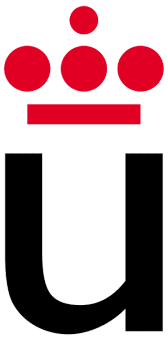

Universidad
Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

CURSO ACADÉMICO 2022/2023

Trabajo Fin de Grado

ANÁLISIS AUTOMÁTICO DE ALTERACIONES EN LA ESCRITURA MANUSCRITA DEBIDAS AL PÁRKINSON

Autor: Alberto Casademunt González

Tutor: Ángel Sánchez Calle

Resumen

La enfermedad de Parkinson es una afección neurodegenerativa crónica que causa temblores, rigidez y bradicinesia (lentitud a la hora de ejecutar movimientos), entre otros síntomas. Estos síntomas pueden presentar alteraciones en la escritura, algo característico del Párkinson, produciéndose deformaciones en los trazos al escribir y perturbaciones en la caligrafía de los sujetos (es decir, la fluidez en la motricidad de la escritura se degrada) . La detección de esas deformaciones y perturbaciones es usada en medicina como soporte para un posible diagnóstico.

El objetivo de este trabajo es llevar a cabo la detección de las alteraciones de forma programática mediante el uso de una red neuronal convolucional. Para ello haremos uso de la base de datos PaHaW (Parkinson's disease handwriting database), que consta de 75 sujetos que han llevado a cabo una serie de 8 tareas mediante escritura *online*.

Este trabajo se centra en la segunda tarea. De cada tarea se obtienen una serie de recortes con los que se entrena la red convolucional. Una vez entrenada, la red permite realizar predicciones sobre cada recorte de forma individual. El proyecto hace uso de esas predicciones para clasificar trazos, letras y tareas.

El código ha sido desarrollado en Python haciendo uso de librerías como NumPy y PyTorch. Además se ofrece un cuaderno de Jupyter para facilitar su uso. Durante el desarrollo se han llevado a cabo una serie de pruebas que han servido para ajustar valores como el tamaño de los recortes o los sujetos usados para entrenar la red convolucional, intentando conseguir una corrección en las predicciones superior al 60%.

A lo largo de esta memoria se describirá en profundidad la base de datos usada, el proceso aplicado a cada tarea para generar los conjuntos de datos necesarios para entrenar la red neuronal convolucional, la estructura de la red neuronal convolucional, la obtención de las predicciones y el uso de esas predicciones para clasificar trazos, letras y tareas.

Palabras clave: PaHaW, red neuronal convolucional, Python, *CNN*, aprendizaje computacional, enfermedad de Parkinson.

Summary

Parkinson's disease is a chronic neurodegenerative disorder that causes tremors, stiffness, and bradykinesia (slowness of movement), among other symptoms. These symptoms can cause handwriting alterations, and their detection is used as support for a possible medical diagnosis.

The purpose of this work is to carry out the detection of handwriting alterations using a convolutional neural network. For this, we will use the PaHaW (Parkinson's disease handwriting database), which consists of 75 subjects who have carried out a series of eight tasks through online writing.

This work focuses on the second task. For each task, a series of clippings is generated to train the convolutional network. Once trained, the network is able to make predictions of each clipping individually. The project uses these predictions to classify strokes, letters and tasks.

The code has been developed in Python using libraries such as NumPy and PyTorch. Additionally, a Jupyter notebook is offered to facilitate its use. During the development, a series of experiments have been performed to adjust values such as the size of the clippings and the subjects used to train the convolutional network, trying to achieve a prediction accuracy greater than 60%.

Through this document, we will describe the database, the process applied to each task to generate the datasets used to train the convolutional neural network, the structure of the convolutional neural network, how to get the predictions and how to use those predictions to classify strokes, letters and tasks.

Keywords: PaHaW, convolutional neural network, Python, CNN, Machine Learning, Parkinson's Disease.

Agradecimientos

En primer lugar me gustaría expresar mi más sincero agradecimiento a mi tutor, Ángel, por darme la oportunidad de llevar a cabo este proyecto junto a él. A lo largo de la realización de este trabajo siempre ha hecho gala de su paciencia, saber docente, conocimientos e implicación en el campo de la Visión Artificial.

Gracias a aquellos con los que empecé esta aventura que ha sido el Grado en Ingeniería Informática. Antonio, David, Francisco Tomás y Narciso. Los buenos momentos vividos en aquél primer año me acompañarán toda la vida.

Gracias también a Manuel y Miryam, por estar ahí en los buenos y en los malos momentos.

A mis padres Rosalía y Juan José. Por su eterna paciencia, confianza y comprensión. Sin ellos jamás podría haber llegado a este punto.

A mi hermano Jordi, por acompañarme desde que tengo uso de razón y haberme servido siempre como ejemplo a seguir.

Tabla de contenido

Índice de figuras

Índice de tablas

1. Introducción

El Párkinson es una afección degenerativa del cerebro asociada a varios síntomas motores (lentitud de movimientos, temblores, rigidez, trastornos de la marcha y desequilibrio) y a una amplia variedad de complicaciones no motoras. Se desconocen las causas de la enfermedad, pero se cree que puede deberse a una compleja interacción entre factores genéticos y la exposición a factores ambientales como los plaguicidas, los disolventes y la contaminación atmosférica a lo largo de la vida. A nivel mundial, la discapacidad y las defunciones debidas a la enfermedad de Parkinson están aumentando más rápidamente que las de cualquier otro trastorno neurológico [7].

El diagnóstico de la enfermedad de Parkinson puede ser establecido por trabajadores de salud no especializados en el campo neurológico, algo que es especialmente importante en zonas que no disponen de servicios neurológicos especializados.

1.1. Motivación del proyecto

Actualmente no existe un método objetivo para diagnosticar la enfermedad de Parkinson. El diagnóstico se basa en el análisis y monitoreo de los síntomas, en ocasiones a lo largo de varios meses. La probabilidad de obtener un diagnóstico erróneo es de aproximadamente un 25% [2].

El desarrollo de herramientas que ayuden a realizar un diagnóstico precoz es beneficioso para que los pacientes puedan empezar el tratamiento necesario lo antes posible.

Algunos de los síntomas motores de la enfermedad de Parkinson como los temblores y la rigidez pueden producir alteraciones en la escritura manuscrita [2], y la detección de esas alteraciones es una de las herramientas que pueden usarse durante el diagnóstico de la enfermedad.

1.2. Objetivos del proyecto

Este proyecto tiene como objetivo clasificar tareas manuscritas y trazos de esas tareas en dos grupos: H (sanos) y PD (*Parkinson's Disease*). Para ello se hará uso de una red neuronal convolucional (*CNN*) que intentará identificar alteraciones en los trazos debidas a la enfermedad de Parkinson.

Para conseguir el objetivo es necesario llevar a cabo una serie de pasos:

1. **Procesar la base de datos.** La base de datos usada en este trabajo ofrece muestras de escritura *online*. Es necesario crear procesos para poder obtener imágenes a partir de los puntos de coordenadas.
2. **Generar un conjunto de datos.** Para poder entrenar la red neuronal convolucional es necesario generar un conjunto de datos. Este conjunto ha de estar equilibrado, es decir, ha de tener un número similar de muestras de cada clase (en este caso H y PD).
3. **Crear y entrenar la *CNN*.** En este proyecto se hace uso de una red neuronal convolucional multicapa. Una vez creada se entrena con el conjunto de datos generado y permite realizar las predicciones.
4. **Procesar las predicciones.** Una vez generadas las predicciones de una tarea es necesario procesar los datos obtenidos para obtener las predicciones sobre los trazos de esa tarea y sobre la propia tarea. Más concretamente, se trata de determinar los fragmentos de trazos donde se aprecian síntomas de Párkinson en una tarea y, también, clasificar las letras de la tarea como pertenecientes a sujetos H o PD.

1.3. Organización de la memoria

Primero, se hará un breve resumen de varios documentos previamente publicados que tienen relación con la detección de alteraciones en la escritura debidas a la enfermedad de Parkinson, lo que será útil para conocer diferentes métodos y clasificadores usados y poder tener un conocimiento mayor de los resultados obtenidos.

A continuación, se describirá en profundidad la base de datos usada en este proyecto y los procesos aplicados sobre ella para generar el conjunto de datos necesario para entrenar la *CNN*. También se explicará la *CNN* usada y las diferentes capas que la componen.

Seguidamente, se mostrarán diferentes parámetros usados a la hora de crear el conjunto de datos y los resultados obtenidos, siempre intentando mejorar la corrección en las predicciones.

Por último, se analizará todo el trabajo realizado, las dificultades encontradas, las soluciones planteadas y los resultados obtenidos, y en base a ello se establecerán las conclusiones del trabajo.

En la sección final se incluirán una serie de anexos sobre la plataforma usada para el desarrollo del proyecto y el entrenamiento de la *CNN* y una explicación detallada de los parámetros necesarios para poder reproducir los resultados mostrados.

2. Estado del arte

La detección automática de alteraciones en la escritura para clasificar sujetos en diferentes categorías (p. ej., sexo, mano de escritura, etc.) cuenta con una bibliografía relativamente escasa; mucho más a la hora de centrarse en alteraciones debidas a diferentes enfermedades médicas.

El objetivo de la creación de la base de datos PaHaW [2] fue, en parte, intentar fomentar el nacimiento de nuevos trabajos e investigaciones y proporcionar información adicional a otras bases de datos ya existentes en ese momento.

A continuación se presenta un breve análisis de diferentes trabajos relacionados con la detección de la enfermedad de Párkinson en la escritura manuscrita.

- "*Evaluation of handwriting kinematics and pressure for differential diagnosis of Parkinson's disease*" [2]
  Este artículo fue escrito por los creadores de la base de datos PaHaW para describir la base de datos, mostrar una serie de pruebas realizadas sobre ella y presentar posibles caminos a seguir para los investigadores futuros. Las pruebas se centran en las características cinemáticas que ofrece PaHaW gracias a la escritura *online*, como la presión o la velocidad. Para realizar la clasificación se muestra el uso de tres técnicas de aprendizaje automático: *SVM*, clasificación *AdaBoost*, y el algoritmo *K-NN*.

- "*Analysis of in-air movement in handwriting: A novel marker for Parkinson's disease*" [1]
  Al igual que el artículo anterior, éste también fue escrito por los creadores de la base de datos PaHaW. En esta ocasión los autores se enfocan en el análisis de las características cinemáticas asociadas al movimiento del bolígrafo cuando se encuentra en el aire. Para clasificar los sujetos usaron una máquina de vectores de soporte (*SVM)*.

- "*Dynamic Handwriting Analysis for the Assessment of Neurodegenerative Diseases: A Pattern Recognition Perspective*" [4]

  Este trabajo se centra en mostrar los resultados más relevantes en la clasificación de la enfermedad de Parkinson y la enfermedad de Alzheimer usando el análisis de las características *online* que ofrecen diferentes bases de datos. La conclusión final es que la escritura manuscrita es un buen biomarcador a la hora de detectar la enfermedad de Alzheimer y la enfermedad de Parkinson.

- "*Computational Analysis of Open Loop Handwriting Movements in Parkinson's Disease: A Rapid Method to Detect Dopamimetic Effects*" [3]

  El trabajo de este artículo se focaliza en analizar los efectos de la apomorfina en la escritura de personas afectadas por la enfermedad de Parkinson. El estudio se centró en 16 sujetos que presentaban síntomas tempranos de la enfermedad de Parkinson con respuesta a la *L-DOPA* (levodopa, precursor metabólico de la dopamina), 6 sujetos con enfermedad de Parkinson y alteraciones motoras asociadas a la *L-DOPA*, y 7 sujetos con Párkinson sin respuesta a la *L-DOPA*. Se le pidió a todos los sujetos realizar una serie de tareas usando una tableta digitalizadora. Posteriormente se les administró una inyección de apomorfina y se les pidió repetir las tareas. La conclusión del trabajo fue que el análisis asistido por computación de la escritura manuscrita puede ser usado como un método rápido y objetivo para conocer los efectos relacionados con la dopamina sobre las propiedades cinemáticas de la escritura.

Como conclusión del estado del arte de este trabajo, puede indicarse que la mayoría de la bibliografía establecida se centra en el análisis y procesado de los datos que ofrece la escritura *online* a la hora de crear clasificadores, siendo mucho menos común un enfoque como el seguido en este proyecto, que es usar las coordenadas de la escritura *online* para generar imágenes de recortes de trazos y entrenar una red neuronal convolucional usando las imágenes generadas.

3. Base de datos

En este proyecto se ha hecho uso de la base de datos PaHaW (*The Parkinson's Disease Handwriting Database*). PaHaW fue creada por el BDALab (*The Brain Diseases Analysis Laboratory*) en colaboración con el Centro de desórdenes del movimiento (Primer Departamento de neurología, Universidad de Masaryk) y el Hospital Universitario Santa Ana en Brno, República Checa.

3.1. Base de datos PaHaW

A cada sujeto se le pidió que completase una serie de tareas de escritura manuscrita en base a una plantilla previamente completada, mostrada en la **Figura 3.1**. Antes de empezar cada tarea los sujetos pudieron observar la plantilla completada y tuvieron limitaciones en el número de repeticiones de sílabas/palabras.

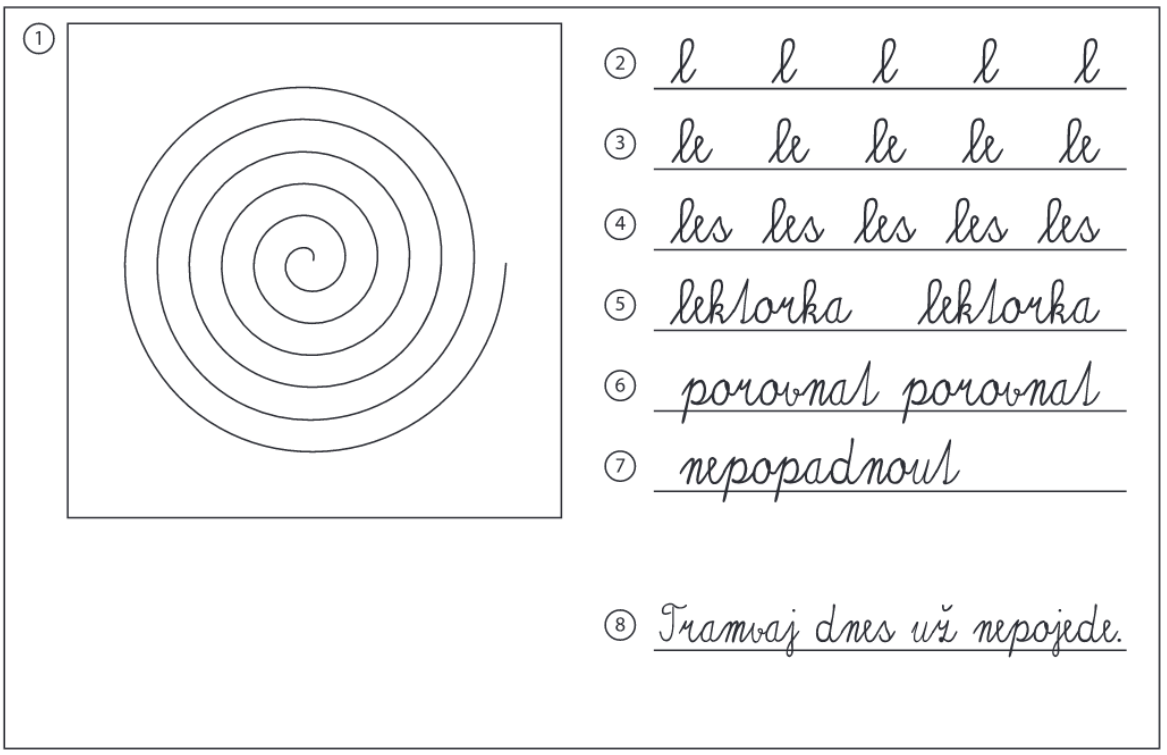

Fig. 3.1 Plantilla usada por los sujetos durante la creación de la base de datos PaHaW.

