiales secuencialmente, lo que formalmente llamamos redes neuronales artificiales.

Redes neuronales artificiales

Agrupaciones de neuronas artificiales organizadas en capas conectadas entre sí. Como se ha mencionado anteriormente, su estructura normalmente se divide en tres capas, la de entrada que representa el vector de características de entrada, las capas ocultas que es dónde se produce el aprendizaje y la capa de salida, encargada de facilitarnos el resultado. La cantidad depende del problema de aprendizaje ante el que nos encontremos.

Dentro de esta estructura podemos observar distintos tipos de estructuras dependiendo de las direcciones que tomen las salidas de las neuronas. Según el flujo de datos en la red podemos encontrar las redes unidireccionales o feedforward, cuyas conexiones de salida sólo se relacionan con capas posteriores y las redes recurrentes o feedback que son aquellas cuyas las conexiones de salida se relacionan con distintas capas independientemente de su situación dentro de la red.

Función de pérdida

También llamada función de coste [11] (loss function) es otro concepto importante para entender las redes neuronales artificiales, nos permite medir la efectividad del modelo, mediante los resultados que nos proporciona podemos mejorar nuestro modelo minimizando los errores. Se suele usar en los algoritmos de aprendizaje supervisado y sirve para estudiar la desviación entre las predicciones que hemos obtenido con el modelo y el valor real. Es una función especialmente útil debido a que uno de los objetivos principales de cualquier red

neuronal se basa en minimizar el resultado de la función de pérdida ya que eso nos indicaría que se ha producido una mejora en la precisión del modelo. Dependiendo del problema podemos encontrarnos con distintos tipos de funciones de pérdida:

- **Log loss, logistic loss o cross-entropy loss o pérdida de entropía cruzada**, empleada en problemas de clasificación, la capa de salida utiliza como función de activación la función softmax. La red devuelve una serie de valores que pueden interpretarse como la probabilidad de que la observación predicha pertenezca a cada una de las posibles clases.
- **Error cuadrático medio** (mean squared error, MSE), función de coste más usada en problemas de regresión.
- **Error medio absoluto** (mean absolute error, MAE) consiste en promediar el error absoluto de las predicciones, mediante esta función, el entrenamiento del modelo se ve menos influenciado que el error cuadrático medio por datos anómalos del conjunto de entrenamiento.

Propagación del error

También conocido como backpropagation [12], se trata de una técnica de entrenamiento que permite que la red neuronal aprenda y corrija sus errores, en modelos de redes neuronales el error se propaga hacia atrás desde la señal de error a otras capas, esto es debido a que el error de capas anteriores influye directamente en el de capas posteriores. Cuanto más profundicemos en su propagación más se disuelve el error, haciendo que en las capas más alejadas la variación de los pesos sea mucho menor.

Hiperparámetros

Las redes neuronales se caracterizan en parte por su gran flexibilidad, esto es debido a que por una parte, son capaces de generar modelos que aprenden relaciones muy complejas pero en ciertos casos pueden sufrir de lo que se llama un problema de sobreajuste (overfitting), esto produce una incapacidad al tratar de predecir sobre datos que no se habían usado con anterioridad, la forma que tenemos de minimizar este problema y conseguir modelos que nos sirvan pasa por configurar los hiperparámetros de forma adecuada, entre ellos encontramos:

- **Número y tamaño de las capas**

  Las capas [13] determinan en gran medida la complejidad del modelo y por tanto su potencial capacidad de aprendizaje. La capa de entrada y de salida son sencillas, la de entrada tiene tantas neuronas como predictores y la de salida tiene una neurona en problemas de regresión y tantas como clases en problemas de clasificación. Estos valores generalmente se establecen de forma automática en función del conjunto que tengamos de entrenamiento. Como tal, se suele especificar normalmente el número de capas intermedias (ocultas) y el tamaño de las mismas. A más neuronas y capas, mayor complejidad de las relaciones que es capaz de aprender el modelo, si el número de parámetros a aprender aumenta, aumenta con ello el tiempo de entrenamiento.

- **Learning rate (ratio de aprendizaje)**

  Establece la rapidez con la que pueden cambiar los parámetros de un modelo a medida que se aprende (optimiza). Es uno de los más complicados de establecer, ya que depende en gran medida de los datos e interacciona con el resto de hiperparámetros. Si es muy grande, es posible que el proceso de optimizar salte de una región a otra sin ser capaz de aprender. En cambio, si es muy pequeño el proceso de entrenamiento puede tardar mucho sin llegar a su compleción.

- **Algoritmo de optimización**

  Los primeros en ser utilizados fueron el descenso de gradiente y el descenso de gradiente estocástico, el problema que presentaban es que daban errores a mayor tamaño de las redes, es decir, del número de capas y de neuronas. En muchas regiones del espacio de búsqueda el gradiente se aproxima mucho a cero, lo que hace que se quede estancada en estas regiones. Como solución se desarrollaron modificaciones del descenso de gradiente que pudiesen adaptar el learning rate en función del gradiente y subgradiente. Acelerando o desacelerando el proceso de aprendizaje según las características de la región del espacio de búsqueda en el que se encuentren. Su elección tiene un impacto muy grande en el aprendizaje de los modelos.

- **Regularización**

  Se usan para reducir en la medida de lo posible el sobreajuste (overfitting) de los modelos, ya que un modelo que sufre de sobreajuste memoriza los datos de entrenamiento pero no puede predecir correctamente sobre datos que no ha visualizado previamente.

Una de las formas en que aplicamos las técnicas de Machine Learning Operations (MLOps) es mediante el ajuste de hiperparámetros. El modelo que hemos desarrollado está diseñado para encontrar automáticamente los valores óptimos de estos hiperparámetros. Esto significa que cualquier persona que desee utilizar el modelo no tendrá que perder tiempo ajustándolos manualmente. Este proceso se considera una parte crucial de la automatización en el monitoreo y mantenimiento del modelo. En el contexto de MLOps, este enfoque se conoce como 'autoajuste' u 'optimización automática' y es especialmente relevante.

