UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Identificación de Parkinson por visión artificial

Presentado por Catalin Andrei Cacuci
en Universidad de Burgos — 20 de mayo de 2023
Tutora: Alicia Olivares Gil
Cotutor: Álvar Arnaiz González

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

Dña. Alicia Olivares Gil y D. Álvar Arnaiz González, profesores del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Catalin Andrei Cacuci, con DNI X7451927L, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Identificación de Parkinson por visión artificial.

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

En Burgos, a 20 de mayo de 2023

Vº. Bº. de la tutora:	Vº. Bº. del cotutor:
Dña. Alicia Olivares Gil	D. Álvar Arnaiz González

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

La enfermedad de Parkinson es una enfermedad neurodegenerativa que afecta a millones de personas en todo el mundo. Se caracteriza por la degradación de las células nerviosas que controlan el movimiento, lo que puede provocar temblores, rigidez muscular y pérdida del control postural [8]. Aunque no existe una cura para la enfermedad de Parkinson, la detección temprana y el tratamiento adecuado pueden mejorar significativamente la calidad de vida de los pacientes.

En los últimos años, la inteligencia artificial se ha convertido en una herramienta de gran utilidad para el diagnóstico y tratamiento de enfermedades, incluida la enfermedad de Parkinson. En este proyecto, se ha desarrollado un sistema basado en inteligencia artificial para crear modelos de aprendizaje automático capaces de detectar si un individuo padece la enfermedad de Parkinson a partir de un vídeo de la persona realizando un gesto característico con la mano, además de cierta información adicional sobre el individuo.

Interactuar con los modelos creados de forma directa puede resultar complicado si no se dispone de ciertos conocimientos previos sobre programación e inteligencia artificial. Con el objetivo de facilitar esta tarea para la persona media se ha desarrollado una aplicación web que simplifica el uso de los modelos.

Los resultados obtenidos indican que los modelos creados son capaces de detectar la enfermedad de Parkinson con bastante precisión, lo que sugiere que podrían ser de utilidad para la detección de la enfermedad. Además, el enfoque basado en vídeo es no invasivo y fácil de usar, lo que lo hace potencialmente útil en entornos clínicos y de telemedicina.

Por último, me gustaría dar las gracias a todos aquellos que han dedicado

su tiempo y esfuerzo a la obtención de los vídeos utilizados para entrenar los modelos, incluyendo la Asociación de Parkinson de Burgos, el personal del Hospital Universitario de Burgos y el equipo de la Dra. Esther Cubo.

2. Objetivos del proyecto

El objetivo principal de este proyecto es la creación de un sistema basado en la inteligencia artificial que permita la indentificación de la enfermedad de Parkinson de forma consistente utilizando como entrada un vídeo de un gesto específico (además de otra información), además este sistema debería ser utilizable por la persona media. Para alcanzar este objetivo se plantean los siguientes objetivos específicos:

1. Revisar los trabajos previos relacionados para obtener unos conocimientos iniciales sobre el tema e identificar qué métodos funcionan mejor y peor.

2. Diseñar e implementar una secuencia de pasos para extraer características relevantes de los vídeos para su uso en algoritmos de aprendizaje automático.

3. Buscar un modelo que se ajuste considerablemente bien a los datos disponibles mediante alguna de las técnicas existentes en el campo de la inteligencia artificial para este propósito.

4. Crear una aplicación web para interactuar de forma sencilla con los modelos y obtener predicciones. Debido al público de una aplicación de este tipo se debe tener muy en cuenta la accesibilidad a la hora de diseñarla.

5. Añadir un panel de administración a la aplicación para actulizar el modelo utilizado para realizar las predicciones.

6. Documentar el trabajo realizado.

3. Conceptos teóricos

Este capítulo define algunos de los conceptos teóricos que se mencionan en este documento.

3.1. Minería de Datos

La minería de datos es un área de la inteligencia artificial que consiste en el diseño de algoritmos y modelos que permitan a los ordenadores aprender la regla general que define un conjunto de datos sin ser explícitamente programados para ello.

La minería de datos incluye, además de algoritmos de aprendizaje, todas las metodologías y técnicas utilizadas para el tratamiento y filtrado del conjunto de datos disponible.

Preprocesado

La información con la que se entrene un modelo de aprendizaje automático determina en gran medida el rendimiento que podrá alcanzar, debido a esto es muy frecuente realizar un paso previo a la creación del modelo, denominado *preprocesado* [9].

El objetivo del preprocesado es transformar la entrada de datos iniciales con el fin de permitir y facilitar que un modelo se adapte a los mismos.

El preprocesado también tiene una gran influencia sobre el tiempo de computación necesario para entrenar un modelo y sobre la complejidad que necesitará para adaptarse a los datos.

El preprocesado suele componerse de algunos de los siguientes pasos.

Extracción de características

La extracción de características es el proceso de identificar, seleccionar y transformar atributos relevantes de los datos de entrada para su uso en un modelo. Por ejemplo, en este proyecto, se han extraido características como la velocidad o amplitud de movimiento a partir de un vídeo.

Existen diversas técnicas para la extracción de características, incluyendo las selección manual de características, o técnicas automatizadas como la reducción de dimensionalidad, la extracción de características basada en redes neuronales [2], entre otras. La elección de la técnica de extracción de características depende del conjunto de datos, del problema específico de aprendizaje automático que se está abordando y del tipo de modelo de aprendizaje automático que se está utilizando.