Las tareas fueron realizadas sobre una tableta Wacom Intuos 4M con una frecuencia de muestreo de 150 Hz. Además de generar señales durante la escritura, también se generaron cuando el bolígrafo se encontraba en el aire.

La base de datos cuenta con 37 sujetos que sufren la enfermedad de Parkinson y 38 sujetos sanos de control. Los dos grupos están formados por sujetos que coinciden en edad y sexo. Todos los sujetos son diestros y de nacionalidad checa.

	A	B	C	D	E	F	G	H	I	J
1	ID	Nationality	Sex	Disease	PD status	Age	Dominant hand	LED	UPDRS V	Length of PD
2	00001	Czech	F	PD	ON	68	R	1115	2	6
3	00002	Czech	F	PD	ON	78	R	2110	2	8
4	00003	Czech	F	PD	ON	69	R	1556.6	2	7
5	00004	Czech	F	PD	ON	79	R	1691	2	12
6	00005	Czech	F	PD	ON	69	R	600	2	2
7	00006	Czech	F	PD	ON	57	R	1271.66	2	9

Fig. 3.2 Datos que ofrece PaHaW sobre cada uno de los sujetos.

En la **Figura 3.2** se han mostrado los datos que ofrece la base de datos PaHaW, y son los siguientes:

- **ID:** La ID de cada sujeto se encuentra en el rango 1-98. Al ser 75 sujetos hay algunos números sin usar. Es necesario conocer este valor para poder acceder a las tareas de cada sujeto.
- **Nationality:** En este caso todos los sujetos son de la República Checa.
- **Sex:** Los sujetos están divididos en 39 hombres y 36 mujeres.
- **Disease:** El valor será *PD* para los sujetos que padecen la enfermedad y *H* para los sujetos sanos de control.
- **PD status:** Ofrece información similar a la columna anterior. Si el valor es *ON*, el sujeto padece PD, si no tiene valor es un sujeto de control.
- **Age:** La edad media de los sujetos es de 65,82 años y la mediana de 65.
- **Dominant hand:** Todos los sujetos son diestros.
- **LED:** *Levodopa equivalent dose*. La administración de un medicamento que produce los mismos efectos antiparkinsonianos que 100 mg de levodopa [2].

- **UPDRS V:** Es la escala más usada a la hora de calcular la severidad de la enfermedad de Parkinson. Se basa en entrevistas y observaciones clínicas centradas en la evaluación de las funciones motoras del paciente [4].
- **Length of PD:** Los sujetos PD llevan sufriendo la enfermedad una media de 8,38 años y una mediana de 8.

Como se usa una tableta digital para guardar las tareas, la base de datos PaHaW ofrece datos de escritura de tipo *online*. En vez de ofrecer textos digitalizados lo que ofrece es una lista de puntos de coordenadas con unas características asociadas a cada punto.

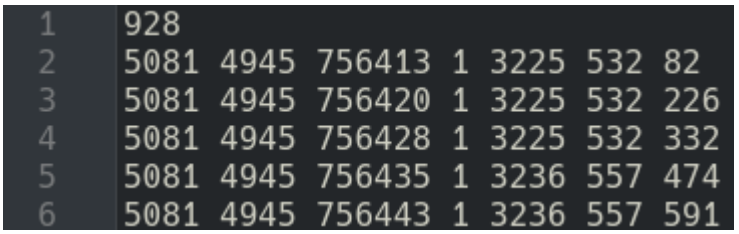

Fig. 3.3 Datos que ofrece PaHaW sobre cada tarea.

En la **Figura 3.3** se muestra el inicio de un archivo *.svc* asociado a una de las tareas que ofrece PaHaW. La primera línea indica el número de puntos de coordenadas almacenados de esa tarea. La segunda línea y sucesivas marcan un nuevo punto de coordenadas. Al ser la frecuencia de muestreo de 150 Hz, por cada segundo se generarán 150 puntos de coordenadas. En cada punto de coordenadas la información guardada es la siguiente:

1. **Coordenada Y.**
2. **Coordenada X.**
3. **Marca de tiempo:** Momento en el que se ha generado el punto de coordenadas en milisegundos.
4. **Estado:** 1 (en superficie) o 0 (en el aire).
5. **Acimut:** Inclinación de la proyección del bolígrafo sobre la superficie respecto al norte de la misma.
6. **Altura:** Ángulo de inclinación del bolígrafo respecto a la horizontal de la superficie.
7. **Presión:** Presión del bolígrafo sobre la superficie.

4. Red neuronal

Este proyecto hace uso de una red neuronal convolucional (también llamada *ConvNet* o *CNN*), que forma parte del concepto *machine learning* (aprendizaje computacional).

4.1. Aprendizaje computacional

El aprendizaje computacional es una rama de la Inteligencia Artificial que se centra en el uso de datos y algoritmos para imitar la forma en la que los humanos aprenden, aumentando de forma gradual su precisión [13].

4.2. Redes neuronales

Las redes neuronales, también llamadas redes neuronales artificiales (*ANN*) o redes neuronales simuladas (*SNN*) son un subconjunto del aprendizaje computacional y marcan la base de los algoritmos de *deep learning* (aprendizaje profundo).

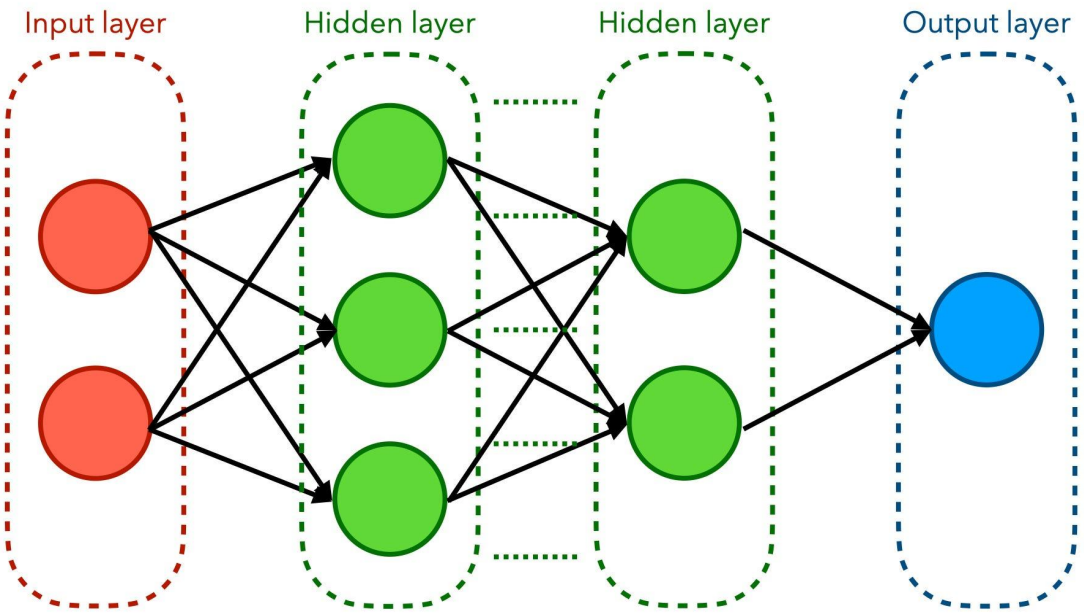

Fig. 4.1 Ilustración de una red neuronal con dos capas ocultas [6].

Las redes neuronales están formadas por una serie de capas de nodos, teniendo una capa de entrada, una de salida y una o más capas ocultas entre ellas, tal y como se muestra en la **Figura 4.1**. Cada nodo está conectado con otro y tiene asociado un peso y umbral. Si la salida de un nodo está por encima del umbral elegido, ese nodo es activado y manda los datos a la siguiente capa de la red. Si, por el contrario, la salida no sobrepasa el umbral, el nodo no se activa y no transmite los datos.

Para poder aprender y mejorar su precisión, las redes neuronales necesitan un conjunto de datos de entrenamiento. Una vez entrenada, la red neuronal puede ser usada para clasificar grandes cantidades de datos en poco tiempo [12].

4.3. Red neuronal convolucional

En este trabajo el clasificador usado es una red neuronal convolucional (*CNN* de ahora en adelante). Una *CNN* se diferencia de otras redes neuronales por un mejor rendimiento a la hora de clasificar imágenes y audio. Este tipo de redes suelen estar formadas por tres tipos de capas.

4.3.1. Capa convolucional

La capa convolucional es la base sobre la que se construye una *CNN*. En una *CNN* siempre hay una capa de este tipo en la entrada, aunque puede haber más posteriormente. Es aquí donde se consume el mayor tiempo de cálculo. La capa convolucional recibe una entrada, por ejemplo, una matriz de píxeles. La entrada se recorre celda por celda con un filtro o *kernel*, que normalmente es una matriz de tamaño 3×3 o superior.

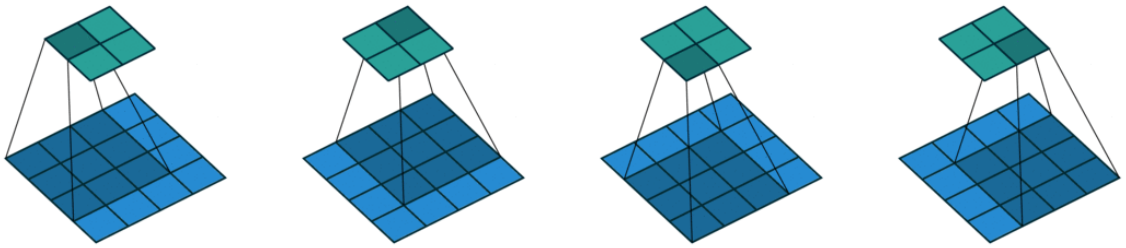

Fig. 4.2 Kernel de 3×3 aplicado a una entrada de tamaño 4×4 [10].

A la hora de aplicar un *kernel* hay una serie de valores que hay que tener en cuenta:

- ***Stride*:** Marca el salto que dará el kernel entre aplicación y aplicación. En la **Figura 4.2** el *stride* es 1.

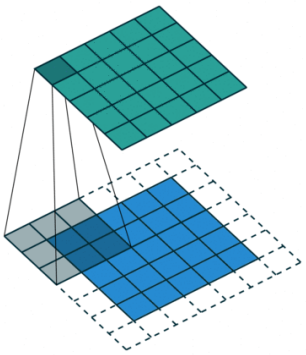

Fig. 4.3 Uso de *padding* sobre una entrada al aplicar un kernel [10].

- ***Padding*:** Sirve para que cada píxel de la entrada pueda ser expuesto al centro del *kernel*. Si no se utiliza *padding*, dado un *kernel* de 3×3, si el primer píxel de la entrada se encuentra en (0, 0) el centro del kernel empezará en (1, 1). El efecto de no utilizar *padding* es, además, que la salida será de menor tamaño que la entrada. Mostrado en la **Figura 4.3**.

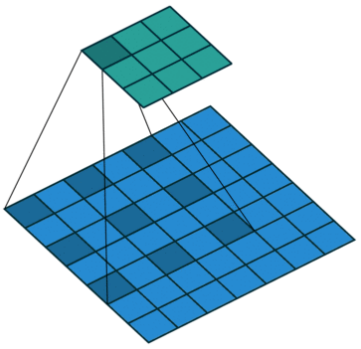

Fig. 4.4 Kernel con *dilation* 1 [10].

- ***Dilation*:** Marca el espacio entre los puntos del *kernel*. En la **Figura 4.4** se muestra visualmente.
- **Tamaño del *kernel*:** Puede ser un entero que marque el lado de un cuadrado o dos números que marquen la altura y la anchura. Lo usual es que sea una matriz cuadrada de 3×3.

4.3.2. Capa *pooling* (de agrupación)

Este tipo de capas sirven para reducir la dimensionalidad. Al igual que en las capas convolucionales se aplica un *kernel* sobre la entrada, pero mientras que el *kernel* de las primeras está compuesto por una serie de pesos, en el caso de las capas de agrupación existen dos tipos de *kernel*:

- ***Max pooling*:** Es el tipo más usado. A medida que el *kernel* recorre la entrada selecciona los máximos valores que encuentra.
- ***Average pooling*:** A medida que el *kernel* recorre la entrada calcula el valor medio de todas las celdas que abarca.

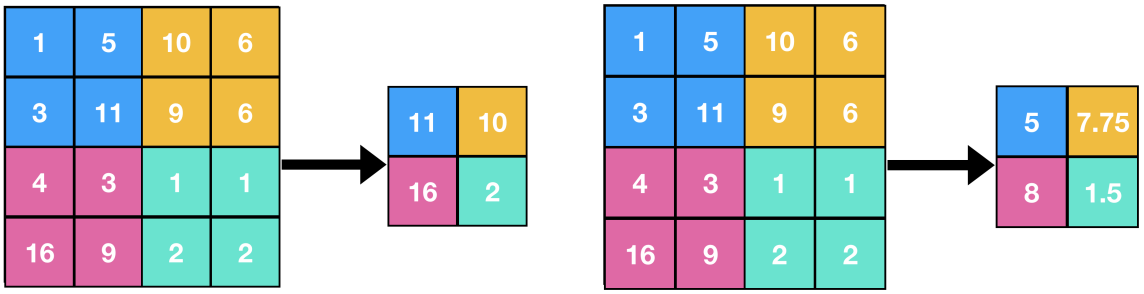

Fig. 4.5 Diferencias entre *max pooling* (izquierda) y *average pooling* (derecha) [8].

En la **Figura 4.5** puede verse que en estas capas se pierde mucha información, pero por contra tienen beneficios como reducir la complejidad, mejorar la eficiencia o reducir el riesgo de *overfitting* (la red neuronal tiene una corrección alta al generar predicciones sobre el conjunto de entrenamiento pero no cuando genera predicciones sobre entradas desconocidas) [11]. Los valores a tener en cuenta son similares a los de las capas convolucionales: *Stride, padding, dilation* y tamaño de *kernel*.

4.3.3. Capa completamente conectada

Mientras que una capa convolucional depende de un *kernel* que se desplaza sobre la entrada para generar la salida, en el caso de este tipo de capas todos los puntos de la entrada están conectados a todos los puntos de la salida. En todas las *CNN* hay al menos una capa completamente conectada para producir la salida final, tal y como se muestra en la **Figura 4.6**.

Fig. 4.6 Capa convolucional (izquierda) y capa completamente conectada (derecha) [9].