Distribución del modelo

Como hemos mencionado anteriormente, la estructura de nuestro modelo se basa en las redes U-Net, un sistema creado por Ronneberger et al. y Long et al. capaz de combinar la información semántica con los detalles locales de la imagen. Su nombre proviene de su arquitectura en forma de U.

Para implementar la funcionalidad de segmentación de imágenes, se utilizó un modelo de red neuronal U-Net, cuyo código base fue obtenido de un repositorio de GitHub [14]”

Por supuesto se realizaron modificaciones específicas para adaptar el modelo a nuestro conjunto de datos y requisitos de implementación. En específico los pasos que se realizaron en el cuaderno de Google para la consecución del modelo fueron los siguientes:

- Una parte de **Imports**, con todos los *imports* necesarios para poder generar el modelo de aprendizaje automático.

- **Rutas para las imágenes:** todas las rutas necesarias para acceder a los datasets que van a ser usados tanto para entrenamiento como para validación. Rutas que pueden ser modificadas para poder reentrenar el modelo con datos adicionales, permitiendo mantener y mejorar el rendimiento del modelo con el tiempo, y dotando al modelo de la capacidad de adaptarse a cambios en la distribución de datos y necesidades de rendimiento.

- **Visualización de imágenes con máscaras:** una serie de funciones que nos sirven para comprobar el formato de las imágenes y de las máscaras, realizando las transformaciones necesarias con el objetivo de poder usarlas en el modelo.

  En este apartado se recorren todos los archivos dentro de la carpeta de entrenamiento y para cada imagen se abre su correspondiente máscara, las convierte en arreglos NumPy y muestra la información de cada archivo.

  El objetivo de esto es entender mejor la estructura de las imágenes y máscaras utilizadas en el proyecto, todo con el objetivo de garantizar y facilitar su mantenimiento en el tiempo.

  Además se incluye una función que convierte las máscaras multiclase en máscaras binarias, esto es debido a que el dataset que estábamos usando utilizaba una máscara por cada célula y se buscaba entrenar al modelo a diferenciar entre las “células” de las “no células”, de manera que se cubran todas las posibilidades a la hora de usar máscaras.

- **Creación del dataset para entrenamiento y validación:** se centra en la creación de un conjunto de datos personalizado para entrenamiento y validación del modelo.

  Generamos un conjunto de datos personalizado, de manera que el formato de las imágenes o su disposición en la carpeta no sea un impedimento para usarlas en el entrenamiento de modelo.

- **Visualización de datos:** simplemente un código para ver de una forma visual las imágenes con sus respectivas máscaras.

- **Funciones para entrenamiento del modelo:** aquí encontramos las funciones principales que se usan a la hora de entrenar el modelo. Donde encontramos las siguientes:

  - Función *accuracy*: se encarga de calcular la precisión del modelo utilizando el conjunto de datos de los Dataloader (los que proporcionan los datos para evaluar el modelo). Esta función es especialmente importante en el contexto de

Machine Learning Operations (MLOps) ya que permite evaluar el rendimiento del modelo en tiempo real, relacionándose con uno de los objetivos de monitorear modelos en producción. La precisión es una métrica clave que permite evaluar qué tan bien está funcionando el modelo en comparación con los datos de prueba o validación.

    - Función *train*: función que como su nombre indica, se encarga del entrenamiento en sí del modelo. Posteriormente se explicará con más detalle.
    - Función *find_learning_rate:* función que se encarga de encontrar la tasa de aprendizaje óptima utilizando el método de búsqueda de tasa de aprendizaje. Esta es otra de las funciones especialmente importantes en el contexto de Machine Learning Operations, esto es debido a que puede considerarse parte del monitoreo y mantenimiento del modelo, encontrar el learning rate óptimo es una parte crítica en el proceso de entrenamiento del modelo, tener una función que ayude a encontrarlo es una práctica común en MLOps, ya que garantiza que el proceso de entrenamiento sea más eficiente y efectivo. Esto también puede ser parte de un proceso de optimización continua, lo que se relaciona con el mantenimiento del modelo.

- **Entrenamiento:** entrenamiento del modelo.

- **Prueba sobre imágenes:** simplemente es la prueba sobre las imágenes del set de pruebas para comprobar el funcionamiento de nuestro modelo.

- **Guardamos el modelo:** líneas que sirven para guardar el modelo en formato .pth como hemos visto anteriormente, con el objetivo de usarlo en el script que usará el modelo dentro de la interfaz, además de permitir a cualquier usuario guardar y gestionar modelos, siendo esto una parte importante en la gestión del ciclo de vida de Machine Learning.

Visualmente se pueden diferenciar claramente dos partes principales en el modelo, una ascendente y otra descendente como podemos observábamos en la Figura 4.

La primera de ellas consta de una serie de capas de convolución que se mezclan con otras de downsampling (también llamadas de agrupación o pooling) que se basan en disminuir la resolución y de extraer la significación semántica tanto de la imagen como de distintas partes de la misma. La segunda fase, ascendente, es la que se encarga de construir el mapa de segmentación semántica y está formada por capas de convolución e incremento de la resolución (upsampling).

De forma general podemos entender la convolución en dos sentidos: si se acompaña de una disminución de resolución (downsampling, pooling) se encarga de extraer rasgos de la imagen inicial, por el contrario, si viene acompañada de incrementos de resolución su objetivo es el de añadir rasgos y estructuras a la imagen que esté procesando. En la siguiente Figura 20 se muestra un esquema de forma general sacado de nuestro código con los nombres que se les ha puesto a cada una de las funciones que representan las convoluciones.