Selección de características

Los datos de entrada pueden ser muy complejos y estar compuestos por una gran cantidad de información redundante o no relevante para el modelo. La selección de características se utiliza para identificar las características más relevantes y representativas de los datos, que pueden ser utilizadas para entrenar modelos de aprendizaje automático con mayor eficacia.

En la selección de características, se pueden utilizar técnicas como el análisis de componentes principales (PCA) [5], el análisis discriminante lineal (LDA) [14] o pruebas de significancia, entre otras.

Normalización

Existen modelos muy sensibles a que existan diferencias en la escala de los distintos atributos, como, por ejemplo, *k-nearest neighbors*, por lo que es muy habitual que en la fase de preprocesado se realize una normalización de los datos, es decir, transformarlos de tal forma que utilicen la misma escala, en general se suelen tomar valores en los intervalos $[0, 1]$ o $[-1, 1]$.

Aunque lo más habitual es que la normalización se haga sin distorsionar las diferencias existentes entre los valores previos, existen situaciones en las que puede ser ventajoso utilizar un método de normalización que sí altere estas diferencias, por ejemplo, la normalización por cuantiles [10], en la que se modifican los valores para que sigan una distribución normal, lo que consigue que, si existen valores muy lejanos a los valores más comunes (*outliers*), estos se acerquen al resto.

Aprendizaje supervisado

Dentro del aprendizaje automático existen tres ramas o variantes dependiendo del conjunto de datos del que se disponga, ya que estos datos son los que determinan las técnicas, algoritmos y metodologías que se podrán utilizar, estas variantes son el aprendizaje no supervisado, en el que los datos no están etiquetados, es decir, el atributo que se desea predecir es desconocido (el ejemplo más característico es el *clustering*), aprendizaje semisupervisado, en el que solo una parte de los datos están etiquetados, y el aprendizaje supervisado, en el que el conjunto de datos está completamente etiquetado, este último es el tipo de aprendizaje utilizado realizado en el proyecto y del que trata este apartado.

El enfoque del aprendizaje supervisado consiste en la utilización de dos conjuntos de datos, uno de entrenamiento (con datos etiquetados) y otro de test (con datos no etiquetados), a continuación se utiliza un algoritmo de aprendizaje para crear un modelo que «aprenda» de las instancias del conjunto de entrenamiento una regla o procedimiento que le permita identificar los datos del conjunto de test con la mayor precisión posible [4].

Clasificación

En el ámbito del aprendizaje supervisado, clasificar instancias en categorías predeterminadas es uno de los principales problemas a resolver.

En general, la resolución de un problema de clasificación se caracteriza por delimitar qué zonas del espacio que contiene todas las instancias posibles pertenecen a cada categoría, es decir, definir fronteras a partir de las cuales las instancias pasan de una categoría a otra.

Existe una gran cantidad de métodos y algoritmos de clasificación que se pueden utilizar para determinar estas fronteras, dependiendo del problema específico que se intente resolver unos algoritmos se comportarán mejor que otros.

Sobreajuste

Normalmente solo se dispone de una pequeña muestra para entrenar un modelo que encuentre la regla general que define la población y se adapte lo mejor posible a esta.

El problema de esto es que, si se utilizan todos los datos disponibles durante el entrenamiento, no existe ninguna forma para verificar el comportamiento del modelo cuando se encuentre con instancias que no ha visto

antes. Esto hace posible que el modelo "memorice" los datos con los que se ha entrenado pero no generalice bien al encontrarse con datos nuevos. Esto se conoce como sobreajuste.

Una solución muy común al sobreajuste es separar los datos disponibles en dos grupos, un conjunto de entrenamiento y un conjunto de test o prueba. Durante el entrenamiento el modelo solo tendrá acceso al conjunto de datos de entrenamiento, mientras que el conjunto de test es utilizado para determinar el rendimiento del modelo sobre datos nuevos mediante una de las métricas de evaluación que se verán a continuación.

Evaluación del modelo

Para determinar si un modelo es mejor que otro es necesario definir alguna métrica que asigne un valor numérico al rendimiento de cada modelo.

La métrica más intuitiva es lo que se conoce como precisión (*accuracy*), que es simplemente la proporción de predicciones (clasificaciones) acertadas del total de predicciones realizadas.

Al igual que ocurren con el algoritmo de clasificación empleado, la métrica que se use debe ser seleccionada en función del problema en cuestión. Una situación muy común en la que utilizar la precisión no es lo más ideal es cuando se trabaja con conjuntos de datos desbalanceados, es decir, cuando existen más instancias de una clase que de otra.

Lo anterior es muy común en el ámbito médico cuando se intenta crear un modelo que determine si un paciente padece una enfermedad concreta o no. En estas situaciones se suele tener un grupo de control muy grande que no padece la enfermedad y un grupo relativamente pequeño que sí la padece. Si, por ejemplo, las proporciones de clases son 95 % y 5 % respectivamente, un modelo que siempre prediga que el paciente no padece la enfermedad en cuestión obtendrá una precisión del 95 %, lo que podría dar la falsa impresión de que se ajusta bien a los datos, cuando, en realidad, es un modelo completamente inútil.

Algunas alternativas a la precisión incluyen:

- **Sensibilidad**: Para una clase concreta, indica la capacidad del modelo de clasificar correctamente instancias de esa clase. En el caso médico esta métrica podría ser interesante para encontrar aquellos pacientes que sí padecen la enfermedad, aunque se clasifiquen mal algunos que no la padecen.