4.3.4. Entrenamiento de la *CNN*

Para entrenar la red neuronal convolucional es necesario tener en cuenta los siguientes parámetros:

- **Tamaño de *batch*:** Indica el número de muestras que la *CNN* procesa antes de actualizar el modelo. A mayor tamaño, mayor uso de memoria y menor tiempo de entrenamiento, produciendo además mejores resultados para el conjunto de entrenamiento. Usar tamaños más comedidos reduce el uso de memoria y contribuye a evitar que el entrenamiento genere *overfitting*, pero a cambio el tiempo para entrenar la red neuronal aumenta.
- ***Epochs*:** Define las iteraciones que el algoritmo de entrenamiento se aplicará a todo el *dataset* de entrenamiento. En el transcurso de un *epoch* todas las entradas del *dataset* tienen la oportunidad de actualizar los parámetros internos del modelo [14].

5. Solución informática

En este apartado se hablará en profundidad de la solución informática que ha sido desarrollada para intentar conseguir los objetivos expuestos en la introducción de este documento. Para lo más relevante de esta solución se ha utilizado:

- **Lenguaje de programación:** *Python*.
- **Tratamiento de la base de datos:** *pandas* (*Python Data Analysis Library*).
- ***CNN*:** *PyTorch*.
- **Generación de imágenes:** *Matplotlib*.
- **Generación de estadísticas:** *scikit-learn* y *seaborn*.

5.1. Estructura de archivos y de clases

Los archivos que el proyecto necesita para funcionar son:

- **PaHaW/PaHaW_public:** Contiene un directorio por cada sujeto. Cada directorio contiene 8 archivos *.svc*, uno por cada tarea.
- **PaHaW/PaHaW_files/corpus_PaHaW.xlsx:** Archivo que contiene los datos de todos los sujetos. Los datos que necesita este proyecto son la ID y el estado PD.

En la **Figura 5.1** se muestra la estructura de archivos usada por el proyecto en mayor detalle. Con esos archivos se generan las instancias de las clases mostradas en la **Figura 5.2.** y los recortes usados para entrenar la *CNN* y realizar las predicciones.

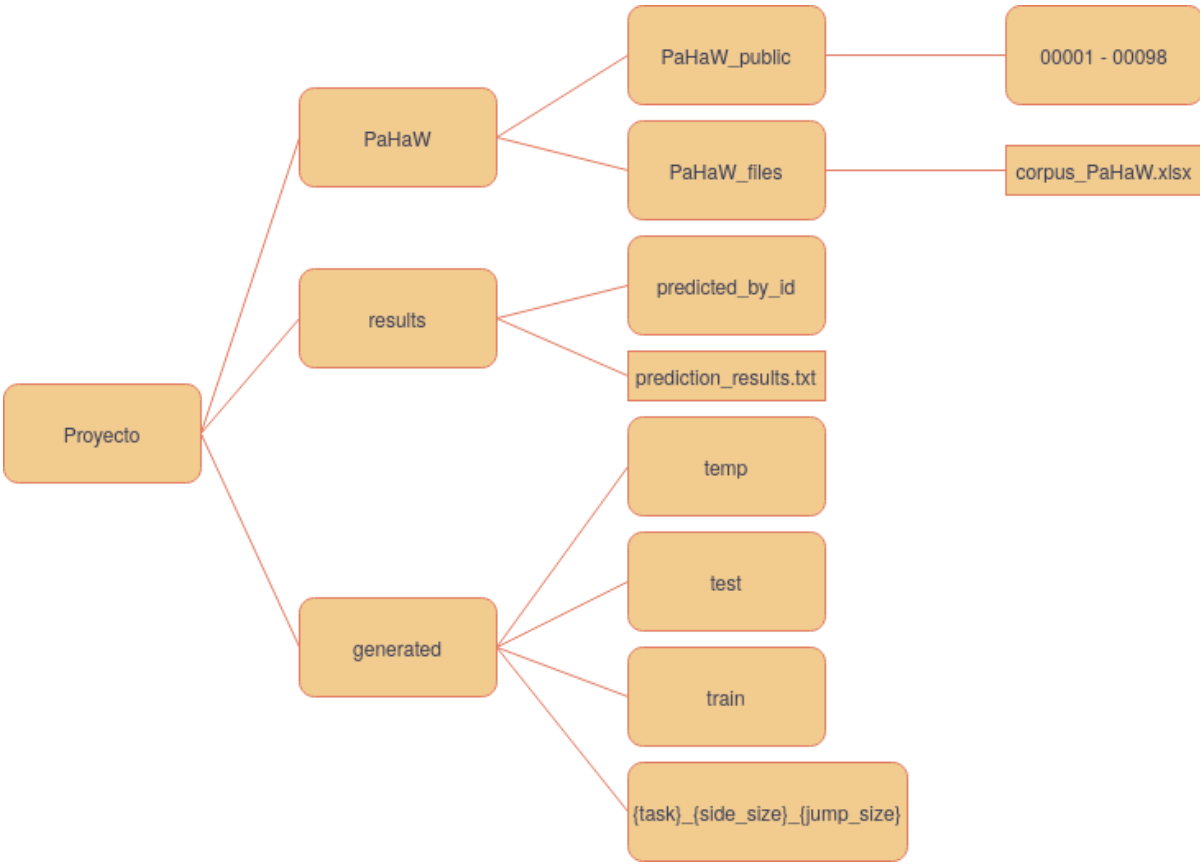

Fig. 5.1 Estructura de archivos del proyecto.

El proyecto genera una serie de archivos tanto para su funcionamiento interno como para mostrar el resultado final.

- **Directorio *generated*:** Aquí es donde el proyecto almacena los archivos que genera a partir de la base de datos PaHaW y del entrenamiento de la *CNN*. Contiene directorios con imágenes de recortes generados y modelos de redes neuronales convolucionales previamente entrenados. Los recortes se almacenan en un directorio según su tamaño y salto.
- **Directorio *results*:** Aquí es donde se almacenarán los resultados finales de las predicciones realizadas. Contiene un directorio ***predicted_by_id*** con las predicciones realizadas sobre la tarea de cada uno de los sujetos. También se genera el archivo ***prediction_results.txt*** al que se añadirán datos estadísticos sobre el conjunto de todas las predicciones realizadas en esa ejecución. Por último se generan dos imágenes que contienen las matrices de confusión asociadas a los datos estadísticos ya mencionados.

El proyecto se ha dividido en varios ficheros de código.

- **pahaw_loader.py** Contiene las clases mostradas en el diagrama de la **Figura 5.2** y el método que carga la base de datos y genera las instancias necesarias de las mismas.
- **nn.py** Aquí está la clase que define la estructura del *dataset*, la estructura de la *CNN* y el método para entrenarla.
- **PD_Prediction.ipynb** Se ofrece un cuaderno de Jupyter para facilitar el uso del proyecto.

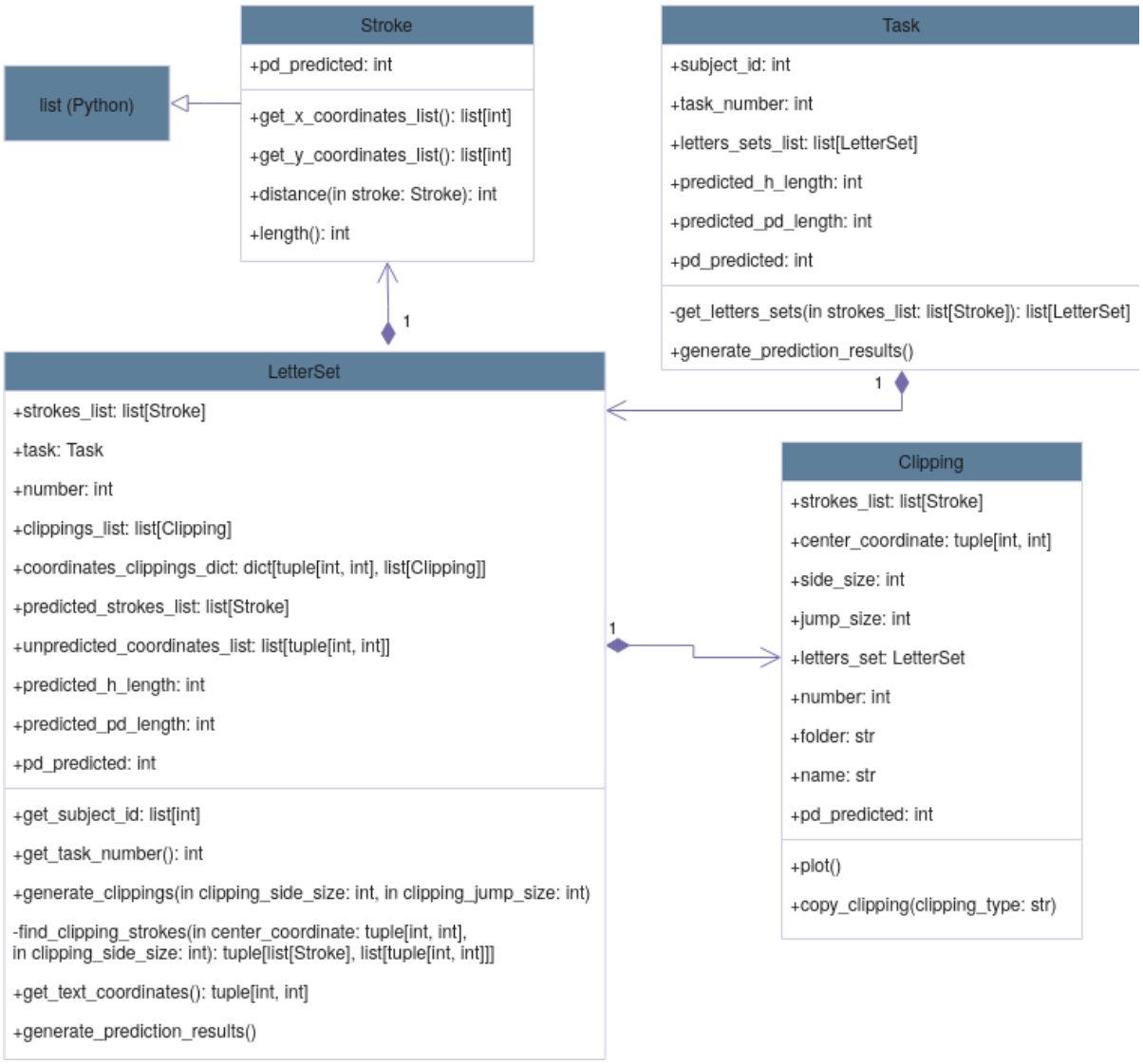

Fig. 5.2 Diagrama UML de las clases usadas para almacenar y tratar las tareas cargadas.

5.2. Generación de recortes

Este proyecto ha centrado sus objetivos en la Tarea 2 de la base de datos PaHaW, donde los sujetos escriben una secuencia de letras. Por similitudes el trabajo realizado también puede ser usado para las Tareas 3 y 4, algo que se mostrará en los anexos finales de esta memoria. Por lo tanto, a partir de ahora se hará referencia únicamente a la Tarea 2 salvo que se indique lo contrario.

Fig. 5.3 Plantilla de la Tarea 2 de la base de datos PaHaW.

La base de datos PaHaW ofrece un total de 7 datos sobre cada punto de coordenadas que forma parte de una tarea. Como este trabajo se centra en realizar una clasificación basada en imágenes, los únicos campos relevantes son los de las coordenadas (x, y) y el del estado del bolígrafo (en superficie o en el aire). En la **Figura 5.4** se muestran los datos de la escritura *online* de la Tarea 2 del sujeto 53. Aunque este trabajo no los utilice de forma programática, puede ser interesante su observación.

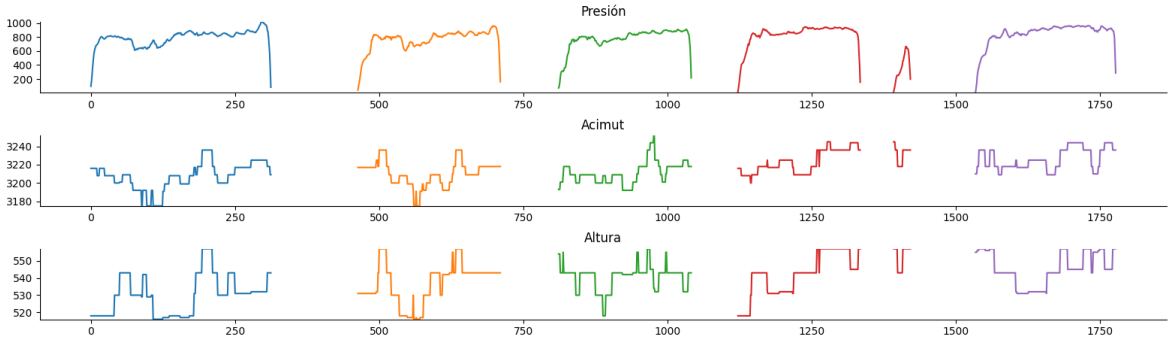

Fig. 5.4 Datos *online* asociados a la Tarea 2 del sujeto 53.

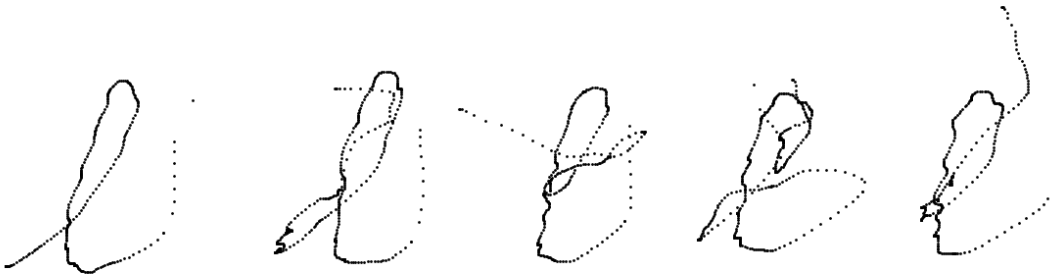

Fig. 5.5 Todos los puntos de coordenadas del sujeto 53.

Hay que recorrer los archivos de cada tarea que nos ofrece PaHaW. Una primera aproximación es la de la **Figura 5.5**, mostrar todos los puntos para conocer mejor lo que ofrece la base de datos. Se aprecian las cinco letras pero también otros puntos ajenos a ellas. Son puntos en el aire.

Fig. 5.6 Los puntos de coordenadas del sujeto 53 descartando los puntos en el aire.

Una vez se descartan los puntos en el aire quedan cinco letras bien definidas como puede verse en la **Figura 5.6**. El siguiente paso es mostrar las líneas entre los puntos.

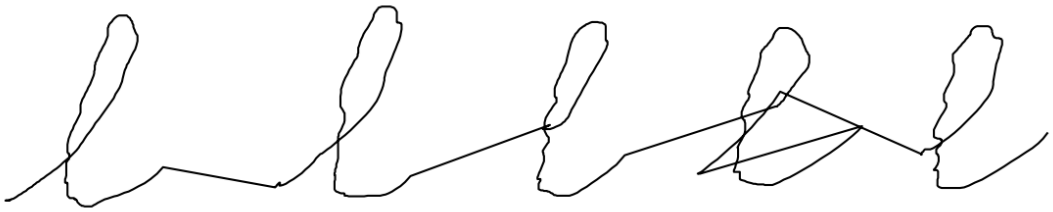

Fig. 5.7 Líneas entre los puntos de coordenadas del sujeto 53.