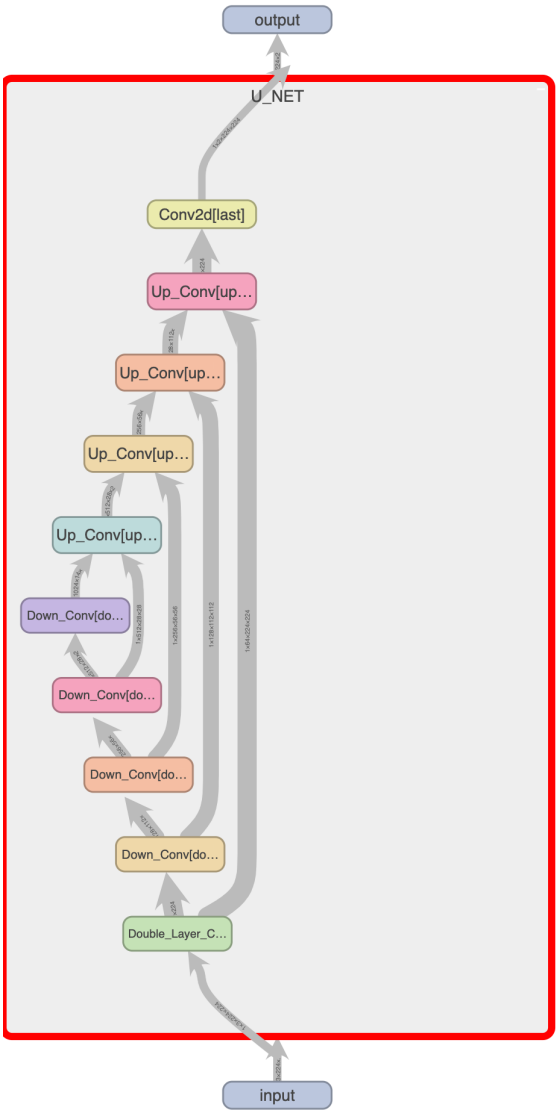

Figura 20: Estructura del modelo desarrollado (U-Net)

Determinar los hiperparámetros de forma manual puede llegar a ser un proceso complicado que se facilita automatizándolo, gracias a funciones que calculan los parámetros óptimos para el modelo que estamos utilizando, además de definirlos en varias partes del código, tanto en la instancia del propio modelo, como en las creaciones del optimizador del modelo o del scheduler, el cual se trata de un componente que ajusta de forma automática la tasa de aprendizaje durante el entrenamiento del modelo, permitiendo configurar la tasa de aprendizaje de una forma dinámica a medida que se va entrenando.

Entrenamiento

Dentro del código tenemos una función llamada *train* (entrenamiento) que se encarga de realizarlo, dentro de ella se ajustan los parámetros del modelo para que pueda aprender de los datos de entrenamiento que anteriormente hemos ajustado para poder ser usados y de esta manera realizar predicciones más precisas. Esto es, como hemos mencionado anteriormente, una parte crítica del proceso de entrenamiento de modelos, tener una función que ayude a encontrar el Learning Rate óptimo es una práctica común en Machine Learning Operations (MLOps), garantizando que el proceso de entrenamiento sea más eficiente y efectivo. El proceso es el siguiente:

- **Preparación**: antes de comenzar el entrenamiento, el modelo se traslada a la unidad de procesamiento de gráficos (GPU) si está disponible ya que realiza cálculos mucho más potentes, acelerando el proceso. La función itera a través de un número determinado de épocas, que representan como hemos mencionado anteriormente la cantidad de veces que el modelo verá todos los datos de entrenamiento.

- **Iteración sobre los lotes**: dentro de cada época, el modelo itera a través de los datos de entrenamiento que le hemos proporcionado. Un lote es un pequeño grupo de ejemplos de datos que se utilizan para actualizar los parámetros del modelo. Cada lote contiene imágenes con sus correspondientes máscaras, utilizadas para entrenar el modelo desarrollado en tareas específicas, como en este caso de detección de células.

- **Cálculo de predicciones**: para cada lote, el modelo toma las imágenes de entrada y calcula sus predicciones utilizando una serie de capas y operaciones matemáticas. Las predicciones son estimaciones de las clases o características que el modelo aprende a reconocer en las imágenes proporcionadas, como en nuestro caso, las células.

- **Cálculo de pérdida:** tras realizar las predicciones, se calcula la pérdida o costo. Esta medida determina cuán diferentes son las predicciones del modelo de las máscaras reales proporcionadas de cada imagen. El objetivo es conseguir minimizar la pérdida, ajustando los parámetros.

- **Retropropagación:** tras el cálculo de la pérdida, da lugar el proceso de retropropagación. Los gradientes, que representan cómo cambiar los parámetros para reducir la pérdida mencionada anteriormente, se calculan y propagan a través de la red neuronal en sentido contrario, es decir, hacia atrás.

- **Actualización de parámetros:** tras el cálculo de los gradientes, el optimizador ajusta los parámetros del modelo para disminuir la pérdida en el siguiente ciclo. Se hace de poco a poco este paso.

- **Control de la tasa de aprendizaje:** opcionalmente usamos un scheduler, para controlar el cambio de la tasa de aprendizaje durante el entrenamiento permitiendo una mejora en el modelo que supone una convergencia más rápida y mejora su desempeño.

- **Métricas de entrenamiento y validación:** se calculan en cada iteración de los lotes, proporcionan una idea de cómo está aprendiendo el modelo y de si sus predicciones mejoran a medida que avanza el entrenamiento.

- **Visualización de resultados:** opcionalmente, en intervalos regulares, es necesario imprimir resultados de entrenamiento, como puedan ser la pérdida, la precisión u otras métricas, permitiéndonos monitorear el progreso y realizar ajustes en caso de que sea necesario.

Como resumen, la función de entrenamiento o *train* como dice su propio nombre, es el núcleo de entrenamiento del modelo. A través de un proceso iterativo, ajusta de una forma gradual todos los parámetros del modelo con el objetivo de mejorar sus predicciones en función de los datos de entrenamiento. Cada época representa un ciclo completo a través de los datos con el objetivo final de que el modelo aprenda patrones en los datos que nos permita en un futuro realizar predicciones precisas en imágenes que no ha visto antes.

Validación

La validación en el entrenamiento de nuestro modelo se realiza principalmente en una función llamada accuracy, la cual calcula y devuelve métricas de rendimiento en el conjunto de validación (anteriormente separado dentro del dataset) con el objetivo de monitorear y evaluar la efectividad del modelo al ser entrenado, otra de las prácticas importantes y comunes en Machine Learning Operations (MLOps).

Tras cada época (epoch) en la función de train, se llama a la función accuracy para determinar el rendimiento del modelo en el conjunto designado para validación. La función calcula la pérdida promedio, la precisión, el índice de similitud de Imagen de Coeficiente Sørense-Dice (también conocido como “coeficiente DICE”) y el índice de superposición de unión (también conocido como “coeficiente IoU”) en el conjunto de validación.