- **Especificidad**: Para una clase concreta, indica la capacidad del modelo para clasificar correctamente instancias que no pertenecen esa clase.
- **Valor-F1**: Es una medida que combina la sensibilidad y la especificidad mediante una media armónica.

Fuga de información

En el campo del aprendizaje automático las fugas de información se producen cuando, de alguna forma, se utiliza información perteneciente al conjunto de datos de test para entrenar un modelo, esto no significa únicamente utilizar instancias de test durante el entrenamiento, las fugas de información se pueden producir de forma mucho más sutil y ser difíciles de detectar. Por ejemplo, si se utilizase el conjunto de test para seleccionar las características durante el preprocesado se estaría produciendo una fuga de información.

Las fugas de información no son algo que debería ser siempre evitado, existen situaciones en las que son necesarias dependiendo del caso concreto. Pero sí es importante que sean detectadas y tenidas en cuenta a la hora de analizar los resultados obtenidos, ya que, en caso contrario, se podría llegar a conclusiones equívocas (demasiado optimistas por ejemplo).

Validación cruzada

En un apartado anterior se ha visto que para evitar el sobreajuste se puede dividir la muestra de datos disponible en un conjunto de datos de entrenamiento y uno de test. Al realizar esta división es posible que, en especial al trabajar con muestras pequeñas, se realice una división que no sea útil para validar el modelo. Por ejemplo, si por casualidad se seleccionase un conjunto de test "fácil de predecir" con el conjunto de entrenamiento dado, se medirá un rendimiento mayor al real.

La solución a esto es la validación cruzada, que consiste en utilizar múltiples pares de conjuntos de validación y test, de tal forma que se entrenen y validen tantos modelos como cantidad de estos pares y cada instancia sea utilizada para la validación al menos una vez, el resultado final de la validación es la media de la métrica elegida de cada iteración.

Existen varios métodos para determinar el número de iteraciones a realizar y el tamaño de los grupos de entrenamiento y test en cada iteración, por ejemplo:

- ***Leave-One-Out***: Para una muestra de tamaño N, consiste en realizar N iteraciones de entrenamiento y validación utilizando cada vez una única instancia para el conjunto de test. Es un método que se acerca bastante al rendimiento que se obtendría si se utilizase la muestra entera como conjunto de entrenamiento, pero tiene el problema de que se van a tener que realizar N iteraciones, lo que implica un tiempo de computación necesario muy grande.

- ***K-fold***: Consite en dividir la muestra en conjuntos de entrenamiento y test de proporciones $(K-1)/K$ y $1/K$ respectivamente, realizando un total de K iteraciones. Además se puede realizar varias repeticiones de este tipo de validación cruzada para reducir la variabilidad del error obtenido.

Optimización de hiperparámetros

Los algoritmos de clasificación suelen tener parámetros que determinan la forma en la que se van a ajustar a los datos, por ejemplo, la tasa de aprendizaje o el número de iteraciones máximas, estos valores se denominan hiperparámetros.

El proceso de búsqueda dentro del espacio que contiene todas las combinaciones posibles de hiperparámetros para un algoritmo concreto se denomina optimización de hiperparámetros.

En general, la optimización de hiperparámetros consiste en probar diferentes combinaciones de valores y realizar el proceso de entrenamiento y test mediante validación cruzada con una métrica apropiado hasta dar con la combinación más adecuada. Existen varias formas para realizar esto, como, por ejemplo:

- ***Grid search***: Proceso mediante el cual se definen los valores posibles que puede tomar cada parámetro y se prueba cada combinación de estos valores hasta encontral la mejor. Es una búsqueda exhaustiva que puede dar lugar a una gran cantidad de combinaciones, lo que implica un tiempo de computación muy grande.

- ***Randomized search***: Proceso similar al anterior en el que se especifican los valores posibles para cada parámetro, pero en este caso se realiza la búsqueda de forma aleatoria de forma que se limita el tiempo de computación necesario. Se puede utilizar para obtener una vista general del espacio de combinaciones de parámetros para realizar

posteriormente una búsqueda más exhaustiva en un subespacio más pequeño.

- ***Técnicas de optimización***: Existen diversos métodos que intentan determinar la forma que tiene el rendimiento del modelo con respecto al espacio de combinaciones de parámetros posibles utilizando iteraciones anteriores para buscar los valores óptimos sin tener que probar todas las combinaciones como ocurre con *grid search*. Un ejemplo de esto es la Optimización Bayesiana [13] en la que se intenta predecir la forma que toma la métrica de evaluación en función de los hiperparámetros elegidos, y así realizar una búsqueda más efectiva.

4. Técnicas y herraminetas

En el área de la inteligencia artificial, y en general en la informática, la conocida expresión «Pararse a hombros de gigantes» tiene mucho sentido, ya que es muy difícil, y contraproducente, crear algo por completo desde cero, siempre se utilizan herramientas, librerías o frameworks (abstracciones en general) creados por otras personas para acelerar el trabajo y permitir que al programador centrarse en la lógica de negocio que quiere implementar.

Este capítulo enumera las técnicas y herramientas de otras personas que se han utilizado durante la realización de este proyecto.

4.1. Técnicas

Esta sección detalla las distintas técnicas que fueron utilizadas durante el desarrollo de este proyecto.