Como puede observarse en la **Figura 5.7** mostrar directamente las líneas entre los puntos no es suficiente, es necesario agrupar los puntos en diferentes trazos para poder mostrar la tarea correctamente. Para ello se hace uso del estado del bolígrafo en cada punto. Un nuevo trazo se crea cada vez que el bolígrafo pasa de estar en el aire a estar sobre la superficie.

Fig. 5.8 Trazos del sujeto 53. Cada color representa un trazo.

Como puede verse en la **Figura 5.8**, agrupar las coordenadas sobre la superficie en trazos permite mostrar la tarea de forma correcta. Una vez se han agrupado los puntos en diferentes trazos se procede a agrupar los trazos en letras tal y como se muestra en la **Figura 5.9**.

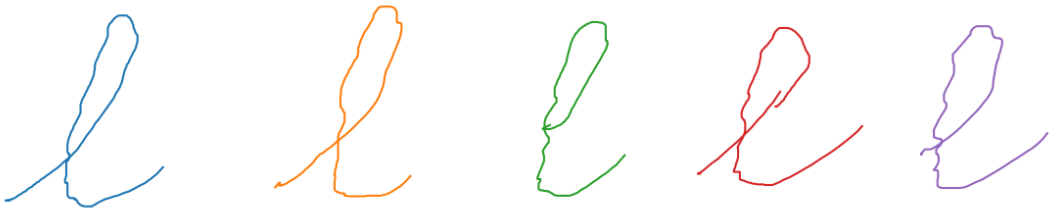

Fig. 5.9 Trazos del sujeto 53 agrupados en letras. Cada color representa una letra.

Agrupar los trazos en letras permite realizar predicciones a ese nivel. Para ello se ordenan los trazos por longitud y se agrupan todos los trazos en torno a los cinco trazos más largos. Se ha adaptado el método para que funcione con algunas tareas que no siguen la plantilla como la mostrada en la **Figura 5.10**.

Fig. 5.10 Tarea del sujeto 57 con seis letras.

El conjunto de datos sobre el que se centra el trabajo es una serie de recortes realizados sobre los puntos de la tarea. Sobre la **Figura 5.11** se muestran algunos de esos recortes.

Fig. 5.11 Diferentes recortes de tamaño 45 marcados sobre la tarea del sujeto 53.

Cada recorte se genera sobre un punto de coordenadas, marcando éste el centro del mismo. En las **Figuras 5.11 y 5.12** se presentan los recortes correspondientes a los puntos 300, 600, 900 y 1200 de la Tarea 2 del sujeto 53.

Fig. 5.12 Los recortes mostrados en la **Figura 5.11** ya generados.

Para generar los recortes es necesario recorrer todas las coordenadas de todos los trazos en orden, generando los trazos del propio recorte de forma similar a como se generaron para la tarea. Cada vez que una coordenada entra en el perímetro del recorte se genera un nuevo trazo. De cada recorte se guarda, además, una lista de todas las coordenadas que contiene. La cantidad de recortes generados y su tamaño depende de dos valores: tamaño de salto entre recortes y tamaño de los recortes, respectivamente.

5.2.1. Tamaño del salto entre recortes

La cantidad de recortes generados depende del salto que exista entre cada recorte. Si el tamaño de salto es 1 se generará un recorte por cada punto de coordenadas. Si, por ejemplo, fuese 5, se generarían recortes para los puntos de coordenadas 0, 5, 10, 15…

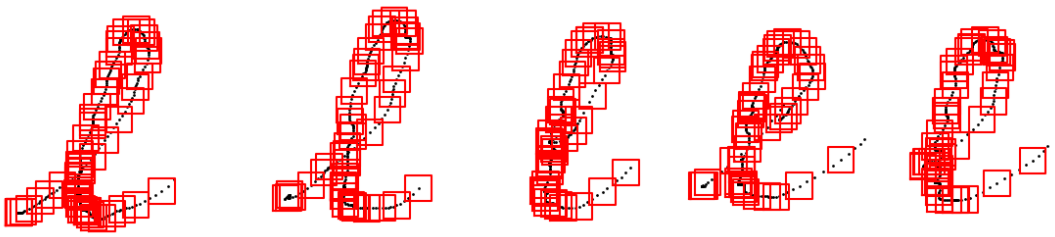

Fig. 5.13 Recortes del sujeto 53 con salto 5 y tamaño 45.

En la cuarta letra de la **Figura 5.13** se presenta el problema de que el tamaño de salto sea mayor de 1. Hay puntos que pueden quedar fuera de los recortes y por tanto fuera de las predicciones que realice la *CNN*. En la **Figura 5.14** se muestra una ampliación de la zona que presenta el problema.

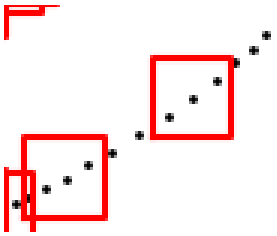

Fig. 5.14 Zoom sobre la zona afectada por el tamaño de salto 5.

Por lo tanto para este proyecto se ha decidido usar un tamaño de salto 1. De esta manera se genera un recorte por cada punto y como se aprecia en la **Figura 5.15** el problema se solventa.

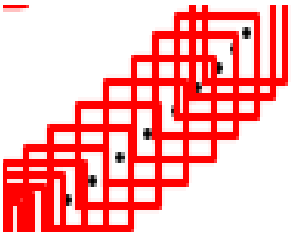

Fig. 5.15 Zoom sobre la misma zona con tamaño de salto 1.

De forma añadida, el hecho de generar un recorte por cada punto hace que como resultado se obtengan un total de 83.955 recortes, de media 1119,4 por cada tarea. Aún así, para poder generar predicciones correctas es necesario que en cada recorte estén presentes al menos dos puntos para poder representar un trazo.

5.2.2. Tamaño de los recortes

Una vez establecido que lo óptimo es usar un tamaño de salto 1 ahora el objetivo es que todos los recortes tengan al menos dos puntos, para lo que será necesario encontrar un tamaño de recorte mínimo a partir del cual se cumple la condición para todos los recortes generados.

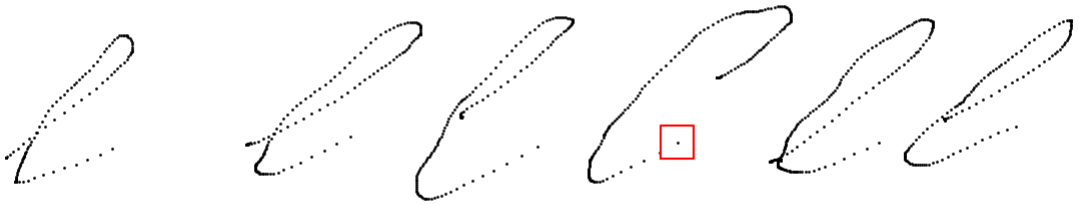

Fig. 5.16 Con tamaño de recorte 69 hay un recorte de los 83955 que solo tiene un punto.

Realizando pruebas se ha calculado que el tamaño mínimo de recorte para que todos los recortes cumplan este requisito es de 71. En la **Figura 5.16** se muestra que con tamaño de recorte 69 todavía existen recortes que contienen un solo punto de coordenadas. Para ver por qué es un problema que un recorte tenga un solo punto de coordenadas en la **Figura 5.17** se muestra la imagen que corresponde al recorte marcado en la **Figura 5.16** con tamaño de recorte 69 y 71. Observando las imágenes generadas queda claro que para dibujar un trazo es necesario tener al menos dos puntos de coordenadas dentro del recorte.

Fig. 5.17 Mismo recorte con tamaño 69 (izquierda) y 71 (derecha).

5.3. Creación del conjunto de datos

Un conjunto de datos está formado por un diccionario de imágenes y un diccionario de etiquetas asociadas a cada imagen. Las imágenes para el diccionario se generan como se explicó en el apartado anterior, y las etiquetas son el estado H o PD del sujeto al que pertenece cada recorte, indicando H que el recorte pertenece a un sujeto sano de control y PD que pertenece a un sujeto diagnosticado con Párkinson. En este proyecto el conjunto de datos contiene un tercer diccionario que almacena todas las instancias de la clase recorte. Esto permite que, una vez realizada una predicción sobre el conjunto de datos, se pueda asociar una predicción a cada uno de los puntos de coordenadas que forman parte de cada recorte.

5.4. *CNN* usada en el proyecto

En la **Figura 5.18** se muestra la *CNN* usada en este proyecto. Se trata de una red neuronal convolucional formada por dos capas convolucionales, dos capas *max pooling* y tres capas completamente conectadas con la siguiente estructura:

1. **Capa convolucional:** Capa con un canal de entrada (el canal de color de una imagen en escala de grises) y 6 canales de salida. Usa un *kernel* de 3✕3 con *stride* 1 y sin *padding*.
2. **Función de activación:** *ReLU (Rectified Linear Unit)*. Se transmiten los valores positivos y los negativos pasan a ser cero. **ReLU(x) = max(0, x)**
3. **Capa *max pooling*:** Usa *kernel* y *stride* de tamaño 2. Por lo tanto la dimensión de la salida será la mitad que la de la entrada.

4. **Capa convolucional:** Capa con seis canales de entrada (los canales de salida de la primera capa convolucional) y 16 canales de salida. Al igual que la primera capa usa un *kernel* de 3×3 con *stride* 1 y sin *padding*.
5. **Función de activación:** *ReLU*.
6. **Capa *max pooling*:** Usa *kernel* y *stride* de tamaño 2. Por lo tanto la dimensión de la salida será la mitad que la de la entrada.
7. ***Flattening* de los datos:** Es un paso necesario para las capas densas. Para calcular el tamaño del *array* se tienen en cuenta los 16 canales de salida de la segunda capa convolucional y el tamaño del lado de las imágenes usadas. En concreto el tamaño del array se calcula mediante la siguiente operación:
    - Lado original: A modo de ejemplo las imágenes de entrada tendrán lado 71.
    - Capa convolucional: Al no usar *padding* el tamaño del lado pasa a ser 69.
    - Capa *max pooling*: El tamaño pasa a ser la mitad. En este caso, al ser la entrada impar, la salida será la mitad redondeando hacia abajo [17]. Por lo tanto el lado de la salida será 34.
    - Capa convolucional: Al no usar *padding* el tamaño del lado pasa a ser 32.
    - Capa *max pooling*: El tamaño pasa a ser la mitad, 16.
    - El *flattening* tendrá un tamaño de 16x16 con 16 canales ($16 \times 16 \times 16$).

    Por lo tanto la operación para calcular el tamaño del *flattening* es:

$$((floor((lado\ -\ 2)\ /\ 2)\ -\ 2)\ //\ 2)^2 \times 16$$

8. **Capa completamente conectada:** Capa con tamaño de entrada definido en el anterior punto y tamaño de salida 120.
9. **Función de activación:** *ReLU*.
10. **Capa completamente conectada:** Capa con tamaño de entrada 120 y tamaño de salida 84.
11. **Función de activación:** *ReLU*.
12. **Capa completamente conectada:** Capa con tamaño de entrada 84 y tamaño de salida 2 (H o PD).

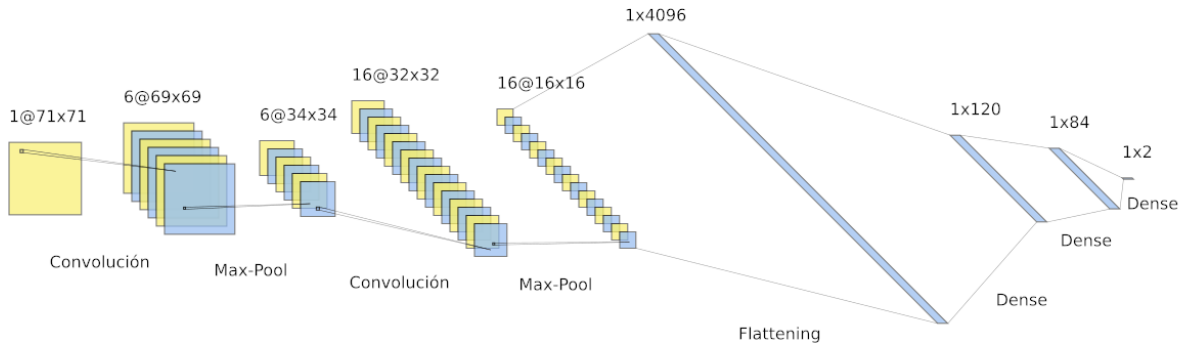

Fig. 5.18 Esquema de la red neuronal convolucional usada en este proyecto para un tamaño de recorte 71.

5.4.1. Datos adicionales sobre la *CNN*

- ***Kernel* (capas convolucionales):** En este proyecto se hace uso del método estándar que ofrece *PyTorch* para inicializar el *kernel* de convolución [16].
- **Función de pérdida:** Como función de pérdida se usa *CrossEntropyLoss*, que es equivalente a combinar *LogSoftMax* y *NLLLoss*. Reducir la *Cross entropy* (Entropía cruzada) de la salida de un modelo de *CNN* es equivalente a mejorar la corrección de las predicciones que genera el modelo [15].
- ***Epochs*:** En este proyecto se han utilizado 30 *epochs* en todos los casos.
- ***Batch size*:** El *batch size* usado en este proyecto ha sido 10.

5.5. Generación y tratamiento de las predicciones

Para ejemplificar la generación de las predicciones y su tratamiento se ha realizado el entrenamiento de la *CNN* usando tamaño de recorte 71 y todos los sujetos para entrenamiento y test con división mostrada en la **Tabla 5.1**.

Sujetos para entrenamiento	Sujetos para test
62 (82,67%).	13 (17,33%)

Tabla 5.1 División para entrenamiento y test usando todos los sujetos de PaHaW.

Otros valores a tener en cuenta son los *epochs* (30) y el *batch size* (10).

Epoch	*Train loss*	*Train accuracy*	*Test loss*	*Test accuracy*
5	0,4161	77,82%	1,2278	52,18%
10	0,2591	87,21%	2,0774	52,32%
15	0,1730	92,01%	3,3253	51,46%
20	0,1236	94,53%	3,6210	51,82%
25	0,0966	95,92%	5,1317	50,73%
30	0,0796	96,81%	5,0581	52,00%

Tabla 5.2 Entrenamiento de la *CNN* usando 62 sujetos con tamaño de recorte 71.

En la **Tabla 5.2** se observa una corrección para los recortes del grupo de test cercana al 50%, pero el objetivo es realizar las predicciones sobre las tareas, las letras y los trazos, no sobre los recortes. Aunque el propio entrenamiento ofrece datos sobre la corrección, para saber la corrección real es necesario procesar las predicciones realizadas a nivel de recorte para transformarlas en predicciones a nivel de fragmentos de trazos, letras y tareas.

Cada letra tiene un diccionario en el que la clave es cada una de las coordenadas que la forman y el valor una lista de los recortes que contienen dicha coordenada. Esto permite generar una predicción sobre cada una de las coordenadas de cada letra. Para ello se consultan las predicciones de cada uno de los recortes que la contienen y se decide la predicción de la coordenada en base a la predicción dominante. Si la coordenada está presente en un número par de recortes y las predicciones están equilibradas (esto es, hay el mismo número de predicciones H y PD) la predicción de esa coordenada se decide de manera aleatoria.

