

# Identificación de pacientes de Parkinson a través de la voz

---

Julia Serrano Arrese c200119

José Alberto Cañibano López a180192

# ÍNDICE

---

**01**

Introducción

**02**

Dataset

**03**

División

**04**

Ondas de audio

**05**

Espectrogramas

**06**

Preprocesamiento

**07**

Modelo y entrenamiento

**08**

Evaluación

# Introducción

Breve introducción sobre el proyecto



## Objetivo

Aprendizaje automático  
Identificación Parkinson  
Grabaciones voz



## Dataset

MDVR-KCL



## Relevancia

Detección temprana  
Mejorar tratamiento y calidad de vida

# Dataset

Conjunto de datos utilizado para el proyecto



## Fuente

- Mobile Device Voice Recordings at King's College London
- Zenodo
- Conversión 16 bit



## Descripción

- Grabaciones de voz
- Pruebas: lectura de texto, diálogo espontáneo



## Etiquetas

- HC: Healthy Control Subject
- PD: Parkinson's Disease Subject



## Muestras

- Total de 73 muestras:
- 42 HC
  - 31 PD

# División

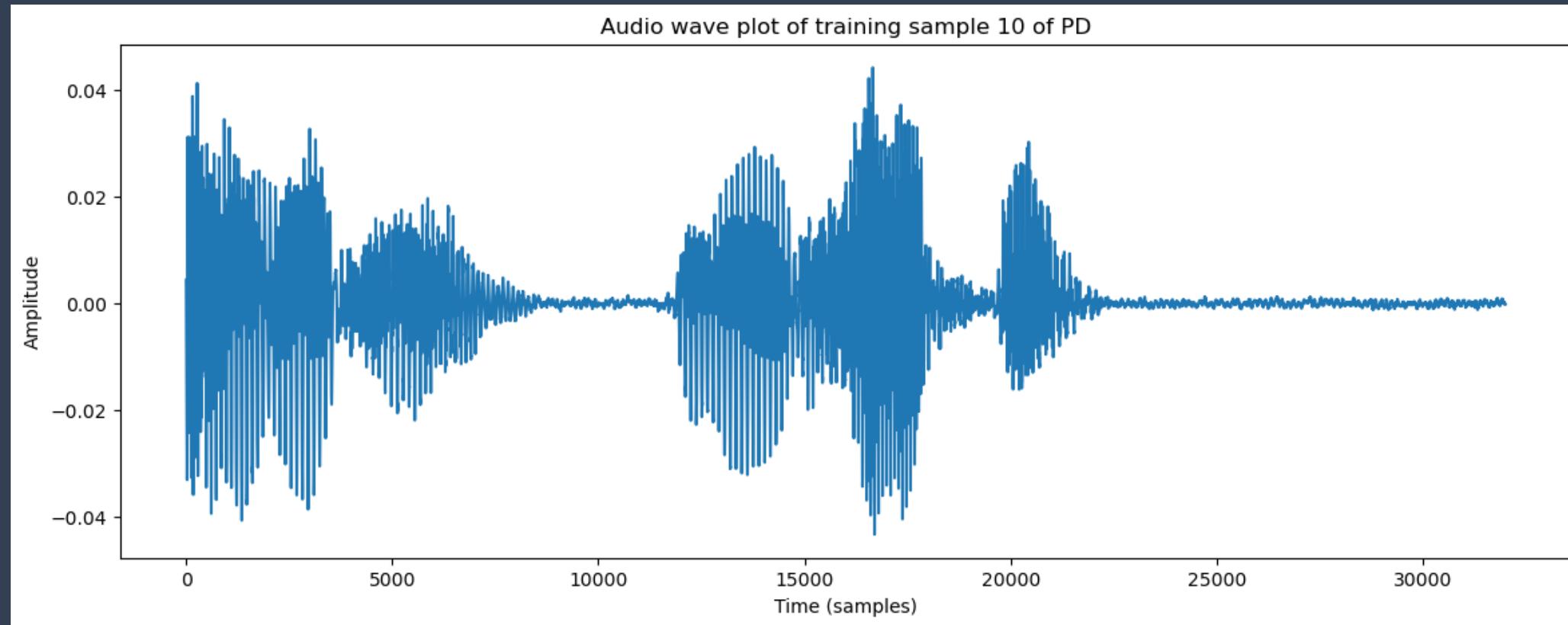
1. Dividimos en sets de entrenamiento, validación y prueba
2. Obtenemos 40 fragmentos aleatorios de 2 segundos por cada archivo de audio

```
def get_random_fragment(num.fragments, filename, sr, duration):  
    # Carga el archivo de audio  
    audio, _ = librosa.load(filename, sr=sr)  
  
    # Divide el audio en fragmentos de "duration" segundos  
    fragment_list = librosa.util.frame(audio, frame_length=sr*duration, hop_length=sr*duration)  
  
    # Obtiene el número de fragmentos (columnas) que hay en fragment_list.  
    num_frames = fragment_list.shape[1]  
  
    # Obtiene "num.fragments" números aleatorios  
    indices = tf.random.uniform(shape=(num.fragments,), maxval=num_frames, dtype=tf.int32)  
  
    # Lista vacía "fragments_random" que almacenará los fragmentos aleatorios.  
    fragments_random = []  
  
    # Recorre los índices aleatorios y para cada uno de ellos obtiene el fragmento correspondiente  
    for idx in indices:  
        fragment = fragment_list[:, idx]  
        fragments_random.append(fragment)  
  
    # Devuelve la lista de fragmentos aleatorios.  
    return fragments_random  
}
```