CRISP-DM

«Cross-industry standard process for data mining», conocido como (CRISP-DM), es un modelo estándar abierto del proceso que describe los enfoques comunes que utilizan los expertos en minería de datos [7]. Este estándar surgió a finales de los 90 a partir de la inexistencia y necesidad de un enfoque ampliamente aceptado para la minería de datos. Para conseguir esto, CRIPT-DM es agnóstico a la industria y herramientas que se utilicen, lo que permite que sea utilizado en un amplio abanico de situaciones, lo que facilitó su adopción.

CRISP-DM divide el ciclo de vida de un proyecto de minería de datos en las siguientes 6 fases.

1. **Entendimiento del negocio**: La fase más importante de cualquier proyecto de minería de datos, consiste en obtener un entendimiento sobre los objetivos del proyecto desde el punto de vista del negocio y redefinir estos objetivos como un problema de minería de datos diseñado para cumplir con estos objetivos.

2. **Entendimiento de los datos**: La fase de entendimiento de datos comienza con una recolección inicial de datos. A continuación, se procede a analizar los datos para obtener cierta familiaridad con los mismos, identificar problemas y realizar hipótesis iniciales sobre la información oculta que pueda existir.

3. **Preparación de los datos**: La fase de preparación de los datos cubre todas las actividades realizadas para la construcción del conjunto de datos final a partir de la información sin procesar que se va a utilizar para entrenar los modelos de aprendizaje. Esta fase incluye la selección, limpieza, integración y formateado de los datos.

4. **Modelado**: En esta fase se seleccionan varios algoritmos de aprendizaje, cuyos parámetros deben ser calibrados para ajustarse al conjunto de datos de forma óptima. Diferentes algoritmos imponen diferentes requisitos sobre el conjunto de datos sobre el que puede trabajar, por lo que es posible que se deba volver a la fase anterior para reajustar el conjunto de datos final. Esta fase incluye la selección de los algoritmos a utilizar, la creación de los modelos y la evaluación del rendimiento de los mismos.

5. **Evaluación**: Antes del despliegue del modelo final es de gran importancia reevaluar si el modelo cumple realmente con los objetivos de negocio establecidos en la primera fase. Aquí se deberá determinar cómo se van a utilizar los resultados de la minería de datos. Los procesos clave de esta fase son la evaluación de los resultados, la revisión del proceso utilizado y la detemrinación de los pasos siguientes.

Las fases anteriores tienen una naturaleza cíclica, los conocimientos obtenidos en cada ciclo de CRISP-DM se utilizan para replantear los objetivos de negocio y repetir el proceso con los nuevos conocimientos.

4.2. Herramientas de desarrollo

Esta sección detalla las distintas herramientas, librerías y frameworks que fueron utilizadas durante el desarrollo de este proyecto.

Fedora Linux

Aunque al iniciar el proyecto se utilizó Windows 10 como sistema operativo principal, la mayor parte del desarrollo ha sido realizado en la distribución de Linux Fedora, en concreto versiones 37 y 38.

Fedora se caracteriza por que sus repositorios se actualizan regularmente para incluir las últimas versiones estables de la mayoría de programas, al contrario de lo que ocurriría en una distribución como Debian estable. Pese a esto, Fedora es considerada como una distribución bastante estable y, duarante el desarrollo de este proyecto, no se han encontrado problemas fuera de lo común.

Han sido instalados de forma nativa Python 3.9 para la fase de investigación, TEX Live para la compilación de esta documentación y Docker para el desarrollo de la aplicación web.

Python

Python es un lenguaje de programación diseñado con el objetivo de ser fácil de escribir, para lograr esto existe una gran cantidad de abstracciones y detalles de los que se encarga, mientras que en otros lenguajes debería encargarse el programador. Esto tiene muchas ventajas, pero también algunas desventajas, como, por ejemplo, que se pierde el control sobre la ejecución del programa que podría dar un lenguaje como C o C++, lo que significa que, en general, un mismo programa escrito completamente en Python va a ser más lento que su contrapartida escrita en un lenguaje de programación de más bajo nivel en el que se pueden especificar detalles como, por ejemplo, la gestión de memoria.

Pero Python tiene otra cualidad muy importante, y es su facilidad para interactuar con código escrito en C o C++ mediante lo que se conoce como *bindings* o *wrappers*. Esto permite al programador tener mayor control sobre las partes más críticas de un programa, escribiéndolas en C por ejemplo, mientras que las partes de menor importancia se pueden escribir rápidamente en Python.

Un ejemplo perfecto de lo anterior es la librería NumPy, utilizada para la gestión de matrices, entre muchas otras cosas. Esta librería implementa su funcionalidad principal en C y expone una interfaz en Python para facilitar su uso.

Este patrón se repite una y otra vez en el campo del aprendizaje automático, donde los algoritmos de aprendizaje se implementan a bajo nivel

para que su ejecución sea lo más rápida posible, mientras que se expone una interfaz a alto nivel en Python para interactuar con los mismos. Ejemplos de esto son las librerías PyTorch y TensorFlow.

Jupyter Notebook

Jupyter Notebook es una herramienta que permite crear un archivo con extensión `.ipynb`, en el que el código Python se divide en "celdas" que se pueden ejecutar individualmente, pero manteniendo y compartiendo las variables establecidas de forma global entre todas estas celdas.

Esta herramienta fue de especial utilidad durante la fase de investigación, ya que permite observar detenidamente los cambios que se producen entre celdas y, así, encontrar problemas sin tener que ejecutar el código completo desde el principio cada vez que se realice un cambio, como ocurriría con un *debugger*.