Una vez se ha calculado la predicción de todas las coordenadas que forman una letra se procede a generar los trazos de predicción. En estos trazos todas sus coordenadas tienen la misma predicción. Por último, queda calcular la predicción de cada letra y de cada tarea, que dependerá de la predicción de cada uno de los trazos y de su longitud (en términos de

distancia, no de cantidad de puntos que lo forman). En la **Figura 5.19** se muestra la predicción final tras todo el proceso explicado.

Fig. 5.19 Predicción sobre los trazos, las letras y la tarea del sujeto 53.

En el resultado final se pueden observar los trazos generados tras la predicción, mostrándose de color verde los trazos que se consideran H y de color rojo los trazos que se consideran PD. Bajo cada letra se presenta su predicción, H o PD, con los colores ya mencionados. Por último, en el título de la imagen se muestra la predicción de la tarea.

6. Pruebas y experimentos

En este apartado se mostrarán los resultados de diferentes pruebas realizadas con el objetivo de intentar mejorar los resultados de las predicciones. Para ello es necesario explicar los siguientes términos:

- **Etiqueta negativa:** La etiqueta negativa es H (sano).
- **Etiqueta positiva:** La etiqueta positiva es PD (*Parkinson's Disease*).
- **Verdadero positivo (TP):** La predicción y la etiqueta son positivas.
- **Falso positivo (FP):** La predicción es positiva pero la etiqueta es negativa.
- **Verdadero negativo (TN):** La predicción y la etiqueta son negativas.
- **Falso negativo (FN):** La predicción es negativa pero la etiqueta es positiva.

Accuracy (Corrección)

$$Corrección = Predicciones\ correctas\ /\ Predicciones\ totales$$
$$Corrección = (TP + TN)\ /\ (TP + FP + TN + FN)$$

Recall (Exhaustividad)

$$Recall = Verdaderos\ positivos\ /\ Positivos\ reales$$
$$Recall = TP\ /\ (TP + FN)$$

El problema del *recall* es que solamente tiene en cuenta las predicciones positivas. El *recall* de una *CNN* que predice todos los casos de forma positiva es el mismo que el de una *CNN* perfecta.

Precision (Precisión)

$$Precisión = Verdaderos\ positivos\ /\ Predicciones\ positivas$$
$$Precisión = TP\ /\ (TP + FP)$$

F1 Score (Valor-F)

$$F1\ Score\ =\ 2 \times (Recall \times Precisión)\ /\ (Recall\ +\ Precisión)$$

También se hará uso de *matrices de confusión* para poder exponer de forma sencilla los verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos. Cada columna de la matriz representa las instancias de una clase real mientras que cada fila representa las predicciones de cada clase.

Para realizar todos los experimentos de este apartado se han usado 30 *epochs* con *batch size* 10 para entrenar la *CNN*. Todos los conjuntos de entrenamiento están balanceados, incluyen el mismo número de sujetos de las dos clases (H y PD).

6.1. Experimento 1: Búsqueda del tamaño de recorte óptimo

Como se mostró en el **Apartado 5.2.2** el tamaño mínimo de recorte para poder generar trazos es 71, pero es posible que un tamaño de recorte mayor mejore los resultados de las predicciones. Para realizar la siguiente prueba se han utilizado todos los sujetos de la base de datos PaHaW con la división mostrada en la **Tabla 5.1**.

Para saber cómo afecta el tamaño del recorte a la calidad de las predicciones se entrena la *CNN* y se procesan las predicciones con los siguientes tamaños: 71, 91, 111, 131, 151. De esta manera se intenta buscar alguna tendencia a mejor o a peor o un valor que ofrezca mejores resultados. Las predicciones se realizan únicamente sobre el grupo de test.

6.1.1. Tamaño de recorte 71

Epoch	*Train loss*	*Train accuracy*	*Test loss*	*Test accuracy*
5	0,4161	77,82%	1,2278	52,18%
10	0,2591	87,21%	2,0774	52,32%
15	0,1730	92,01%	3,3253	51,46%
20	0,1236	94,53%	3,6210	51,82%
25	0,0966	95,92%	5,1317	50,73%
30	0,0796	96,81%	5,0581	52,00%

Tabla 6.1 Entrenamiento de la *CNN* con tamaño de recorte 71.

La corrección mostrada en las **Tablas 6.1 y 6.2** se acerca al 50% en algunos casos, como en las predicciones realizadas a nivel de letra. En la matriz de confusión de la **Figura 6.1** asociada a las letras puede observarse que para la clase PD la corrección de las predicciones es del 50%.

	Tareas	**Letras**
Corrección	61,54%	53,85%
Precisión	57,14%	50,00%
Recall	66,67%	63,33%
F1 Score	61,54%	55,88%

Tabla 6.2 Valores de la predicción con tamaño de recorte 71.

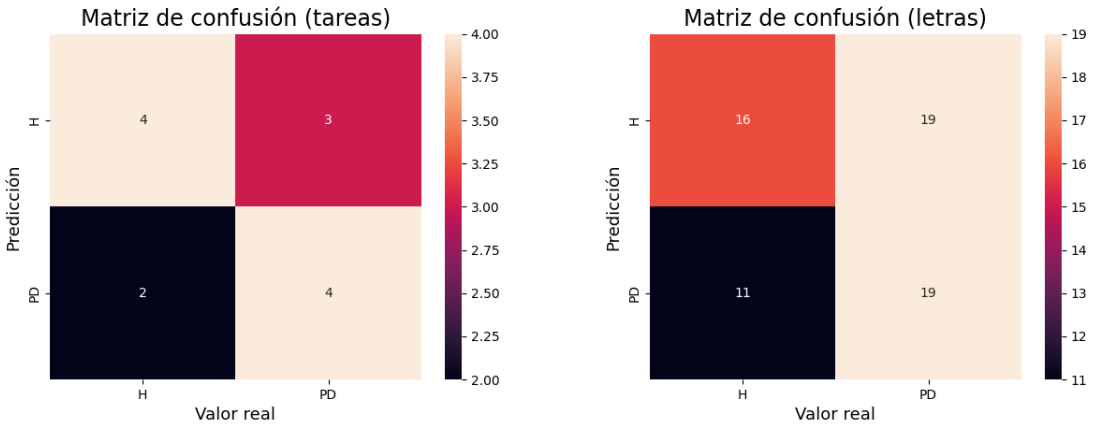

Fig. 6.1 Matrices de confusión con tamaño de recorte 71.

6.1.2. Tamaño de recorte 91

Epoch	*Train loss*	*Train accuracy*	*Test loss*	*Test accuracy*
5	0,3633	81,62%	1,2095	52,18%
10	0,1960	91,01%	2,5793	51,86%
15	0,1185	94,92%	3,8307	52,16%
20	0,0770	96,93%	5,1073	51,44%
25	0,0570	97,76%	5,8809	51,07%
30	0,0439	98,37%	6,6928	50,79%

Tabla 6.3 Entrenamiento de la *CNN* con tamaño de recorte 91.

En este caso en las **Tablas 6.3 y 6.4** se observa que la corrección de las predicciones a nivel de tarea está por debajo del 50%, con valores como el *recall* y el *F1 Score* inferiores al 40%. En la matriz de confusión de la **Figura 6.2** asociada a las tareas se muestra que la corrección de las predicciones para la clase H es del 50% y para la clase PD es del 40%

	Tareas	Letras
Corrección	46,15%	58,46%
Precisión	40,00%	55,56%
Recall	33,33%	50,00%
F1 Score	36,36%	52,63%

Tabla 6.4 Valores de la predicción con tamaño de recorte 91.

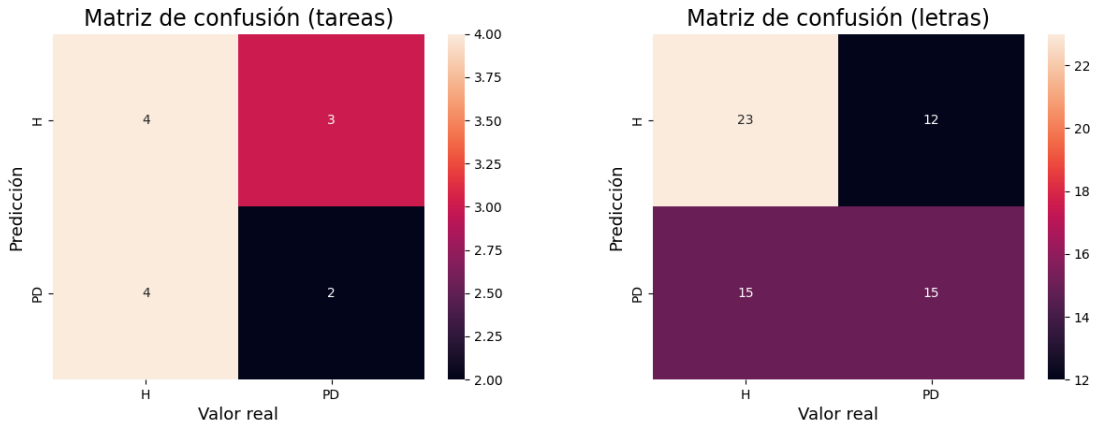

Fig. 6.2 Matrices de confusión con tamaño de recorte 91.

6.1.3. Tamaño de recorte 111

Epoch	*Train loss*	*Train accuracy*	*Test loss*	*Test accuracy*
5	0,2211	89,92%	1,8742	52,46%
10	0,0795	96,72%	3,7517	53,33%
15	0,0427	98,47%	5,2080	52,17%
20	0,0260	99,07%	6,1243	52,85%
25	0,0185	99,37%	6,4278	52,78%
30	0,0141	99,58%	7,3158	52,40%

Tabla 6.5 Entrenamiento de la *CNN* con tamaño de recorte 111.

En las **Tablas 6.5 y 6.6** se muestran los mejores resultados obtenidos hasta el momento, siendo la corrección de las predicciones a nivel de tarea cercanas al 70% y a nivel de letra superiores al 60%. En las matrices de confusión de la **Figura 6.3** queda claro el buen resultado de las predicciones.

	Tareas	**Letras**
Corrección	69,23%	61,54%
Precisión	66,67%	58,06%
Recall	66,67%	60,00%
F1 Score	66,67%	59,02%

Tabla 6.6 Valores de la predicción con tamaño de recorte 111.

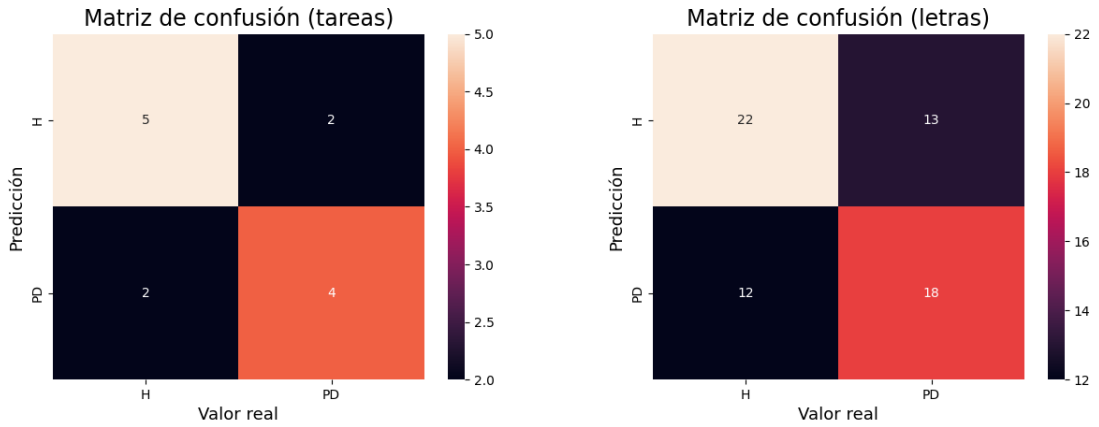

Fig. 6.3 Matrices de confusión con tamaño de recorte 111.

6.1.4. Tamaño de recorte 131

Epoch	*Train loss*	*Train accuracy*	*Test loss*	*Test accuracy*
5	0,1570	93,30%	1,8684	54,44%
10	0,0398	98,59%	3,7999	55,31%
15	0,0189	99,38%	5,1736	54,35%
20	0,0120	99,65%	5,6817	55,14%
25	0,0095	99,70%	7,1166	53,80%
30	0,0079	99,79%	7,1774	54,65%

Tabla 6.7 Entrenamiento de la *CNN* con tamaño de recorte 131.

En los resultados mostrados en las **Tablas 6.7 y 6.8** se descarta la existencia de una tendencia positiva a medida que se aumenta el tamaño de recorte. En este caso los resultados son inferiores a los obtenidos anteriormente, siendo la corrección de las predicciones para la clase PD a nivel de letra cercana al 50%, como puede verse en la **Figura 6.4**.

	Tareas	**Letras**
Corrección	61,54%	58,46%
Precisión	57,14%	54,05%
Recall	66,67%	66,67%
F1 Score	61,54%	59,70%

Tabla 6.8 Valores de la predicción con tamaño de recorte 131.

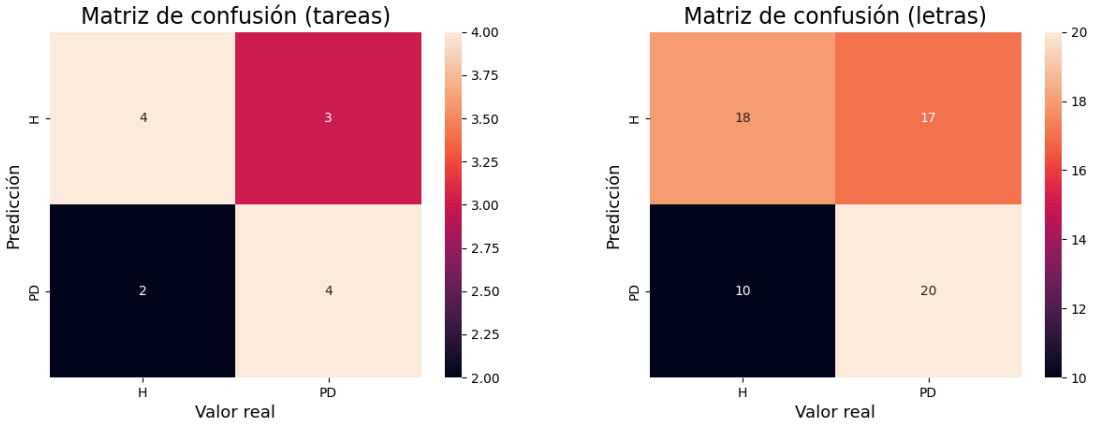

Fig. 6.4 Matrices de confusión con tamaño de recorte 131.

6.1.5. Tamaño de recorte 151

Epoch	*Train loss*	*Train accuracy*	*Test loss*	*Test accuracy*
5	0,0985	96,02%	2,3076	53,61%
10	0,0200	99,33%	4,3869	52,40%
15	0,0104	99,67%	6,1524	53,52%
20	0,0076	99,76%	6,8348	52,13%
25	0,0058	99,83%	6,9353	52,12%
30	0,0050	99,87%	7,4377	52,52%

Tabla 6.9 Entrenamiento de la *CNN* con tamaño de recorte 151.