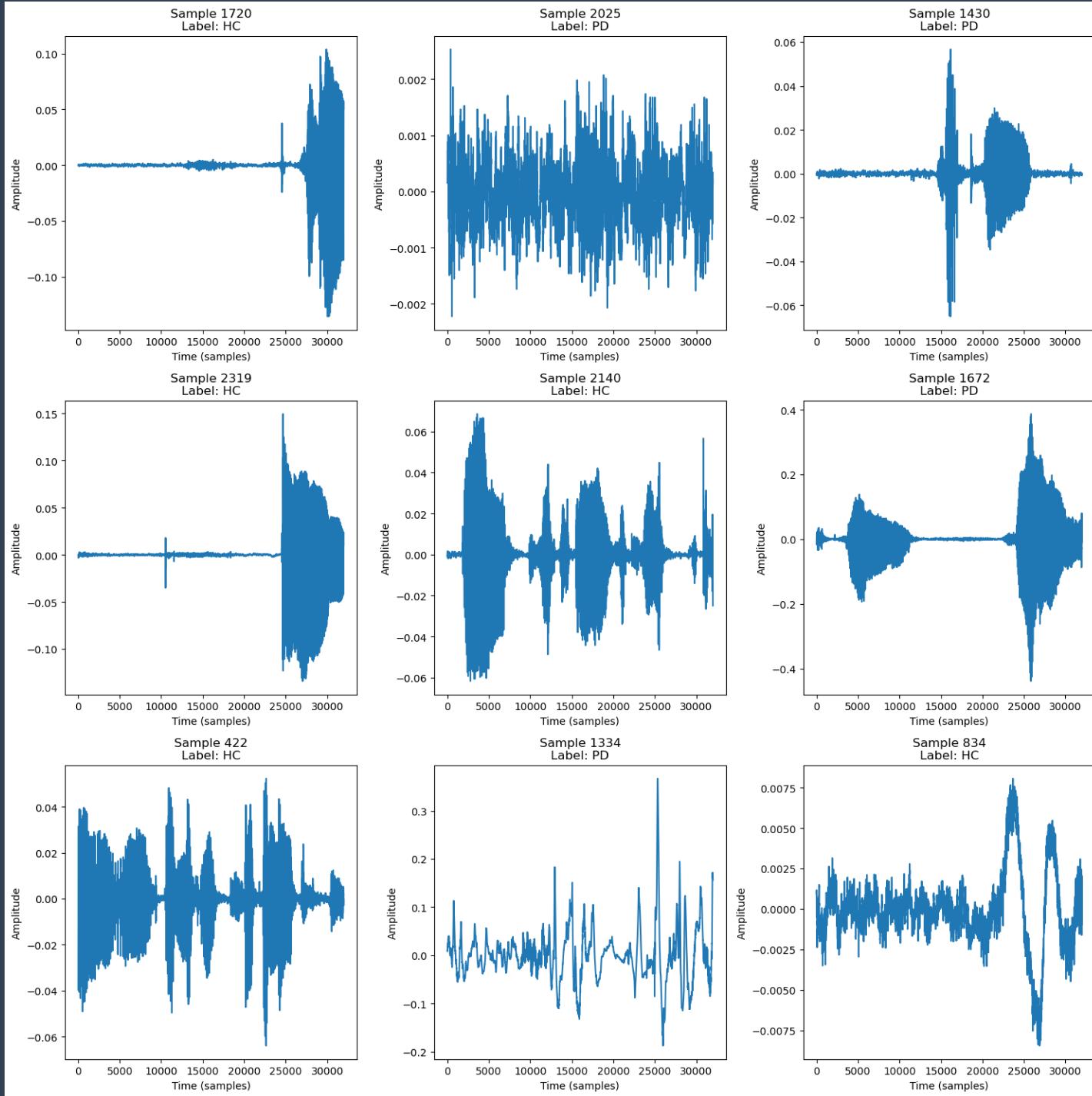
# Ondas de audio

---

Representación gráfica de una señal de audio a lo largo del tiempo.