Ambas métricas se utilizan para medir la similitud entre las máscaras de segmentación predichas por un modelo y las de referencia o “grountruth”. Como tal ambas tienen su rango de valores entre el 0 y el 1, donde el 0 indica que no hay superposición entre máscaras y 1 indica una superposición perfecta. En general, un valor alto en ambos coeficientes indica una mayor similitud y precisión entre segmentaciones.

Como tal ambas métricas se diferencian en su sensibilidad a falsos positivos y falsos negativos, el coeficiente DICE tiende a darle más peso a la presencia de falsos positivos y negativos, es decir, si hay una discrepancia en presencia o ausencia de objetos en las máscaras, el coeficiente DICE se verá más afectado.

Mientras se realiza el entrenamiento, se realiza un seguimiento de las métricas de rendimiento para el conjunto de entrenamiento y el conjunto de validación en cada mini lote. Esto permite monitorear cómo el modelo está mejorando tanto en los datos usados para el entrenamiento como en los usados para la validación.

Aplicación Web

Tal y como hemos mencionado anteriormente, el público objetivo de este trabajo son personas con conocimientos suficientes para ser capaces de utilizar cuadernos de Google Colab, aún así he considerado proporcionar una aplicación para facilitar la visualización de los resultados del modelo. Todo esto con el objetivo de realizar un entorno de trabajo que esté optimizado para su uso por los usuarios, tanto aquellos que disponen los conocimientos como los que no, optimizando el uso del modelo.

Proceso de diseño

En este apartado, se abordará el desarrollo de la interfaz del sistema que hemos desarrollado.

Dicha interfaz se guiará mediante el diseño centrado en el usuario, de manera que será validada con usuarios reales para comprobar su correcto funcionamiento y diseño.

De esta etapa obtendremos el diseño final de la aplicación, con el propósito central de habilitar al usuario que logre ciertos objetivos específicos dentro del ámbito de la aplicación.

En esta perspectiva, se integrarán directrices con el fin de cumplir con metas relacionadas con el intercambio de información eficiente, ya sean la minimización de errores, el aumento de la satisfacción del usuario y la mejora de la facilidad de uso, entre otros aspectos. Se considerarán factores como una jerarquía visual clara y la coherencia del diseño en la aplicación.

El esquema seguido en el documento se basará en las fases comprendidas en el *ciclo de diseño centrado en el usuario*, las cuales son:

1. Análisis de los competidores.
2. Definición de la audiencia.
3. Escenarios de uso.
4. Contenido de la encuesta
5. Flujos de proceso
6. Mapa de sitio
7. Esquema de página
8. Diseño
9. Prototipo
10. Pruebas de usuario
11. Revisión
12. Aprobación

Los cuales podemos observar en la Figura 21.

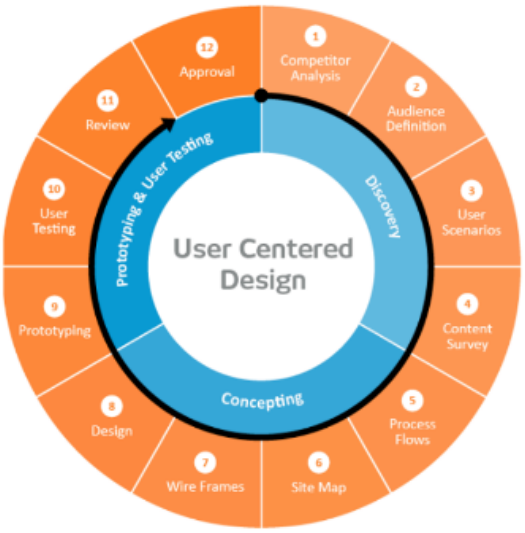

Figura 21: estructura del diseño centrado en el usuario

Cada decisión y paso está detalladamente especificado en el *Anexo III: Proceso de Diseño Centrado en el Usuario*.
A continuación, se presentarán los aspectos más destacados de cada una de las fases:

Fase Inicial

Esta fase se compone de las siguientes tareas:

- Búsqueda de necesidades
- Análisis de la competencia
- Definición de la audiencia
- Observaciones

Búsqueda de necesidades

Con la búsqueda de necesidades busco principalmente cumplir el objetivo de identificar a las personas a las que se dirige el resultado final del proyecto, elegir cuál de ellas va a resultar más necesaria para el público objetivo, identificar problemas subsistentes que puedan existir con cada una de ellas y establecer, mediante patrones, la aplicación que solucione cada uno de ellos.

En este sentido, mi propuesta inicial, ha surgido de la observación en el ámbito de la investigación.

Por tanto a partir de esta idea principal, se procedió a la realización de entrevistas, con el objetivo de averiguar cuáles son las principales necesidades y cómo solucionarlas.

Análisis de la competencia

A la hora de introducir nuevas propuestas, se debe analizar lo que actualmente se está realizando en este ámbito y, dentro de lo posible, adaptarla tras un análisis de la competencia, para poder posicionar nuestra aplicación frente a competidores o posibles competidores.

En el análisis de la competencia, se ha evaluado la oferta existente de aplicaciones destinadas al recuento de células en imágenes, con especial atención en herramientas como CellProfiler. Si bien aplicaciones como CellProfiler tienen la capacidad de realizar tareas complejas de análisis, se ha considerado que su interfaz es bastante compleja y tiene una gran cantidad de funcionalidades que pueden resultar abrumadoras para los usuarios.

La principal diferenciación que busca este proyecto es desarrollar una interfaz que se caracterice por su simplicidad y facilidad de uso. A diferencia de las soluciones existentes, que tienden a ofrecer una gran cantidad de funciones, el enfoque aquí es reducir la complejidad y presentar únicamente las características esenciales para el recuento de células en imágenes.

Definición de la audiencia

Un sistema se usa cuando se utiliza y se dirige por y para un grupo determinado y definido de personas, para realizar tareas concretas en un contexto específico.

La definición de la audiencia consiste en aclarar de manera detallada cuál es el usuario objetivo y sus características porque van a ser finalmente los que tengan que trabajar con el sistema desarrollado.