Teniendo en cuenta los resultados de la prueba anterior y los mostrados en las **Tablas 6.9 y 6.10** queda claro que no hay una tendencia positiva a medida que se aumenta el tamaño de recorte. Como puede observarse en las matrices de confusión de la **Figura 6.5**, la corrección es cercana al 60% tanto a nivel de letra como de tarea.

	Tareas	Letras
Corrección	61,54%	58,46%
Precisión	57,14%	54,29%
Recall	66,67%	63,33%
F1 Score	61,54%	58,46%

Tabla 6.10 Valores de la predicción con tamaño de recorte 151.

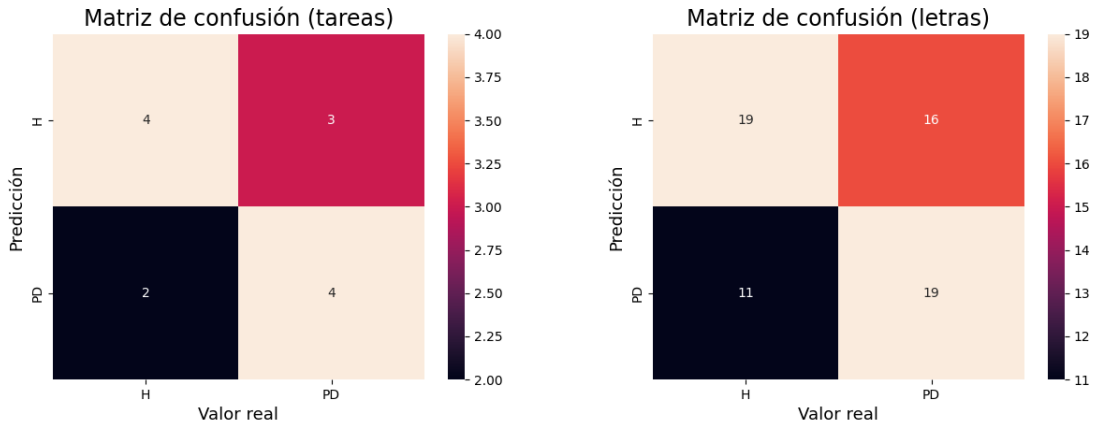

Fig. 6.5 Matrices de confusión con tamaño de recorte 151.

Como resumen, puede concluirse que tras realizar pruebas con varios tamaños no se observa una tendencia clara. De los tamaños probados el que mejores resultados ha ofrecido es 111, por lo que será este tamaño de recorte el usado de ahora en adelante.

6.2. Experimento 2: Configuración de sujetos para entrenar la *CNN*

En el apartado anterior se concluyó que el tamaño de los recortes usados para entrenar la *CNN* y realizar las predicciones sería 111. Ahora queda por decidir qué criterio seguir a la hora de seleccionar los sujetos utilizados para entrenar la misma. En esta sección se van a realizar una serie de pruebas seleccionando los sujetos para entrenar la *CNN* siguiendo diferentes criterios que se explicarán en cada subsección.

6.2.1. Primera configuración: Considerar todos los sujetos de la base de datos PaHaW

Los resultados de las predicciones usando el tamaño de recorte 111 y todos los sujetos para entrenamiento (62) y test (13) se mostraron en el **Apartado 6.1.3**. Aunque son resultados que superan el objetivo del 60% marcado al inicio de este documento, se han hecho otras pruebas para ver si es posible mejorarlos.

6.2.2. Segunda configuración: Considerar todos los sujetos de la base de datos PaHaW seleccionando previamente 16 sujetos en base a sus características PD visibles

Analizando las tareas de todos los sujetos se han seleccionado 8 sujetos H sin características PD visibles y 8 sujetos PD con las características PD más visibles. En la **Figura 6.6** se muestra una tarea sin características PD visibles.

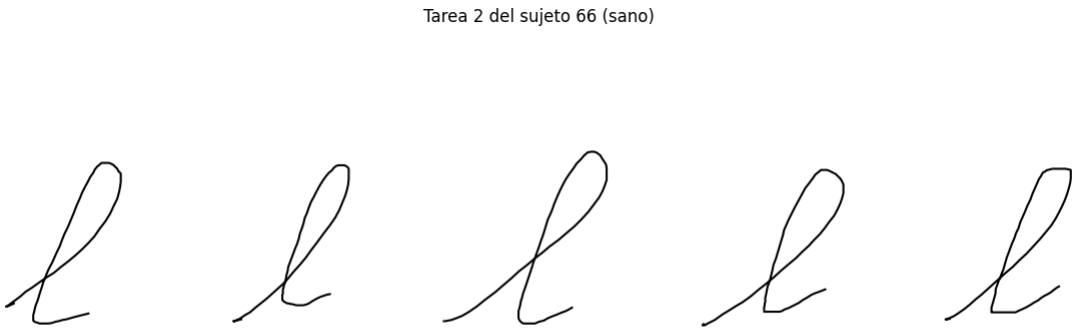

Fig. 6.6 Tarea 2 del sujeto 66. No se aprecian características PD visibles.

Las características PD visibles se han definido como temblores y otras alteraciones respecto a un movimiento suave y armónico que son visibles a simple vista en el trazo. La **Figura 6.7** contiene una tarea con características PD visibles.

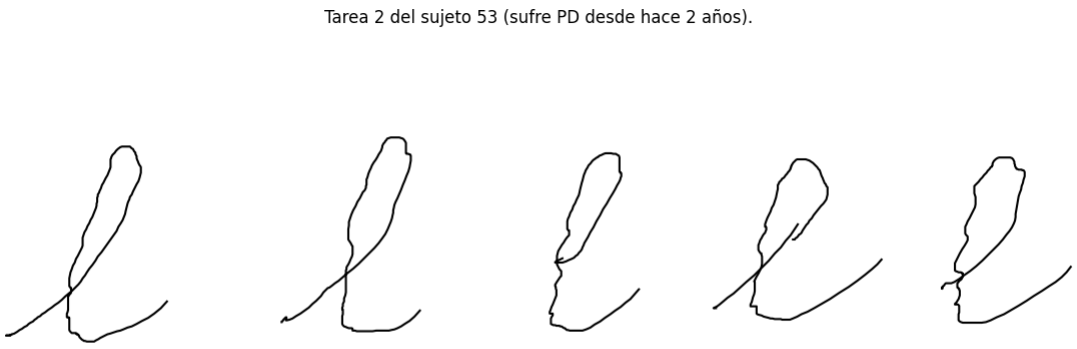

Fig. 6.7 Tarea 2 del sujeto 53. Hay presencia de características PD visibles.

En base a lo explicado se han seleccionado los sujetos listados en la **Tabla 6.11**.

Sujetos H	30, 49, 52, 67, 69, 70, 84, 87
Sujetos PD	4, 7, 9, 17, 18, 19, 53, 55

Tabla 6.11 Sujetos seleccionados en base a sus características PD visibles.

Para generar los conjuntos de entrenamiento (62) y test (13) se siguen los siguientes pasos:

1. La mitad de los sujetos H de la Tabla 6.11 se usan para entrenamiento y la otra mitad para test.
2. La mitad de los sujetos PD de la Tabla 6.11 se usan para entrenamiento y la otra mitad para test.
3. De los sujetos restantes, 5 se usan para test, con lo que se completa el conjunto de 13 sujetos para test.
4. El resto de sujetos se usan para entrenamiento, siendo el tamaño del grupo de entrenamiento de 62 sujetos finalmente.

En la **Tabla 6.12** se muestran los datos de entrenamiento de la *CNN* y en la **Tabla 6.13** los resultados de las predicciones generadas. La **Figura 6.8** presenta la matriz de confusión asociada a la **Tabla 6.13**.

Epoch	*Train loss*	*Train accuracy*	*Test loss*	*Test accuracy*
5	0,2091	90,55%	3,6077	47,75%
10	0,0603	97,70%	7,4966	47,48%
15	0,0275	99,03%	9,8702	48,00%
20	0,0146	99,56%	11,6695	48,21%
25	0,0113	99,66%	12,4524	48,93%
30	0,0107	99,71%	14,2993	47,94%

Tabla 6.12 Entrenamiento de la *CNN* usando 62 sujetos para entrenamiento y 13 para test. Se han seleccionado previamente 16 sujetos en base a sus características PD visibles.

	Tareas	**Letras**
Corrección	61,54%	52,31%
Precisión	57,14%	48,78%
Recall	66,67%	66,67%
F1 Score	61,54%	56,34%

Tabla 6.13 Valores de la predicción sobre los 13 sujetos de test.

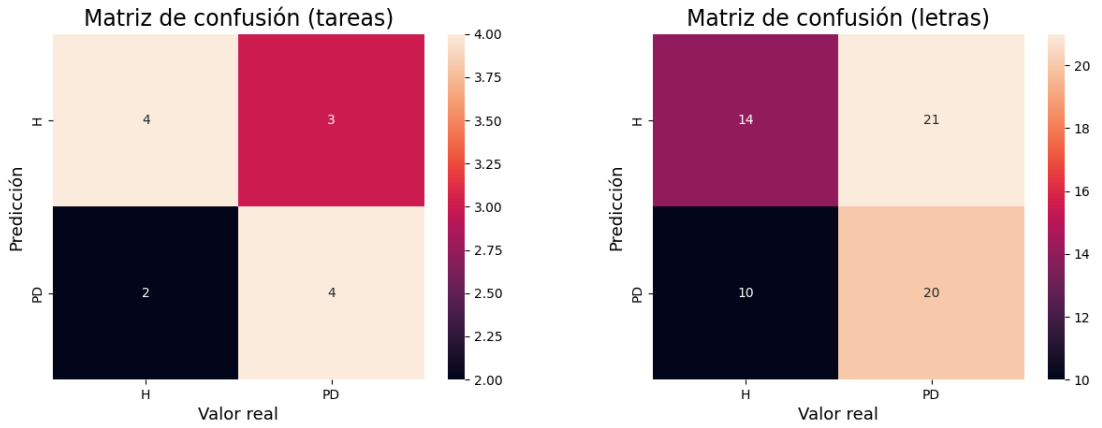

Fig. 6.8 Matrices de confusión de las predicciones realizadas sobre los 13 sujetos de test.

	Tareas (62 sujetos)	Tareas (62 sujetos con 16 seleccionados)
Corrección	69,23%	61,54%
Precisión	66,67%	57,14%
Recall	66,67%	66,67%
F1 Score	66,67%	61,54%

Tabla 6.14 Comparativa de las predicciones generadas a nivel de tarea sobre los 13 sujetos de test respecto a las predicciones generadas en la prueba anterior.

	Letras (62 sujetos)	Letras (62 sujetos con 16 seleccionados)
Corrección	61,54%	52,31%
Precisión	58,06%	48,78%
Recall	60,00%	66,67%
F1 Score	59,02%	56,34%

Tabla 6.15 Comparativa de las predicciones generadas a nivel de letra sobre los 13 sujetos de test respecto a las predicciones generadas en la prueba anterior.

Analizando las comparativas mostradas en las **Tablas 6.14** y **6.15**, parece claro que es mejor utilizar únicamente las etiquetas que ofrece la propia base de datos PaHaW para formar los grupos de entrenamiento y test.

6.2.3. Tercera configuración: Uso de los 16 sujetos seleccionados

El siguiente paso es usar únicamente los sujetos mostrados en la **Tabla 6.11** para entrenar la *CNN*. Para ello se dividen los 16 sujetos como se indica en la **Tabla 6.16.**

Sujetos para entrenamiento	Sujetos para test
14 (87,50%).	2 (12,50%)

Tabla 6.16 División para entrenamiento y test usando los 16 sujetos seleccionados.

En la **Tabla 6.17** se muestran los resultados de entrenamiento de la *CNN* y en la **Tabla 6.18** los resultados de las predicciones generadas. En la **Figura 6.9** se presenta la matriz de confusión asociada a las predicciones de la **Tabla 6.18**.

Epoch	*Train loss*	*Train accuracy*	*Test loss*	*Test accuracy*
5	0,0680	97,34%	0,9830	78,78%
10	0,0193	99,39%	1,4741	79,71%
15	0,0127	99,60%	1,9624	76,47%
20	0,0085	99,74%	2,0680	77,03%
25	0,0036	99,88%	2,5143	74,77%
30	0,0047	99,90%	2,8926	74,24%

Tabla 6.17 Entrenamiento de la *CNN* usando los 16 sujetos seleccionados.

En este caso las predicciones se realizan sobre los 61 sujetos que no se han usado para entrenamiento. Esto incluye los 2 sujetos que se han usado para test y los 59 sujetos restantes.

	Tareas	Letras
Corrección	52,46%	52,77%
Precisión	52,17%	52,10%
Recall	40,00%	41,33%
F1 Score	45,28%	46,10%

Tabla 6.18 Valores de la predicción sobre los 59 sujetos restantes.

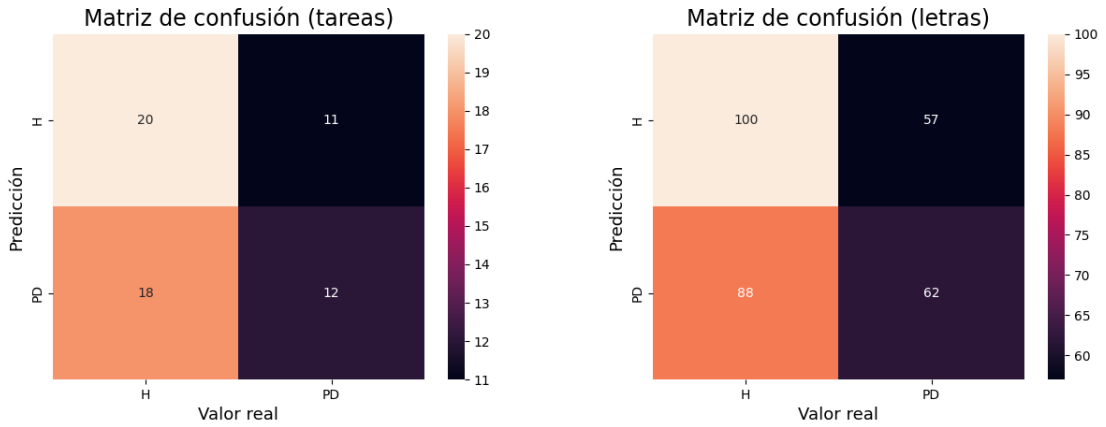

Fig. 6.9 Matrices de confusión de las predicciones realizadas sobre los 59 sujetos restantes.

	Tareas (62 sujetos con 16 seleccionados)	Tareas (16 sujetos seleccionados)
Corrección	61,54%	52,46%
Precisión	57,14%	52,17%
Recall	66,67%	40,00%
F1 Score	61,54%	45,28%

Tabla 6.19 Comparativa de las predicciones generadas a nivel de tarea respecto a las predicciones generadas en la prueba anterior.