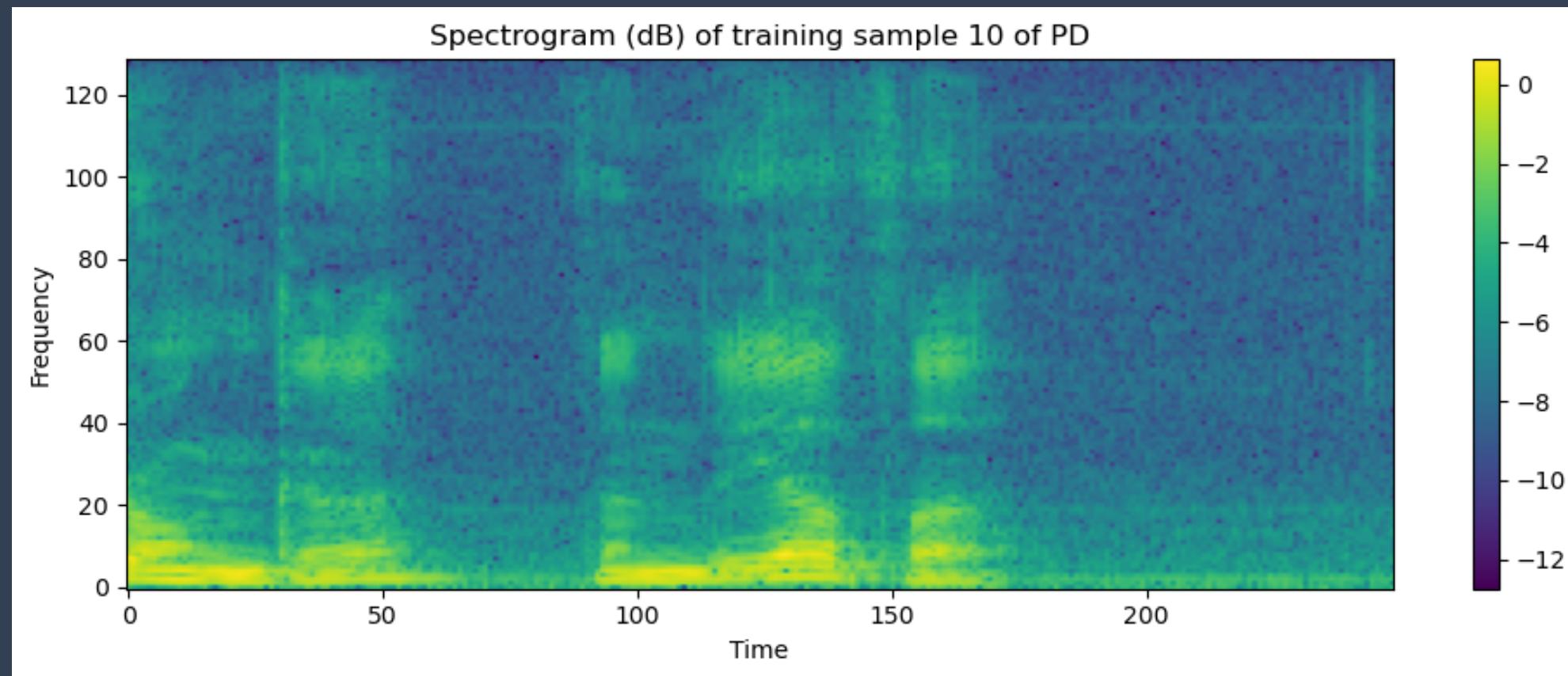
# Visualización de ondas aleatorias

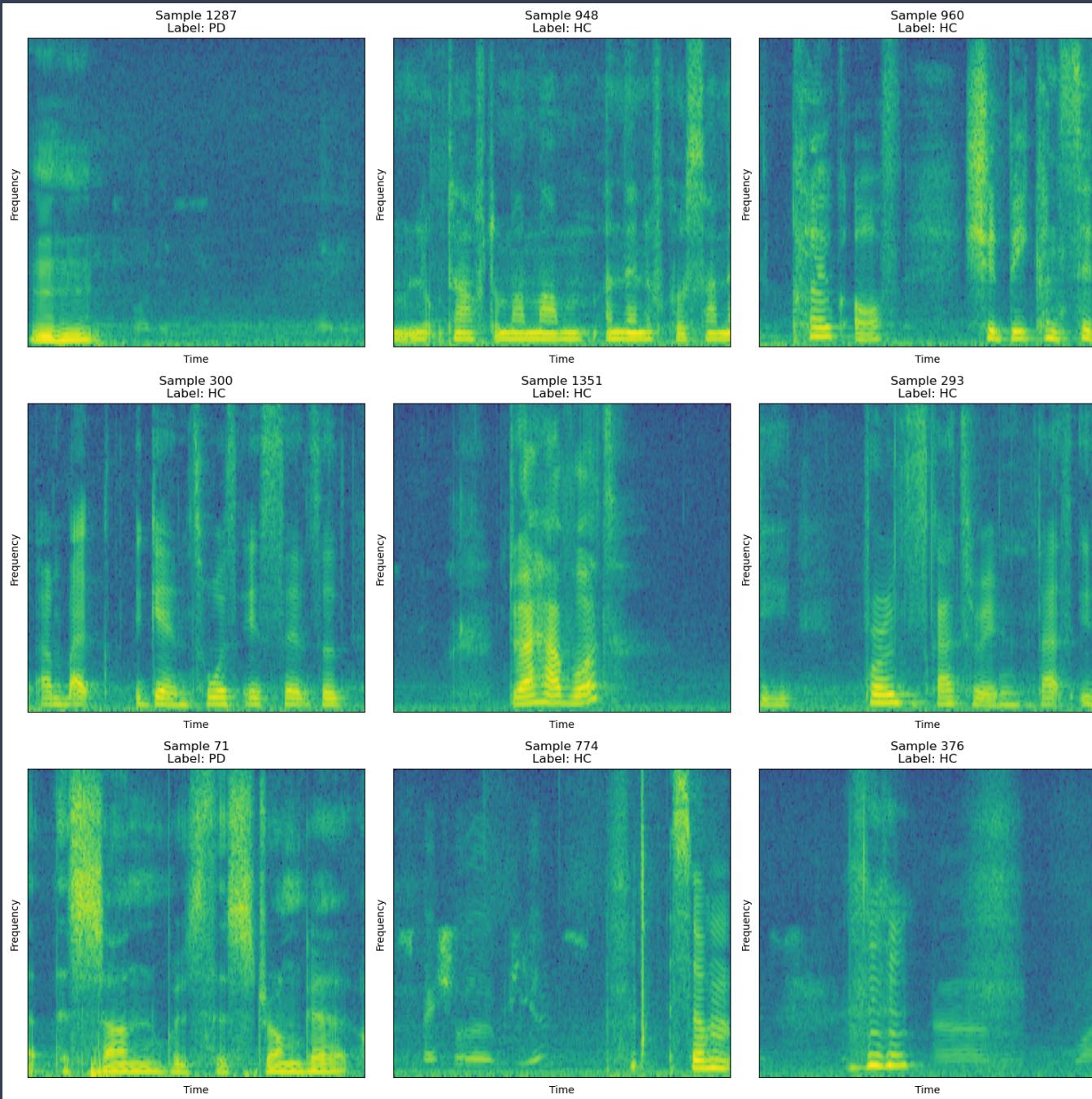


# Espectrogramas

---

Representación gráfica de la evolución temporal del espectro de frecuencia de una señal de sonido





# Visualización de espectrogramas aleatorios

# Preprocesamiento de datos

Realizamos el preprocesamiento de datos sobre cada conjunto de fragmentos

Dataset

Waveform

Espectrograma

Preprocesado



Obtención de un Tensorflow dataset a partir de tensores de fragmentos de audio y etiquetas