Además de lo anterior Jupyter Notebook permite documentar de forma clara el código mediante otro tipo de celda, en el que se puede escribir código en Markdown, que es, en cierto modo, una alternativa menos verbosa a HTML.

Docker

Docker permite la creación y gestión de contenedores, un contenedor se puede ver como un sistema operativo dentro del anfitrión, pero sin llegar a ser una máquina virtual, debido a que se utiliza el *kernel* de Linux del sistema anfitrión y los archivos del contenedor existen en el sistema de archivos del anfitrión. Esto hace que la diferencia de rendimiento entre utilizar un contenedor y ejecutar programas de forma nativa sea mínima.

Un contenedor de Docker se basa en lo que se conoce como una imagen, que es un sistema de archivos preconfigurado para el propósito del contenedor. Se puede encontrar una gran cantidad de imágenes en Docker Hub, como, por ejemplo, postgres, node o python que ya incluyen los ejecutables para utilizar cada herramienta respectiva. Pero, es posible que se necesite mayor control sobre la configuración de la imagen sobre la que se va a construir el contenedor, en estos casos se pueden utilizar lo que se conoce como *Dockerfile* para crear una imagen tomando como base otra imagen ya existente.

Un *Dockerfile* es un archivo que determina la imagen de base a utilizar (cualquier imagen de Docker Hub u otros sitios) y los comandos a ejecutar

para crear la imagen que se desea obtener. Si se desease empezar de cero se puede utilizar una imagen de Ubuntu o Debian como base.

Además de *Dockerfile* existe otro tipo de archivo denominado *docker-compose.yml* utilizado para coordinar sistemas compuestos por multiples contenedores que interactúan entre sí. La aplicación web de este proyecto utiliza *docker-compose.yml* para determinar las interacciones entre los cuatro contenedores creados (web, API, proxy inverso y base de datos).

PyCharm

PyCharm es un Entorno de Desarrollo Integrado (IDE) creado y mantenido por JetBrains, una compañía que se dedica a crear software y cuyos productos más representativos son multitud de IDEs para diferentes lenguajes de programación (IntelliJ (Java), CLion (C y C++), GoLand (Go), etc.).

El objetivo de PyCharm es facilitar el desarrollo de aplicaciones que utilizan Python y da soporte amplio tanto para los habituales ficheros `.py` como para `.ipynb` (Jupyter Notebook).

En este proyecto PyCharm fue utilizado para esarrollar la librería PADDEL durante la fase de investigación, que contiene el código de Python común a los Notebooks de Jupyter utilizados durante la fase de investigación.

Visual Studio Code

Visual Studio Code (VS Code) es un editor de texto orientado al desarrollo creado por Microsoft. Gracias a su gran extensibilidad es posible utilizarlo cómodamente en casi cualquier escenario.

Fue utilizado en la creación de esta documentación en conjunto con la extensión LaTeX Workshop que integra distintos compiladores de LaTeX (pdfTeX, XeTeX, LuaTeX, etc.) dentro del programa y añade otras funcionalidades que hacen trabajar LaTeX mucho más conveniente.

Además, gracias a la extensión Dev Containers, VS Code puede ser lanzado desde contenedores de Docker, permitiendo interactuar con las utilidades del contenedor. Gracias a esto VS Code ha sido la herramienta utilizada para desarrollar por completo la aplicación web.

Git

Actualmente Git es el sistema de control de versiones más extendido en proyectos de código abierto [6], en este proyecto se ha usado para gestionar y guardar de forma segura los cambios que se han ido realizado con el tiempo en la plataforma GitHub.

Aunque existen varias herramientas con interfaces gráficas que envuelven el funcionamiento de Git (GitKraken, GitHub Desktop, etc.) para este proyecto se utilizó directamente el comando `git` desde una terminal de Linux.

OpenCV

OpenCV es una librería de visión artificial que contine varias utilidades para interactuar con archivos de vídeo e imágenes de forma similar a como se trabajaría en un programa como MATLAB. OpenCV está escrito en C y C++ pero tiene *bindings* que permiten interactuar con sus interfaces desde Python.

En este proyecto se usa para leer y decodificar los archivos de vídeo utilizados durante la fase de investigación.

MediaPipe

MediaPipe es una librería de aprendizaje automático mantenida por Google que da acceso a varios modelos preentrenados que se centran en la detección de características corporales a partir de flujos de imágenes, en este caso se utilizó MediaPipe Hands[15], cuyo objetivo es extraer varios puntos que definen un esqueleto ligeramente simplificado de la mano humana.