Es fundamental saber quién será el usuario y para quién está orientada la solución planteada.

Basándome en los resultados obtenidos en las diferentes entrevistas realizadas entre las cuales poseo más variedad de público, puedo concluir que mi público es un público amplio, de un rango de edad aproximadamente entre los 20 y 65 años, tanto investigadores como usuarios sin conocimientos en campos científicos.

Con lo cual tenemos como resumen:

- Audiencia principal: Usuario con conocimientos en IA/ML
    - Investigadores con conocimientos ligeros en programación e informática, los cuales han utilizado herramientas parecidas previamente.
        - Tienen interés en reducir los tiempos de ciertas actividades que puedan ser informatizadas.
        - Saben y conocen los términos de cuadernos de Google, Jupyter o el funcionamiento de forma general de la Inteligencia Artificial.
        - Saben usar los cuadernos de Google y Jupyter y podrían modificar el código con cierta ayuda.
        - Centros de investigación, principalmente del cáncer.
        - Usarán la aplicación sin explicaciones previas.

- Audiencia secundaria: Usuario sin conocimientos en IA/ML
    - Usuario común: personas sin conocimiento previo de programación o informática, los cuales no han utilizado herramientas parecidas con anterioridad.
        - Tienen interés por saber cómo funciona la inteligencia artificial.
        - No conocen los términos de cuadernos de Google, Jupyter o el funcionamiento de forma general de la Inteligencia Artificial
        - No saben usar los cuadernos de Google y Jupyter y no podrían modificar el código ni con ayuda.
        - Usarán la aplicación pero probablemente tendrán que visitar la vista “About us”.

Observaciones

Por tanto podemos determinar de una forma general que las necesidades observadas son:

- Dentro del grupo principal, como son los investigadores:
    - Obtener una herramienta capaz de usar modelos que permitan agilizar ciertas tareas, en específico en imágenes de microscopía, fundamentales a la hora de llevar a cabo múltiples funciones.
    - Detectar las células dentro de las imágenes con el fin de generar cálculos sobre características de cada una de ellas.
    - Capacidad de uso del modelo de una forma sencilla e intuitiva.
- Grupo secundario, es decir, personas sin conocimientos sobre programación:
    - Aprendizaje de forma visual sobre el funcionamiento de un modelo de aprendizaje automático.
    - Uso de las técnicas de machine learning de forma sencilla e intuitiva.

Fases del proceso de creación

Esta fase se compone de las siguientes tareas:

- Generación de ideas intuitivas
- Integración de funcionalidades
- Refinamiento Visual
- Coherencia, pruebas finales y ajustes

Dentro del contexto de una interfaz sencilla diseñada únicamente para carga y análisis de imágenes a través de un modelo de inteligencia artificial, el proceso de creación dentro del enfoque de diseño centrado en el usuario se presentaría de la siguiente manera:

Generación de ideas intuitivas

En esta fase identificamos los objetivos principales de la interfaz: carga y análisis de imágenes. Exploramos las opciones para una navegación sencilla y una presentación clara de la funcionalidad principal que hemos mencionado anteriormente.

Debido a que consideramos elementos con un diseño minimalista y eficaz para que los dos tipos de usuarios objetivos sean capaces de usarla nos quedamos con nuestra idea principal: una

vista en la que sea posible mediante un botón cargar imágenes y mediante otro botón analizarlas, así mantenemos la sencillez y solidez que buscamos en nuestro sistema.

Integración de funcionalidades

He considerado dejar esta fase ya que hemos tenido que integrar dentro de la página el uso del modelo, debido a que la página original sólo tenía un botón, se ha decidido añadir otro debajo, uno con el simple objetivo de cargar las imágenes que queramos y el segundo para poder analizarlas, esto debido a que me parece más sencillo y visual, de forma que el proceso de carga de imágenes goce de un enfoque en la simplicidad y claridad de instrucciones.

Refinamiento Visual

En cuanto a la vista como la vista de “About us” con la definición del sistema y la vista de “Contact us” he decidido mantenerlas, debido a que me parece que la interfaz así mantiene una estética coherente y agradable al usuario, siendo esta más parecida a muchas interfaces a las que el usuario suela visitar.

Considero importante mencionar aquí que se ha decidido mantener la estética de la interfaz utilizada pero, de forma general, se han realizado cambios especificados en el *Anexo III: Proceso de Diseño Centrado en el Usuario*.

Coherencia, pruebas finales y ajustes

Tras haber realizado todos los cambios se ha considerado realizar pruebas finales sobre la interfaz, con la conclusión de que todas las funcionalidades incorporadas funcionan correctamente.

Escenarios de uso

Esta fase se compone de las siguientes tareas:

- Representación de personajes
- Tipos de personas
- Tipos de escenarios

En este apartado se van a generar los escenarios de uso [16], que son narrativas ficticias con la finalidad de destacar las posibles experiencias de varios usuarios con el sistema de desarrollo.

En este sentido, tenemos que observar el comportamiento de nuestros usuarios.

Debemos describir la experiencia del beneficiario de nuestra aplicación, explicando por qué necesita utilizar nuestro sistema, las expectativas que pueda generar de éste y definir el tipo de emociones que está sintiendo al usarla, es decir, la conclusión es que debemos realizar una descripción de forma completa de las circunstancias que rodean la interacción de un usuario con nuestra aplicación, el por qué y cómo el usuario realiza determinadas acciones.

Representación de personajes

Desde el punto de vista de la teoría, los personajes que vamos a detallar son usuarios reales que harán uso de la aplicación que hemos desarrollado.

Tipos de personas

Se ha considerado que tenemos dos audiencias principales, como se ha mencionado anteriormente, encontramos la audiencia primaria y la audiencia secundaria. Por lo tanto en este apartado se usará la "definición de la audiencia" realizada con anterioridad.

Tipos de escenarios

En este apartado se decidió narrar de una manera explícita y actual la descripción de los diferentes personajes que han interactuado con nuestra aplicación.

Fase de conceptualización

Esta fase se compone de las siguientes tareas:

- Análisis de tareas
- Mapa de sitio
- Wireframes y diseño

En esta sección, el objetivo principal es desarrollar un prototipo digital que funcione como una primera representación de la futura implementación práctica.