	Letras (62 sujetos con 16 seleccionados)	Letras (16 sujetos seleccionados)
Corrección	52,31%	52,77%
Precisión	48,78%	52,10%
Recall	66,67%	41,33%
F1 Score	56,34%	46,10%

Tabla 6.20 Comparativa de la predicción a nivel de letra usando solo los seleccionados o los 62 con los seleccionados.

En las **Tablas 6.19 y 6.20** se puede observar que la corrección de las predicciones empeora cuando se usan únicamente los 16 sujetos seleccionados en base a sus características PD visibles para entrenar la *CNN*. Hasta ahora se han obtenido los mejores resultados generando los grupos de entrenamiento y test usando únicamente el estado PD ofrecido por PaHaW.

6.2.4. Cuarta configuración: Uso de los 16 sujetos seleccionados descartando valores atípicos

El problema de entrenar una *CNN* para reconocer recortes con características PD visibles es que eso es lo que hará. Atendiendo a los resultados de las predicciones realizadas sobre las tareas, se han observado una serie de sujetos que no cumplen las características que espera la *CNN*.

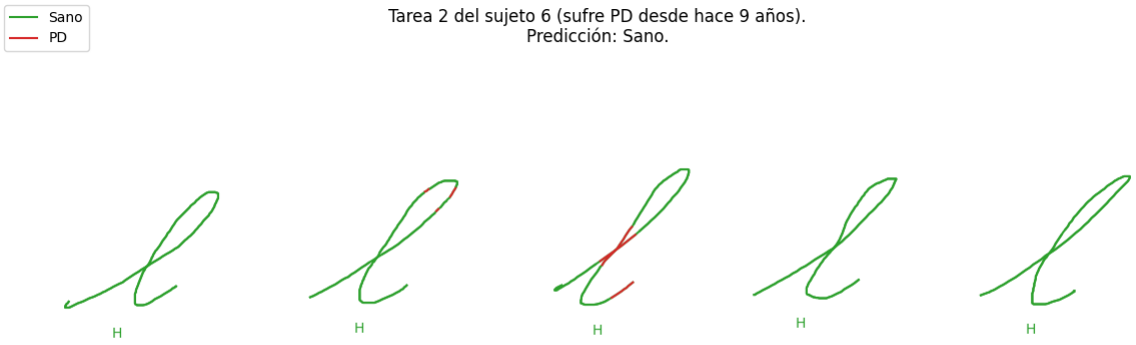

Fig. 6.10 Valor atípico en la detección. Falso negativo.

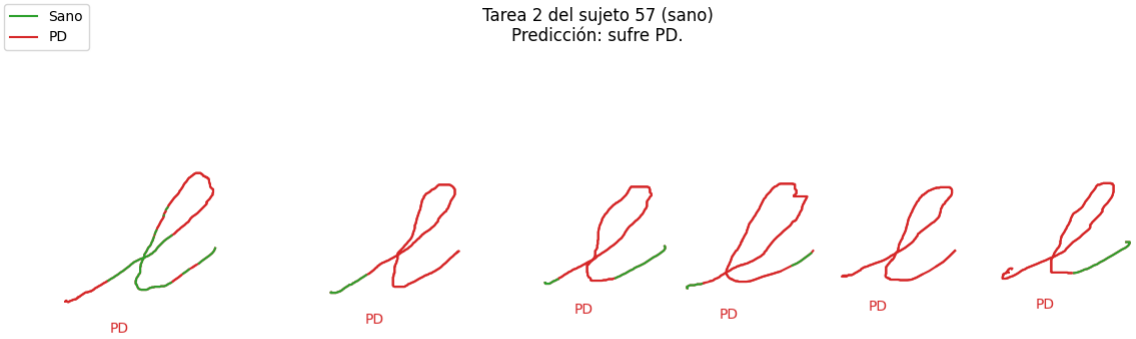

Fig. 6.11 Valor atípico en la detección. Falso positivo.

Como valores atípicos se han considerado todas las tareas en las que tanto la predicción a nivel de tarea como a nivel de cada una de sus letras es incorrecta. En la **Figura 6.10** se muestra un caso de falso negativo y en la **Figura 6.11** uno de falso positivo. Cabe destacar que, tal y como se muestra en la **Tabla 6.21**, la presencia de falsos negativos dobla a la de falsos positivos. Esto podría deberse, entre otros factores, a la medicación asignada a los sujetos PD. [3]

Falsos negativos	20, 24, 33, 43, 44, 74, 75, 78
Falsos positivos	28, 57, 83, 91

Tabla 6.21 Valores atípicos usando los sujetos seleccionados para entrenar la *CNN*.

En esta prueba se entrena la *CNN* con los mismos sujetos que en la prueba anterior, por lo que se han omitido los resultados (pueden consultarse en la **Tabla 6.17**).

En la prueba anterior se llevaron a cabo las predicciones sobre 59 sujetos, en este caso de esos 59 se descartan los 12 sujetos de la **Tabla 6.21**, quedando 47 sujetos en total.

	Tareas	Letras
Corrección	62,26%	61,05%
Precisión	54,55%	52,54%
Recall	54,55%	56,36%
F1 Score	54,55%	54,39%

Tabla 6.22 Valores de la predicción sobre los 47 sujetos restantes tras descartar los valores atípicos.

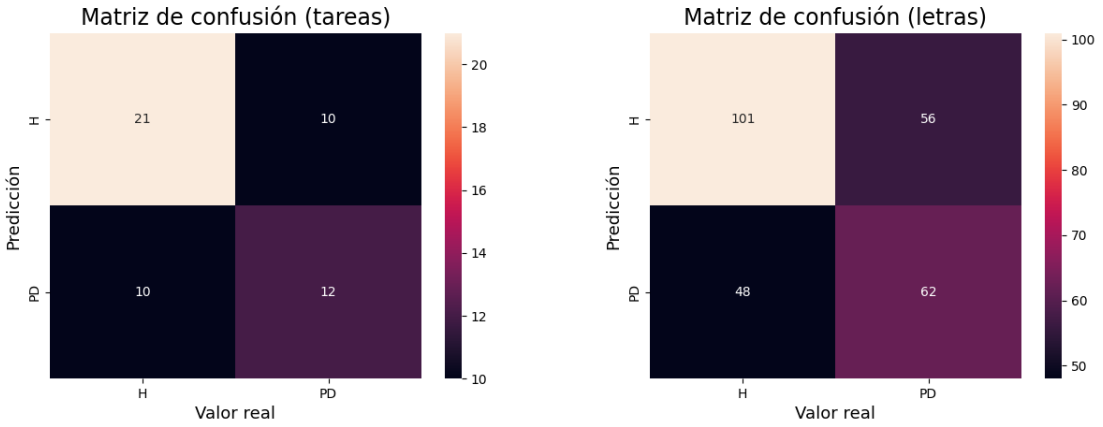

Fig. 6.12 Matrices de confusión de las predicciones realizadas sobre los 47 sujetos restantes tras descartar los valores atípicos.

	Tareas (16 sujetos seleccionados)	Tareas (descartando valores atípicos)
Corrección	52,46%	62,26%
Precisión	52,17%	54,55%
Recall	40,00%	54,55%
F1 Score	45,28%	54,55%

Tabla 6.23 Comparativa de las predicciones a nivel de tarea para 59 sujetos y para 47 (tras descartar los valores atípicos).

	Letras (16 sujetos seleccionados)	Letras (descartando valores atípicos)
Corrección	52,77%	61,05%
Precisión	52,10%	52,54%
Recall	41,33%	56,36%
F1 Score	46,10%	54,39%

Tabla 6.24 Comparativa de las predicciones a nivel de letra para 59 sujetos y para 47 (tras descartar los valores atípicos).

Aunque en un principio parecía un camino a seguir, observando los resultados de la **Tabla 6.22** y la **Figura 6.12** y teniendo en cuenta todas las comparativas mostradas a lo largo de este apartado, incluyendo las mostradas en las **Tablas 6.23 y 6.24**, se puede concluir que la mejor opción a la hora de formar los grupos de entrenamiento y test es usar únicamente el estado PD que ofrece la base de datos PaHaW sobre cada sujeto. Elegir los sujetos en base a sus características PD visibles para entrenar la *CNN* produce peores resultados, llegando a caer la corrección a valores cercanos al 50%.

7. Conclusión

Para finalizar se realiza un análisis de las partes más importantes del proyecto, de los objetivos iniciales y de los resultados obtenidos de las diferentes pruebas realizadas.

7.1. Conclusiones generales

Tomando en consideración el trabajo realizado cabe afirmar que los objetivos principales mencionados en la introducción se han cumplido. A partir de la base de datos PaHaW se ha creado un conjunto de datos para entrenar la *CNN* que permite realizar predicciones a nivel de trazo, letra y tarea con una corrección superior a la del objetivo marcado.

A lo largo del proyecto se han realizado varias pruebas de las que se puede extraer algunas conclusiones relevantes. Se ha observado que para entrenar la *CNN* y realizar las predicciones el tamaño de recorte debe ser mayor de 71, y en base a las pruebas realizadas, el tamaño de recorte óptimo se encuentra en torno a 111. También se ha observado que, contrariamente a lo que la intuición pudiera sugerir, el resultado de las predicciones no mejora seleccionando previamente una serie de sujetos en base a las características PD visibles, generando casos en los que la predicción es incorrecta tanto a nivel de tarea como a nivel de todas y cada una de las letras que la componen al crear un clasificador que se centra únicamente en los casos más extremos tanto a nivel de alteraciones como de falta de ellas.

Aunque este documento se ha centrado en la Tarea 2 de las ofrecidas por PaHaW, en los anexos finales se incluyen una serie de pruebas que sugieren que los resultados de las predicciones obtenidas siguiendo la metodología explicada podrían resultar mejores para las Tareas 3 y 4.

7.2. Trabajo futuro

Aunque se hayan cumplido los objetivos marcados al inicio del proyecto, sobre esta base se podrían aplicar varias mejoras y variaciones.

En primer lugar, este proyecto se centra en una sola tarea de las 8 que ofrece PaHaW. Sería interesante ampliar la esfera de acción para incluir otras tareas, en especial la tarea 1 en la que los sujetos dibujan una espiral, al ser la única tarea de las que ofrece PaHaW que no se centra en escribir letras y palabras. Por otra parte, para realizar las predicciones se ha usado un único modelo de *CNN* básico. Dada la estructura del proyecto debería resultar sencillo aplicar otro tipo de redes neuronales profundas al mismo.

A la hora de seleccionar los sujetos para entrenamiento y test se han utilizado dos criterios (estado PD del sujeto y características PD visibles). Se podría ampliar este trabajo usando otras aproximaciones, como por ejemplo los años que cada persona lleva padeciendo Párkinson o el valor *UPDRS V* (escala usada a la hora de calcular la severidad de la enfermedad de Parkinson).

Otra posible mejora a seguir sería la creación de una aplicación con el trabajo de este proyecto como base, de forma que un sujeto pudiese realizar las tareas en una tableta y obtuviese una predicción en tiempo real.

Referencias

[1] Drotár, Peter, et al. “Analysis of in-air movement in handwriting: A novel marker for Parkinson’s disease.” *Computer Methods and Programs in Biomedicine*, vol. 117, no. 3, 2014, pp. 405-411.

[2] Drotár, Peter, et al. “Evaluation of handwriting kinematics and pressure for differential diagnosis of Parkinson's disease.” *Artificial Intelligence in Medicine*, vol. 67, 2016, pp. 39-46.

[3] Eichhorn TE, et al. “Computational analysis of open loop handwriting movements in Parkinson's disease: a rapid method to detect dopamimetic effects.” *Movement disorders : official journal of the Movement Disorder Society*, vol. 11, no. 3, 1996, pp. 289-97.

[4] Impedovo, Donato, and Giuseppe Pirlo. “Dynamic Handwriting Analysis for the Assessment of Neurodegenerative Diseases: A Pattern Recognition Perspective.” *IEEE Reviews in Biomedical Engineering*, vol. 12, 2019, pp. 209-220.

[5] Legarda Ramírez, Inés. “Síntomas Motores - Conoce el Parkinson.” *Conoce el Párkinson*, 1 de marzo de 2019, https://conoceelparkinson.org/sintomas-motores-del-parkinson/. Accedido el 30 de junio 2023.

[6] Nurfikri, Fahmi. “An Illustrated Guide to Artificial Neural Networks.” *Towards Data Science*, 20 de julio de 2020, https://towardsdatascience.com/an-illustrated-guide-to-artificial-neural-networks-f149a549ba74. Accedido el 2 de julio de 2023.

[7] Organización Mundial de la Salud. *Enfermedad de Parkinson*, 13 de junio de 2022, https://who.int/es/news-room/fact-sheets/detail/parkinson-disease. Accedido el 30 de junio de 2023.

[8] Sharma, Priyanshi. “Everything about Pooling layers and different types of Pooling.” *LaptrinhX*, 30 de abril de 2020, https://laptrinhx.com/news/everything-about-pooling-layers-and-different-types-of-pooling-WM7YxrQ/. Accedido el 3 de julio de 2023.

[9] Unzueta, Diego. “Fully Connected Layer vs Convolutional Layer: Explained.” *Built In*, 18 de octubre de 2022, https://builtin.com/machine-learning/fully-connected-layer. Accedido el 3 de julio de 2023.

[10] Visin, Francesco. “A technical report on convolution arithmetic in the context of deep learning.” *GitHub*, 2016, https://github.com/vdumoulin/conv_arithmetic. Accedido el 2 de julio de 2023.

[11] IBM. “What are Convolutional Neural Networks?” *IBM*, https://ibm.com/topics/convolutional-neural-networks. Accedido el 5 de julio de 2023.

[12] IBM. “What are Neural Networks?” *IBM*, https://ibm.com/topics/neural-networks. Accedido el 5 de julio de 2023.

[13] IBM. “What is Machine Learning?” *IBM*, https://ibm.com/topics/machine-learning. Accedido el 5 de julio de 2023.

[14] Brownlee, Jason. “Difference Between a Batch and an Epoch in a Neural Network - MachineLearningMastery.com.” *Machine Learning Mastery*, 10 de agosto de 2022, https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. Accedido el 10 de julio de 2023.

[15] Flaherty, Jean A. “The Benefits of Cross Entropy Loss.” *ML Review*, 22 de julio de 2019, https://kobejean.github.io/machine-learning/2019/07/22/cross-entropy-loss/. Accedido el 10 de julio de 2023.

[16] He, Kaiming, et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.” *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026-1034.

[17] PyTorch Contributors. “MaxPool2d — PyTorch 2.0 documentation.” *PyTorch*, https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html#torch.nn.MaxPool2d. Accedido el 11 de julio de 2023.

Anexos

En este apartado se incluye contenido que, aunque no tenga una relación directa con el objetivo del proyecto, puede ser relevante para el lector de este documento por diversos motivos.

Anexo I. Plataforma de desarrollo y requisitos de hardware

Este proyecto ha sido desarrollado en un ordenador personal con las siguientes características técnicas:

- **Sistema operativo:** Arch Linux 6.4.1-zen1-1-zen.
- **CPU:** AMD Ryzen 7 5800X.
- **Memoria RAM:** 32 GiB.
- **GPU:** AMD Radeon RX 6650 XT. Mesa 23.1.3.
- **IDE:** Visual Studio Code.