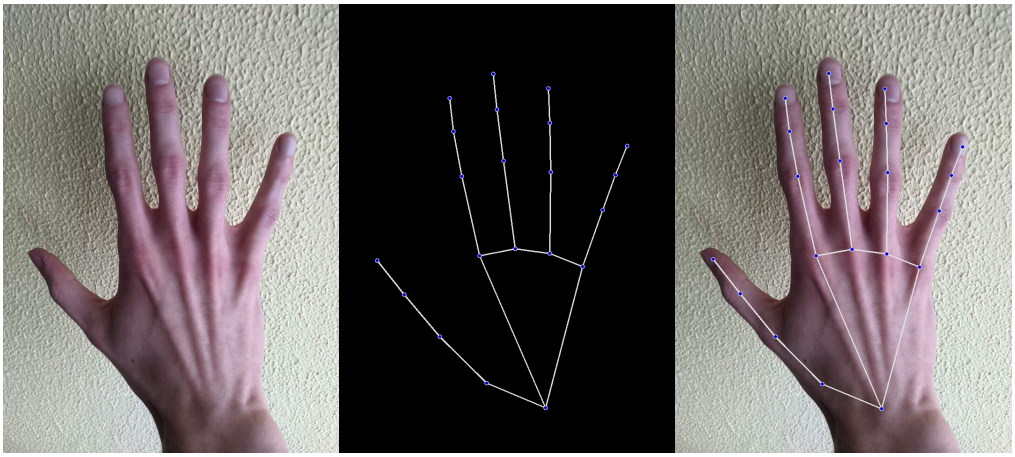

Figura 4.1: MediaPipe Hands

Como se puede ver el resultado es bastante bueno, además, al utilizar vídeos MediaPipe utiliza la información de los fotogramas anteriores para determinar de forma más exacta la pose en los fotogramas posteriores. Esto es muy útil para mejorar la precisión de las poses obtenidas, pero tiene la pega de que, al necesitar información de pasos previos, el proceso de extracción de poses no se puede paralelizar.

FastAPI

FastAPI es un *framework* de Python utilizado para crear APIs de tipo REST que se caracteriza por permitir al programador centrar sus esfuerzos en el código relacionado con la lógica de negocio al realizar la mayoría de acciones comunes entre diferentes APIs por defecto (como la creación de documentación, validación de tipos, serialización y deserialización de peticiones, etc.).

En este proyecto se ha creado una API REST para permitir la interacción con el código de Python utilizado para crear, entrenar y utilizar los modelos desde cualquier dispositivo y lenguaje de programación (siempre y cuando permita realizar peticiones HTTP). Con esto se consigue que en el futuro se puedan crear nuevas formas para interactuar con los modelos sin necesidad de alterar el código ya existente. Se podría, por ejemplo, crear una aplicación móvil para facilitar la toma y subida de vídeos. Además estas nuevas aplicaciones podrían estar creadas por personas ajenas al proyecto que necesiten una implementación diferente para su caso de uso específico.

SvelteKit

Para el frontend de la aplicación, es decir, todo lo que va a ver un usuario de la web (HTML, CSS y JavaScript), se ha utilizado SvelteKit, que es un framework de JavaScript basado en Svelte, es similar a otros frameworks más populares como React o Angular, y se caracteriza por tener una sintaxis muy clara y simple además de reducir el tamaño de los archivos que se envían al navegador del cliente, haciendo que, en general, las páginas creadas con SvelteKit sean más rápidas que sus contrapartidas.

Typescript

JavaScript es un lenguaje de programación que no dispone de tipado estático, lo cual hace muy difícil detectar problemas fuera del tiempo de ejecución.

TypeScript es un lenguaje de programación desarrollado por Microsoft que utiliza una sintaxis de base muy similar a la de JavaScript, pero añadiendo tipado estático. Los navegadores no "entienden" TypeScript, por lo que éste debe ser compilado a JavaScript antes de ser enviado al usuario. Durante esta compilación se pueden detectar multitud de errores gracias al tipado.

Typesafe i18n

Typesafe i18n es una librería escrita en TypeScript que se puede utilizar para internacionalizar aplicaciones web, es decir, añadir soporte para más idiomas, de forma simple y mediante unas convenciones predefinidas.

Para ello se crea un archivo para cada idioma al que se de soporte en el que se definen las variables con las cadenas de texto utilizadas a través de la web en el idioma respectivo. Cada uno de estos archivos debe contener las mismas variables.

A continuación en los lugares donde se usen cadenas de texto se debe importar la variable `LL` que es un objeto con todas las traducciones que se puede usar en la página respectiva para establecer las cadenas de texto dependiendo del idioma.

Axios

Axios es una librería de JavaScript que envuelve la función nativa `fetch` para hacerla más amigable y fácil de usar. Se caracteriza por soportar tipos

de TypeScript y simplificar la realización de peticiones HTTP.

En este proyecto se ha utilizado esta librería para realizar peticiones REST a la API desde el navegador del usuario. Para ello se han creado multitud de funciones que reflejan los *endpoints* de la API y, así, permitir su uso desde el resto del sitio web.

Tailwind CSS

Tailwind CSS es una librería de clases de CSS, su funcionamiento es similar a Bootstrap en cuanto a que existen multitud de clases que realizan diferentes acciones, la diferencia está en que Tailwind CSS es mucho más modular que Bootstrap, es decir, las clases realizan cambios muy pequeños sobre los objetos a los que afectan. Esto hace que Tailwind sea más lento de escribir que su contrapartida, pero al mismo tiempo permite un nivel mucho mayor de configuración sobre el diseño de una página web.

Prettier

Prettier es un formateador de código, creado concretamente para dar un formato consistente a archivos de HTML, CSS, JavaScript, TypeScript, etc.

Caddy

En las fases iniciales de desarrollo se decidió que la API debería estar alojada en el subdominio `api.`, mientras el dominio principal debería llevar a la web (SvelteKit). Esto significa que van a existir dos servicios funcionando de forma paralela (Web y API), y que dependiendo del dominio al que se realize una petición, ésta debería ser redirigida a un servicio u otro.

Esta situación es uno de los casos de uso idóneos para un proxy inverso, que es un servicio que recibe peticiones HTTP, las altera si es necesario, y las redirige al servicio de destino (el cual se determina mediante ciertas reglas de redirección).