Análisis de tareas

Este apartado se dedicará a determinar futuras interacciones entre los usuarios y las distintas acciones de la aplicación, más específicamente, se tendrá en cuenta lo que hace, las actuaciones y qué es lo que necesita conocer.

El flujo de la tarea principal de nuestro sistema es el siguiente:

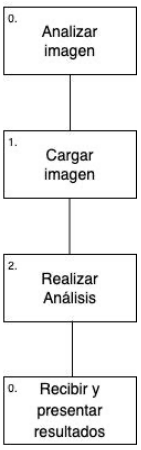

Figura 22: Tarea Analizar Imagen

Mapa de sitio

Procederemos a mostrar el mapa del sitio, el cual dispondrá de toda la información necesaria para que el usuario sea capaz de realizar las acciones en el orden correcto dentro de nuestro sistema planteado.

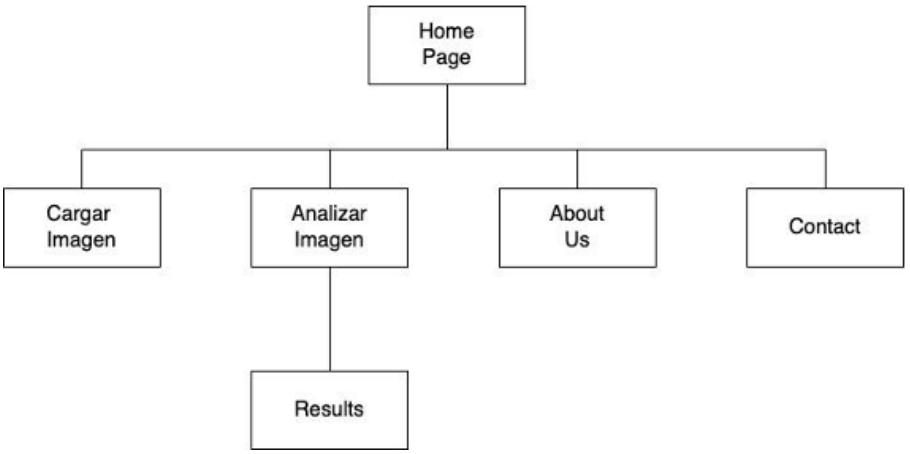

Figura 23: Mapa de sitio de los flujos de navegación de CellIdentifier

Wireframes y diseño

En la fase final de conceptualización, se aborda la representación visual de la aplicación una vez que se ha establecido el flujo de trabajo y la estructura del contenido.

Acompañando esto, es esencial justificar las elecciones relacionadas con la tipografía, los esquemas de colores y la disposición de elementos. Dado que estamos partiendo de una interfaz ya existente, se explicarán en este apartado, las decisiones tomadas al modificar los elementos.

Debido a nuestra elección de utilizar una interfaz preexistente llamada FaztWeb [17], se definirá el wireframe como la interfaz original y se explicarán los cambios que hemos implementado en ella. Este enfoque se caracteriza por su simplicidad y un diseño visual atractivo que evita distracciones de los elementos clave.

De manera que la interfaz final se ve de la siguiente manera:

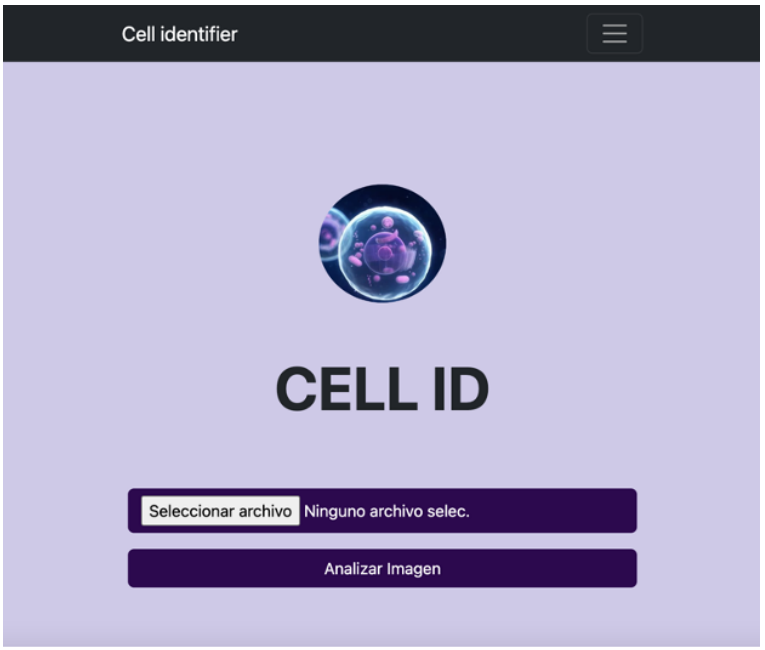

Figura 24: Vista Home

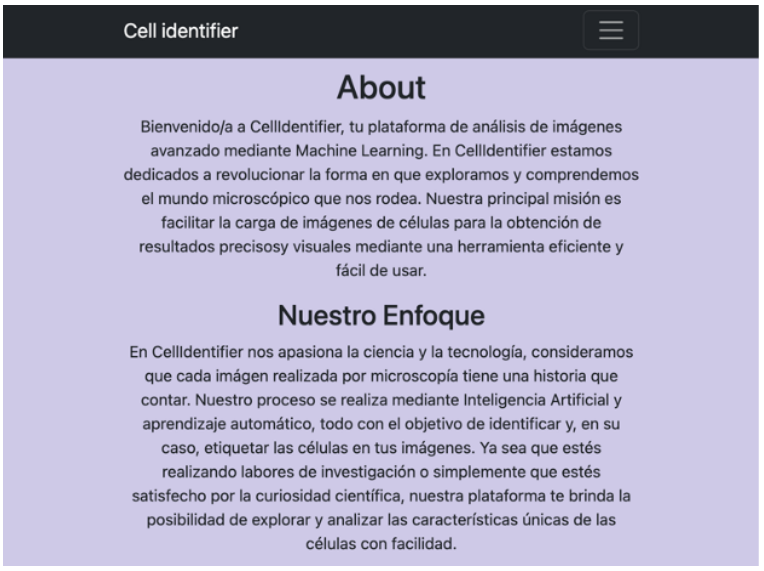

Figura 25: Vista About Us

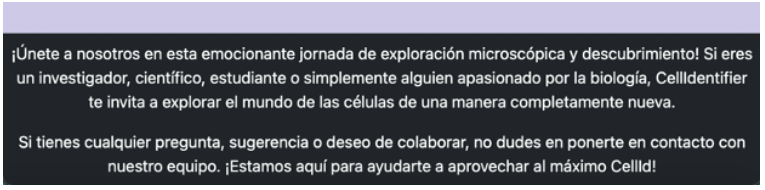

Figura 26: Footer de las vistas Home y Contact Us

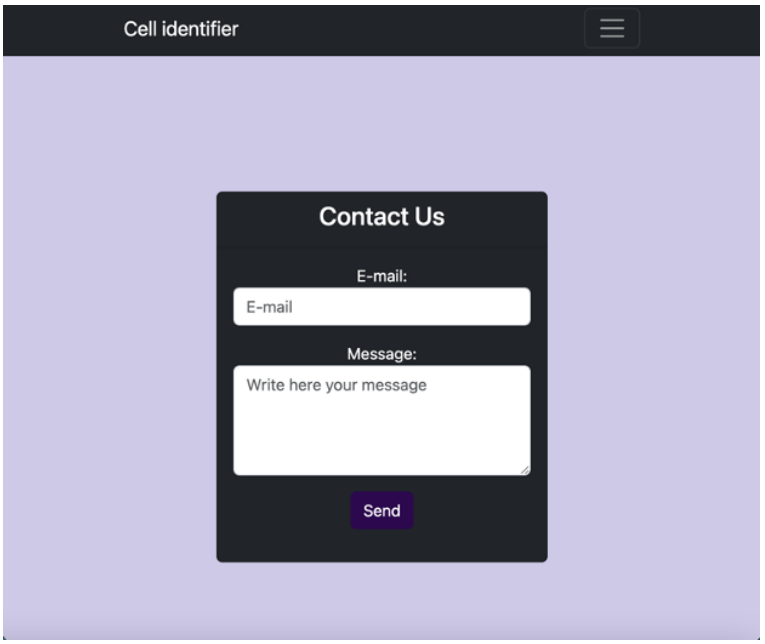

Figura 27: Vista Contact Us

Fase de prototipos y pruebas de usuario

Esta fase se compone de las siguientes tareas:

- Evaluación sobre la primera versión

En este punto, podemos identificar lo que hemos creado hasta ahora como nuestro prototipo final. Aunque el término "prototipo" generalmente se refiere a una versión no funcional que simula un producto digital terminado, en este caso, hemos trabajado con los usuarios para asegurarnos de que la interfaz sea fácil de usar e intuitiva. Por lo tanto, consideramos esta versión como nuestra versión final, lista para su implementación y uso.

Evaluación sobre la primera versión

Con el propósito de evaluar esta inicial iteración del prototipo, se llevó a cabo una prueba con un usuario del grupo secundario.

Dado que este grupo carece de conocimientos informáticos, su capacidad para utilizar la aplicación se considera un indicador sólido de su usabilidad. La prueba se realizó en persona, en un ordenador, y sin restricciones de tiempo. El usuario fue capaz de usar la aplicación sin dificultades, lo que lleva a la conclusión de que no se requieren más modificaciones en este momento.

8. Conclusiones y líneas de trabajo futuras

En esta sección, se expondrán las conclusiones derivadas del proceso de desarrollo del sistema implementado. Asimismo, se explorarán las posibles direcciones futuras que podrían aportar al sistema funcionalidades adicionales y una mayor solidez.

Conclusiones

Como se ha mencionado anteriormente, el Machine Learning o aprendizaje automático se considera una disciplina ampliamente aplicada en diversos aspectos tanto de nuestra vida cotidiana como en campos como el de la investigación.

La oportunidad de participar en un proyecto relacionado con este área ha sido enormemente gratificante y enriquecedor, debido a que no solo se ha profundizado en un tema interesante sino que en su realización se ha encontrado la solución a problemas dentro del campo científico. Además de aprovechar esta situación para evolucionar como persona y como futura profesional.

Inicialmente, se presentó como todo un desafío, ya que no solo implicaba el desarrollo de un modelo de aprendizaje automático que solucionara un problema actual sino que también el desarrollo de una interfaz que pudiera aplicar el modelo desarrollado. A pesar de esto, a medida que avanzaba en el proceso y, tras un considerable esfuerzo, se logró una familiarización con este ámbito de la informática.

Este proyecto ha brindado la oportunidad de aplicar todos los conocimientos adquiridos durante la carrera. Estos conocimientos han sido fundamentales para identificar necesidades actuales entre usuarios reales en un entorno laboral genuino, como es el caso del Centro de Investigación del Cáncer. Como resultado, hemos sido capaces de desarrollar soluciones adecuadas a estas necesidades.

Poder generar una solución funcional ha sido realmente gratificante. A través de todas las etapas y sobre todo a través del proceso de creación del modelo, se proporcionó una comprensión más profunda sobre el funcionamiento del lenguaje Python y la familiarización con lenguajes actuales como JavaScript durante todo el proceso de desarrollo de la interfaz. A partir de este punto, todo el proceso se volvió más fluido y accesible.

La búsqueda de recursos teóricos contribuyó a expandir los conocimientos en un campo tan importante como el Machine Learning. Se ha tomado como premisa fundamental el poseer un entendimiento sólido antes de poder utilizar o transmitir ese conocimiento.

De todo el proceso la parte más interesante ha sido profundizar en el Machine Learning o aprendizaje automático. Esta fase del proyecto ha abierto la puerta a la exploración de una amplia variedad de situaciones en las que esta disciplina puede ser aplicada, revelando su versatilidad en contextos que previamente se desconocía. Antes de esta experiencia, se tenía la impresión de que el Machine Learning era una disciplina sumamente compleja de comprender, sin embargo, el proceso ha demostrado que su entendimiento es alcanzable y sus aplicaciones son igualmente variadas y emocionantes.