Aunque en un principio la idea era usar *PyTorch* junto con *ROCm* (GPU) para hacer uso de la *CNN*, se terminó usando en modo CPU ya que la memoria de la GPU (8 GiB) resultó ser insuficiente. Para medir el uso de memoria se utilizó el método *virtual_memory* del módulo *psutil*. En la **Tabla I.1** se muestra el uso de memoria y el tiempo empleado para entrenar la *CNN* con varios tamaños de recorte.

Tamaño de recorte	Memoria necesaria	Tiempo de entrenamiento
71	9,05 GiB	23 minutos, 9 segundos.
91	10,21 GiB	33 minutos, 47 segundos.
111	9,35 GiB	38 minutos, 29 segundos
131	10,03 GiB	67 minutos, 34 segundos.
151	10,84 GiB	82 minutos, 51 segundos.

Tabla I.1 Medidas de rendimiento para entrenar la *CNN* con 62 sujetos.

Anexo II. Uso del proyecto

En el proyecto se incluye un cuaderno Jupyter para facilitar su uso. En cada celda se incluyen comentarios que especifican qué se realiza en cada celda y qué contienen las diferentes variables.

Requisitos previos

Para ejecutar este proyecto se asume que el usuario dispone de una versión de Python actualizada y tiene instalado el gestor de paquetes *pip*. Para instalar *PyTorch* se recomienda seguir las instrucciones mostradas en su sitio web: https://pytorch.org/

Instrucciones

Aunque se incluyan comentarios en el código para facilitar su uso, aquí se plasman todos los pasos de forma simplificada.

1. **Descarga del proyecto:** El código se encuentra en el siguiente repositorio de *GitHub*:
   https://github.com/vibrantsalt/TFG
   Para descargarlo puede hacerse directamente desde el sitio web o haciendo uso de *git*.
2. **Descarga de la base de datos PaHaW:** Los archivos necesarios y su ubicación se mostraron en el **Apartado 5.1** Las instrucciones para obtener la base de datos PaHaW se encuentran en su sitio web: https://bdalab.utko.fekt.vut.cz/
3. **Instalación de las dependencias necesarias:** En el proyecto se incluye un archivo *requirements.txt* que incluye todas las dependencias y puede ser usado con *pip* mediante el comando:

   ```
   pip install -r requirements.txt
   ```

4. **Ejecución del cuaderno Jupyter:** Si se han seguido todos los pasos deberían ejecutarse todas las celdas.

En el cuaderno Jupyter se incluyen una serie de celdas para facilitar su uso en la plataforma Colab.

Valores relevantes

Para generar los recortes, entrenar la *CNN* y realizar las predicciones hay que tener en cuenta las variables listadas a continuación. Para facilitar su modificación se encuentran todas en la sexta celda del cuaderno Jupyter:

- **model_file_path:** Para facilitar el uso de modelos entrenados en ejecuciones previas del programa se ha añadido una funcionalidad para poder guardar y cargar las *CNN* haciendo uso del módulo *pickle*. Al intentar cargar una *CNN* se realizan una serie de pasos para comprobar que se corresponde con la *CNN* esperada. Al intentar cargar una *CNN* se comprueba si existe la ruta especificada por **model_file_path**, si existe se intenta cargar la *CNN*, pudiendo lanzar un error si no es la *CNN* esperada. Si la ruta no existe, se entrena una nueva *CNN* y se guarda.
- **training_epochs:** Los *epochs* usados para entrenar la *CNN*. En este documento se han usado 30 *epochs* en todos los casos.
- **random_seed:** Es la semilla usada para barajar los grupos H y PD a la hora de formar los grupos de entrenamiento y test. En este documento se ha usado 17 como semilla en todos los casos.
- **task_number:** El número de la tarea de PaHaW que se va a procesar. Hasta ahora este documento se ha centrado únicamente en la Tarea 2, pero el código también está preparado para funcionar con las Tareas 3 y 4.
- **clipping_side_size:** El tamaño del lado de los recortes generados. En el **Apartado 5.2.2** se llegó a la conclusión de que el tamaño mínimo recomendado es 71, y en el **6.1** se mostraron las pruebas realizadas con diferentes tamaños de recorte, llegando a la conclusión de que el tamaño óptimo se encuentra en torno a 111.
- **clipping_jump_size:** Tamaño del salto entre puntos al realizar los recortes. En el **Apartado 5.2.1** se mostró que el tamaño óptimo es 1.

Por último, es interesante hacer notar que la funcionalidad de almacenar las *CNN* también almacena los datos de entrenamiento (*train loss, train accuracy, test loss, test accuracy*) que se generan por cada *epoch*, permitiendo consultarlos posteriormente En cuanto a la función que impide cargar *CNN* incorrectas, los datos que comprueba son: Tarea, sujetos usados para entrenamiento, tamaño de salto, tamaño de recorte y *epochs* usados.

Anexo III. Aplicación del proyecto a las Tareas 3 y 4

Aunque este trabajo se ha centrado en la Tarea 2, el código también ha sido ideado para funcionar con las Tareas 3 y 4. A continuación, se presenta una breve lista de resultados de predicciones realizadas sobre dichas tareas. Para realizar las siguientes pruebas se han usado todos los sujetos de la base de datos PaHaW con la división mostrada en la **Tabla 5.1** y tamaño de recorte 111. En la **Tabla III.1** se muestran los datos asociados al entrenamiento de la red neuronal convolucional para las diferentes tareas.

	Tarea 2	**Tarea 3**	**Tarea 4**
Recortes generados	83955	112217	133535
Memoria usada	9,35 GiB	7,94 GiB	8,50 GiB
Tiempo	38 minutos, 29 segundos.	54 minutos, 10 segundos.	65 minutos, 14 segundos.

Tabla III.1 Diferentes datos sobre el entrenamiento de la *CNN* en base a la tarea objetivo con tamaño de recorte 111.

Tarea 3

Fig. III.1 Plantilla de la Tarea 3 de la base de datos PaHaW.

Observando los resultados de las **Tablas III.2 y III.3** y las matrices de confusión de la **Figura III.2** se puede decir que el trabajo realizado para la Tarea 2 se traslada de forma correcta para esta tarea, superando el 60% de corrección tanto a nivel de tarea como de letra.

Epoch	*Train loss*	*Train accuracy*	*Test loss*	*Test accuracy*
5	0,2115	90,58%	1,5610	57,66%
10	0,0632	97,64%	3,6143	56,65%
15	0,0324	98,83%	4,3927	56,78%
20	0,0228	99,29%	4,8339	57,21%
25	0,0158	99,49%	5,8933	56,39%
30	0,0123	99,63%	5,8668	56,95%

Tabla III.2 Entrenamiento de la *CNN* para la Tarea 3.

	Tareas	**Letras**
Corrección	61,54%	64,62%
Precisión	57,14%	62,07%
Recall	66,67%	60,00%
F1 Score	61,54%	61,02%

Tabla III.3 Valores de la predicción para la Tarea 3.

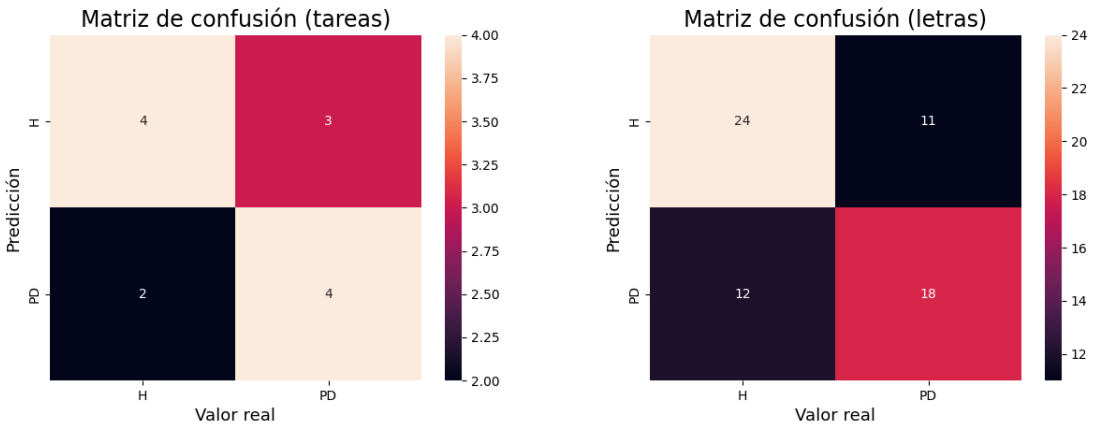

Fig. III.2 Matrices de confusión para la Tarea 3.

En la **Figura III.3** se muestra el resultado final de la predicción generada sobre la Tarea 3 del sujeto 53.

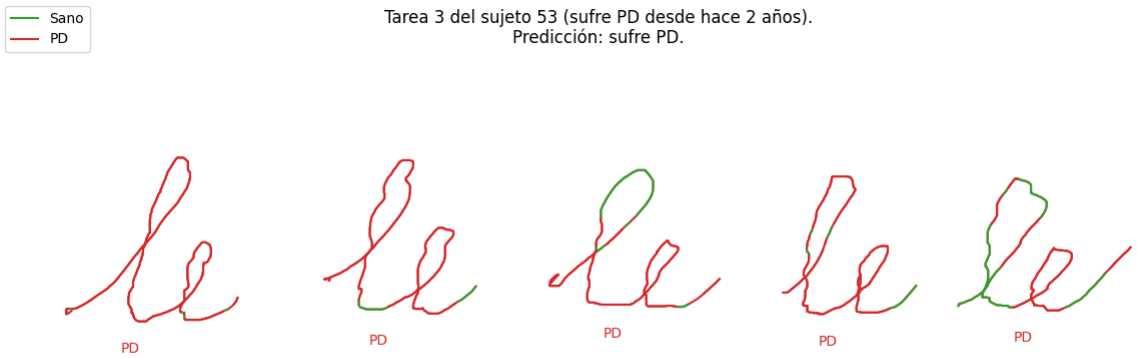

Fig. III.3 Resultado de la predicción realizada sobre la Tarea 3 del sujeto 53.

Tarea 4

Fig. III.4 Plantilla de la Tarea 4 de la base de datos PaHaW.

Epoch	*Train loss*	*Train accuracy*	*Test loss*	*Test accuracy*
5	0,1743	92,48%	2,0304	53,24%
10	0,0420	98,48%	4,3893	53,43%
15	0,0224	99,24%	4,7762	53,72%
20	0,0140	99,56%	5,9187	53,54%
25	0,0109	99,68%	6,1491	54,23%
30	0,0085	99,74%	7,2450	54,20%

Tabla III.4 Entrenamiento de la *CNN* para la Tarea 4.

	Tareas	**Letras**
Corrección	69,23%	65,67%
Precisión	66,67%	63,33%
Recall	66,67%	61,29%
F1 Score	66,67%	62,30%

Tabla III.5 Valores de la predicción para la Tarea 4.

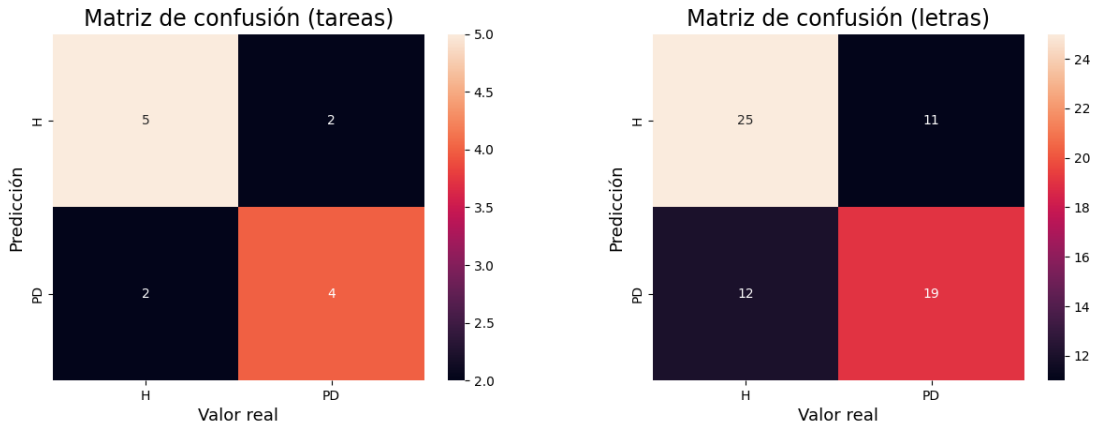

Fig. III.5 Matrices de confusión para la Tarea 4.

Al igual que para la Tarea 3, los resultados mostrados en las **Tablas III.4 y III.5** indican que el trabajo realizado para la Tarea 2 se traslada de forma correcta para la Tarea 4, superando también el 60% de corrección.

En la **Figura III.6** se muestra el resultado final de la predicción generada sobre la Tarea 4 del sujeto 53.

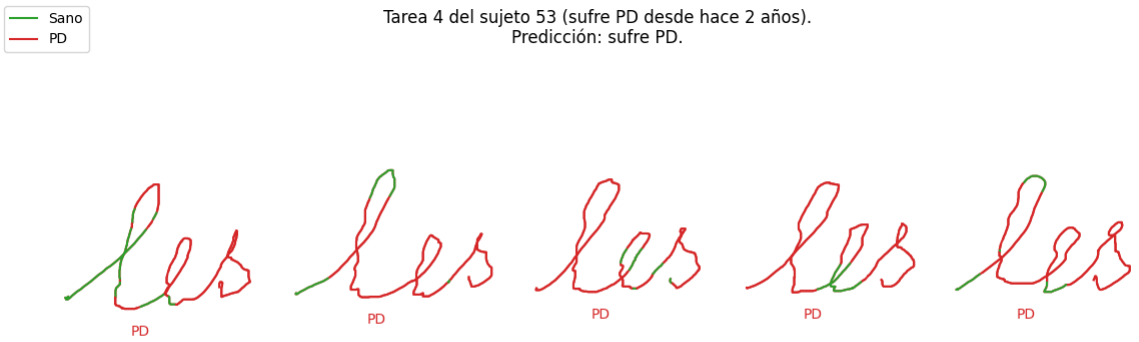

Fig. III.6 Resultado de la predicción realizada sobre la Tarea 4 del sujeto 53.

Comparativa de los resultados para las diferentes tareas

	Tareas (2)	Tareas (3)	Tareas (4)
Corrección	69,23%	61,54%	69,23%
Precisión	66,67%	57,14%	66,67%
Recall	66,67%	66,67%	66,67%
F1 Score	66,67%	61,54%	66,67%

Tabla III.6 Comparativa de la predicción de las diferentes tareas usando 62 sujetos para entrenamiento y 13 para test con tamaño de recorte 111.

	Letras (2)	Letras (3)	Letras (4)
Corrección	61,54%	64,62%	65,67%
Precisión	58,06%	62,07%	63,33%
Recall	60,00%	60,00%	61,29%
F1 Score	59,02%	61,02%	62,30%

Tabla III.7 Comparativa de la predicción de las diferentes tareas usando 62 sujetos para entrenamiento y 13 para test con tamaño de recorte 111.

Los datos mostrados en las **Tablas III.6 y III.7** demuestran que es posible obtener una corrección cercana al 70% siguiendo la metodología explicada a lo largo de este documento tanto para la Tarea 2 como para la 3 y la 4.