En un principio se optó por utilizar Nginx para implementar este proxy inverso con muy buenos resultados gracias a su fácil configuración, rendimiento y simplicidad. Pero surgieron problemas al añadir soporte para utilizar certificados SSL y realizar peticiones seguras mediante HTTPS, ya que era deseable una solución que gestionase de forma automática la petición y renovación de estos certificados y Nginx, debido a su simplicidad no dispone de esta funcionalidad.

Se probaron dos alternativas que solucionaban este problema, Traefik y Caddy, ambas escritas en GoLang. Traefik dispone de una gran cantidad de funcionalidades incluyendo gestión de certificados SSL, entre muchas otras, como un panel de control desde el que visualizar estadísticas sobre el nivel de uso del servicio.

La otra alternativa, Caddy, es mucho más simple, pero realiza la gestión de certificados de forma automática por defecto, e incluso permite el uso de certificados *self-signed* (firmados por uno mismo, en lugar de por una entidad certificadora). Esto es de gran utilidad durante el desarrollo para asegurar que el funcionamiento de la aplicación va a seguir siendo el mismo en el entorno final de producción (ya que hay ligeras diferencias dependiendo de si un navegador utiliza HTTP o HTTPS).

Es por todo lo anterior que, en última instancia, se decidió usar Caddy.

Formateado de código Python

Se han utilizado las siguientes herramientas para realizar el formateado del código escrito en Python de forma automática:

- **Autoflake**: herramienta utilizada para la eliminación de sentencias `import` y variables innecesarias.
- **Isort**: herramienta para la reordenación de sentencias `import`.
- **Black**: formateador de código caracterizado por un enfoque en convenciones, por esto permite una configuración muy limitada por parte del usuario.

Make

A través del proyecto hay comandos de cierta complejidad que se repiten con bastante frecuencia (*e.g.* formatear el código, lanzar contenedores, limpiar contenedores, etc.), para facilitar el uso de estos comandos se ha utilizado la utilidad GNU Make, que viene por defecto en la mayoría de sistemas operativos de tipo Linux y se puede instalar tanto en Windows como en MacOS.

El funcionamiento de Make consiste en la creación de un archivo denominado `Makefile` que contiene diferentes comandos y secuencias de comandos con las dependencias que existen entre los mismos.

Por ejemplo:

```
format:
    autoflake src
    isort src
    black src
```

Un fichero `Makefile` con el contenido anterior permite ejecutar el comando `make format` (siempre desde el mismo directorio en el que se encuentra `Makefile`) para ejecutar la secuencia de comandos denominada `format`, que, en este caso ejecutará varios comandos para realizar un formato del código.

TSFresh

TSFresh [1] es una librería de Python que se encarga del preprocesado y la extracción de características de series temporales.

Una serie temporal es una secuencia ordenada de valores con una componente temporal asociada. Estas secuencias por sí solas no son de mucha utilidad para entrenar algunos modelo, por lo que en la fase de preprocesado se debe realizar una extracción de características sobre las mismas y así obtener características numéricas que sí se pueden utilizar para entrenar un modelo.

TSFresh establece una forma estándar para extraer estas características, permitiendo extraer características propias además de las, aproximadamente, 700 características que extrae por defecto.

ChromeVox

Un aspecto que se tuvo muy en cuenta durante el desarrollo del sitio web es la accesibilidad, en especial para usuarios con visibilidad reducida que dependen de lectores de pantalla para decubrir el contenido de la web.

Para comprobar cómo interactuaría un lector de pantalla con la aplicación se ha utilizado la extensión de Google Chrome ChromeVox.

5. Aspectos relevantes del desarrollo del proyecto

6. Trabajos relacionados

Durante la última década se ha intentado utilizar la visión por computador para evaluar la Enfermedad de Parkinson en múltiples ocasiones con diferentes resultados. Este capítulo recopila de forma resumida algunos de los trabajos más relevantes.

6.1. A computer vision framework for finger-tapping evaluation

Este artículo [3] documenta el uso de visión por computador para determinar el nivel de severidad de la Enfermedad de Parkinson y distinguir entre individuos con esta enfermedad e individuos sin ella.

El método empleado se caracteriza por utilizar vídeos con la cara de la persona y ambas manos a los lados de la cabeza, apuntando las puntas de los dedos hacia la misma. Esto se utiliza para poder normalizar las distancias en base a características faciales.

El estudio se realizó sobre 13 pacientes con la Enfermedad de Parkinson, tomando 17 vídeos de cada paciente durante un día, además de un grupo de control de 6 individuos, tomando 2 vídeos al día durante una semana por cada uno. Aunque algunos de estos vídeos fueron descartados. En total se utilizaron 471 vídeos.

Metodología

1. Se detecta la cara del individuo para la normalización. Esto se basa en que la longitud de la mano de una persona adulta es aproximadamente

igual a la altura de su cara.

2. Se obtiene una serie temporal que representa la amplitud del movimiento de los dedos índice y pulgar de la mano dominante del individuo.

3. Se extraen un total de 15 características de esta serie temporal, por ejemplo:

   - Correlación cruzada media entre los máximos locales de dos intervalos distintos de tiempo de la serie temporal. Esto mismo se realiza también sobre los mínimos locales.
   - Número total de toques de dedos durante la grabación.
   - Velocidad media de la apertura de dedos.
   - Velocidad media del cierre de dedos.
   - ...

4. Se realiza una selección de características eliminando aquellas redundantes y usando el algoritmo chi-cuadrado.

5. Se entrena una máquina de vectores de soporte (SVM) mediante las características obtenidas para realizar la clasificación.