Finalmente, es relevante mencionar la evolución personal que ha acompañado este trabajo. La ejecución de este proyecto ha posibilitado la aplicación de los conocimientos adquiridos a lo largo de cuatro años de formación universitaria. Entre las habilidades más destacadas y valiosas que se han empleado se encuentran las vinculadas a la Ingeniería del Software, la Interacción Persona-Ordenador y la Administración de Sistemas. No obstante, todo el conjunto de conocimientos obtenidos desempeñó un papel fundamental, no solo en la fase de implementación, sino también en la búsqueda de soluciones para superar desafíos imprevistos.

El aporte de este proyecto se refleja en la oportunidad de adentrarse en un lenguaje de programación tan relevante en la actualidad como Python, así como en la expansión de los conocimientos en el ámbito del desarrollo *front-end*, abarcando lenguajes como HTML, CSS y JavaScript.

Este proyecto ha brindado la oportunidad de participar en un contexto que simula un entorno laboral real, lo que ha proporcionado una sensación de preparación más sólida para enfrentar los desafíos del mundo profesional.

Líneas de trabajo futuras

Como evidencia a lo largo de todo el proyecto, la aplicación creada es una herramienta especializada en el ámbito científico, importante al agilizar los tiempos en este dominio. Por este motivo, es crucial no limitarse en este punto, sino más bien considerar la expansión de sus capacidades en el futuro.

A lo largo del desarrollo, han surgido áreas de enfoque para posibles mejoras en el futuro:

- Enfocar en la mejora de la precisión del modelo para lograr predicciones más certeras, culminando en resultados de mayor exactitud.
- Ampliar la funcionalidad del modelo para que, además de discernir "célula" de "no célula", pueda generar máscaras individuales para cada célula, permitiendo analizar sus atributos de forma separada.
- Potenciar la experiencia del usuario mediante mejoras en la interfaz, como la capacidad de cargar múltiples imágenes simultáneamente y visualizar los resultados correspondientes.
- Refinar la presentación de los resultados al permitir la selección de atributos individuales para cada célula dentro de una imagen, características como la intensidad y el brillo.
- Integrar plenamente el formulario de contacto para garantizar su funcionalidad efectiva.
- Desplegar la aplicación en línea para su acceso sin necesidad de configuración local.
- Incluir un versionado de los modelos entrenados.
- Facilitar a los usuarios la opción de exportar los resultados en formato Excel, otorgando una mayor utilidad a los datos obtenidos, incluyendo datos como la precisión del modelo utilizado permitiendo así el rastreo a lo largo del tiempo.

Referencias

[1] BBVA. (2019, 8 de noviembre). Machine Learning: ¿qué es y cómo funciona? Reference: https://www.bbva.com/es/innovacion/machine-learning-que-es-y-como-funciona/

[2] Google Cloud. (s.f.). ¿Qué es la inteligencia artificial o IA? Reference: https://cloud.google.com/learn/what-is-artificial-intelligence?hl=es-419

[3] Rodríguez, Txema (27 de enero de 2017). «Machine Learning y Deep Learning: cómo entender las claves del presente y futuro de la inteligencia artificial». Reference: https://www.xataka.com/robotica-e-ia/machine-learning-y-deep-learning-como-entender-las-claves-del-presente-y-futuro-de-la-inteligencia-artificial

[4] Rodríguez, V. (2018, 30 de octubre). Conceptos básicos sobre redes neuronales. Vincent Blog. Reference: https://vincentblog.xyz/posts/conceptos-basicos-sobre-redes-neuronales

[5] Guía de realización y documentación Proyecto Fin de Carrera en la Ingeniería Técnica Informática", Diaweb, Universidad de Salamanca.

[6] Algotive. (2022, 9 de julio). Historia de la inteligencia artificial, el machine learning y el deep learning.Reference: https://www.algotive.ai/es-mx/blog/historia-de-la-inteligencia-artificial-el-machine-learning-y-el-deep-learning

[7] Redacción APD. (2019, 4 de abril). ¿Cuáles son los tipos de algoritmos del machine learning? Reference: https://www.apd.es/algoritmos-del-machine-learning/

[8] Arrabales, R. (2016, 29 de marzo). Deep Learning: qué es y por qué va a ser una tecnología clave en el futuro de la inteligencia artificial. Xataka. Reference: https://www.xataka.com/robotica-e-ia/deep-learning-que-es-y-por-que-va-a-ser-una-tecnologia-clave-en-el-futuro-de-la-inteligencia-artificial

[9] Martins, J. (10 de octubre de 2022). ¿Qué es la metodología Kanban y cómo funciona? Asana. Reference: https://asana.com/es/resources/what-is-kanban

[10] Matich, D. J. (marzo de 2001). Redes Neuronales: Conceptos Básicos y Aplicaciones. Reference: https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf

[11] Redacción KeepCoding. (11 de enero de 2023). ¿Qué es la loss function? Reference: https://keepcoding.io/blog/que-es-la-loss-function-y-como-funciona/

[12] Unir Revista. (17 de febrero de 2023). ¿Qué es el algoritmo backpropagation para el entrenamiento de redes neuronales? Reference: https://www.unir.net/ingenieria/revista/backpropagation/#:~:text=Backpropagation%20es%20un%20algoritmo%20que,las%20redes%20de%20neuronas%20biol%C3%B3gicas.

[13] Redacción KeepCoding. (13 de enero de 2023). Tipos de capas de red neuronal convolucional. Reference: https://keepcoding.io/blog/tipos-capas-red-neuronal-convolucional/#Capa_convolucional.

[14] JACantoral. (2022). DL_Fundamentals. Repositorio de GitHub. Reference: https://github.com/JACantoral/DL_fundamentals/blob/main/Fundamentals_DL_UNET_4_video_v2.ipynb

[15] Justinmind, "How to design user scenarios: best practices and examples". Reference: https://www.justinmind.com/blog/how-to-design-user-scenarios/.

[16] FaztWeb. (2022). Our First Nodejs/Express Website. Repositorio de GitHub. Reference: https://github.com/FaztWeb/first-node-express-app.