Resultados En cuanto a la distinción entre pacientes de la Enfermedad de Parkinson y el grupo de control se obtuvo una precisión del 95 %, que es una cifra que se debería tomar con precaución debido que, aunque se han utilizado 471 vídeos, estos provienen de únicamente 19 personas.

6.2. The discerning eye of computer vision

En este estudio [12] realizado sobre 39 pacientes con la Enfermedad de Parkinson y sobre un grupo de control de 30 individuos se tomaron vídeos de ambas manos de cada individuo mientras realizan toques de los dedos índice y pulgar (de forma similar a cómo se han tomado las muestras para este trabajo). Dando un total de 133 vídeos (se descartó uno).

De estos vídeos se han extraído diferentes características y comprobado la relación que existe entre éstas y diferentes escalas que clasifican el nivel de gravedad de la Enfermedad en un paciente, como la *Modified Bradykinesia Rating Scale* (MBRS) que categoriza el movimiento de los individuos en 5 niveles, del 0 al 4, siendo 0 un movimiento normal y 4 el nivel de mayor gravedad.

Metodología

1. Se utiliza una librería de visión por computador, en concreto DeepCutLab, para obtener una serie temporal de la amplitud entre las puntas de los dedos pulgar e índice.
2. Se normaliza esta serie temporal utilizando la amplitud máxima detectada, que va a convertirse en el valor 1, siendo todos los demás valores escalados proporcionalmente.
3. Se extraen las siguientes características:
   - Velocidad, calculada como la tasa media de cambio.
   - Variabilidad de la amplitud, calculada como el coeficiente de variación de la diferencia media entre máximos y mínimos de diferentes intervalos de 1 segundo de la serie temporal.
   - Regularidad del ritmo, calculada utilizando la Transformada Rápida de Fourier y, a continuación, midiendo la potencia de la frecuencia dominante más la potencia de las frecuencias en un intervalo de 0.4 Hz alrededor de ésta (un ritmo más regular concentra una mayor potencia en una única frecuencia).

Resultados Se observó una correlación bastante alta entre las características utilizadas y la categoría del individuo dentro de las escalas de medición de la Enfermedad de Parkinson utilizadas medida por un experto en el campo.

6.3. Supervised classification of bradykinesia

Este estudio [11] es muy similar al anteriormente explicado, y está realizado por un equipo compuesto por casi los mismos participantes. En este caso se utilizaron 70 vídeos, de ambas manos de 20 pacientes con la Enfermedad de Parkinson y de un grupo de control de 15 individuos.

Metodología La metodología es prácticamente igual que antes, la diferencia principal está en las características que se extraen de la serie temporal correspondiente con la amplitud, se ha obtenido:

- Frecuencia, medida como la frecuencia máxima de la Transformada Rápida de Fourier de la serie temporal.

- Amplitud, calculada como la densidad espectral, que se ha obtenido mediante la integral cuadrada del espectro de la Transformada Rápida de Fourier.

Con estas características se ha realizado clasificación binaria mediante clasificación bayesiana ingenua (naive bayes), regresión logística y máquina de vectores de soporte, tanto con función lineal como con función de base radial.

Resultados Los mejores resultados se obtuvieron con máquina de vectores de soporte con función de base radial, que coincide en un 73 % de los casos con la clasificación de expertos en el campo.

7. Conclusiones y Líneas de trabajo futuras

Bibliografía

[1] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77, 2018.

[2] Nathan Intrator. Feature extraction using an unsupervised neural network. In *Connectionist Models*, pages 310–318. Elsevier, 1991.

[3] Taha Khan, Dag Nyholm, Jerker Westin, and Mark Dougherty. A computer vision framework for finger-tapping evaluation in parkinson's disease. *Artificial intelligence in medicine*, 60(1):27–40, 2014.

[4] Erik G Learned-Miller. Introduction to supervised learning. *I: Department of Computer Science, University of Massachusetts*, page 3, 2014.

[5] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.

[6] OpenHub. MS Windows NT kernel description. https://www.openhub.net/repositories/compare. Accessed: 2023-03-13.

[7] Colin Shearer. The crisp-dm model: the new blueprint for data mining. *Journal of data warehousing*, 5(4):13–22, 2000.

[8] Wikipedia. Enfermedad de parkinson — wikipedia, la enciclopedia libre, 2023. [Internet; descargado 25-enero-2023].

[9] Wikipedia contributors. Data pre-processing — Wikipedia, the free encyclopedia, 2023. [Online; accessed 15-May-2023].

[10] Wikipedia contributors. Quantile normalization — Wikipedia, the free encyclopedia, 2023. [Online; accessed 1-May-2023].

[11] Stefan Williams, Samuel D Relton, Hui Fang, Jane Alty, Rami Qahwaji, Christopher D Graham, and David C Wong. Supervised classification of bradykinesia in parkinson's disease from smartphone videos. *Artificial Intelligence in Medicine*, 110:101966, 2020.

[12] Stefan Williams, Zhibin Zhao, Awais Hafeez, David C Wong, Samuel D Relton, Hui Fang, and Jane E Alty. The discerning eye of computer vision: Can it measure parkinson's finger tap bradykinesia? *Journal of the Neurological Sciences*, 416:117003, 2020.

[13] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019.

[14] Petros Xanthopoulos, Panos M Pardalos, Theodore B Trafalis, Petros Xanthopoulos, Panos M Pardalos, and Theodore B Trafalis. Linear discriminant analysis. *Robust data mining*, pages 27–33, 2013.

[15] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.