Obtención de formas de onda de audio a partir del dataset anterior



Transformación de las formas de onda en espectrogramas de frecuencia-tiempo



Preprocesamiento final de los datos para ser utilizados en nuestro modelo

# Modelo de red neuronal

---

Red convolucional (CNN)  
simple

## Descripción del modelo

Capas de preprocessamiento



Batch size de 64

## Hiperparámetros del modelo

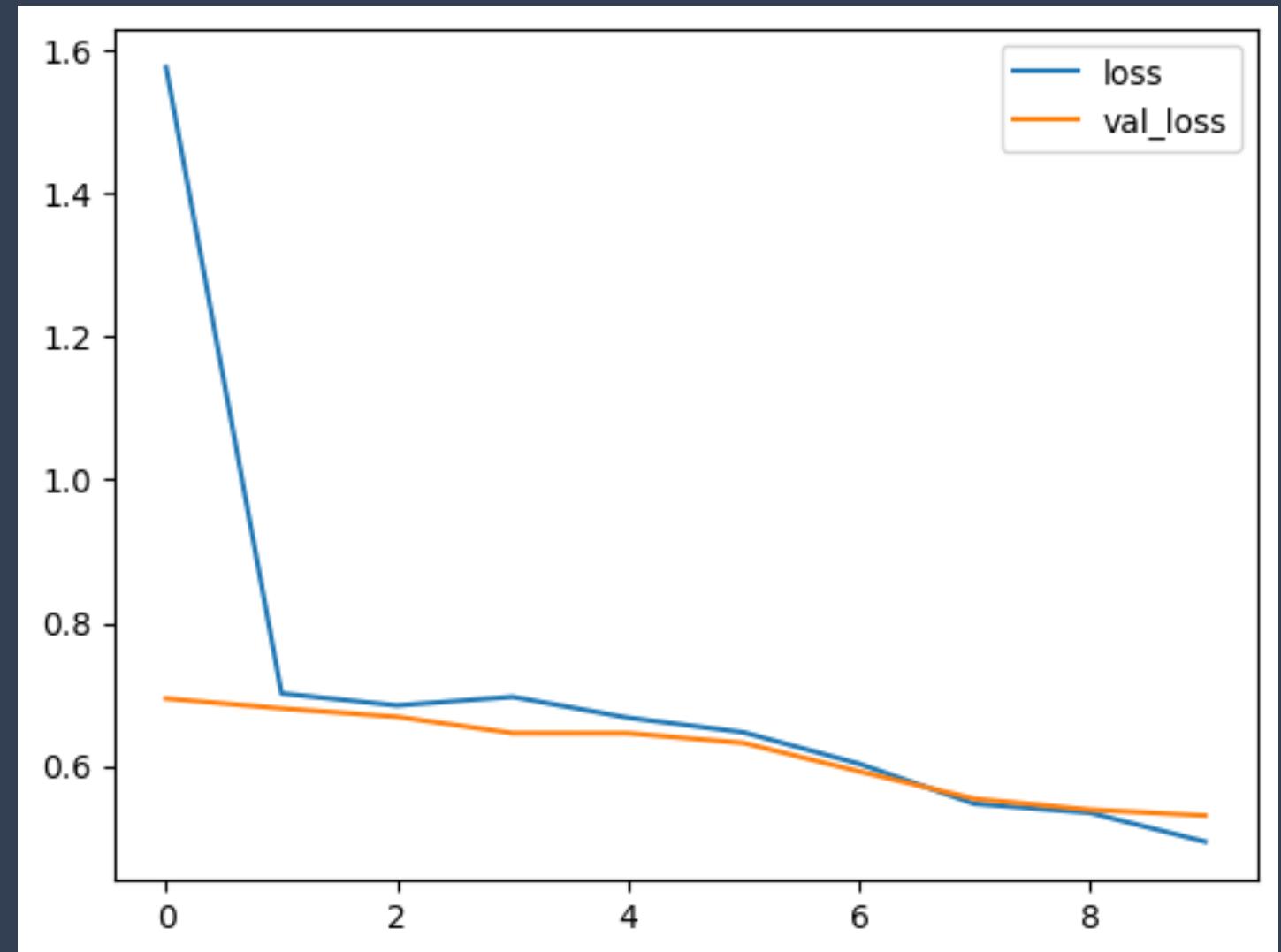
Reducción de latencia a  
través de Dataset.cache y  
Dataset.prefetch

Optimizador Adam y  
pérdida de entropía cruzada

# Entrenamiento del modelo

---

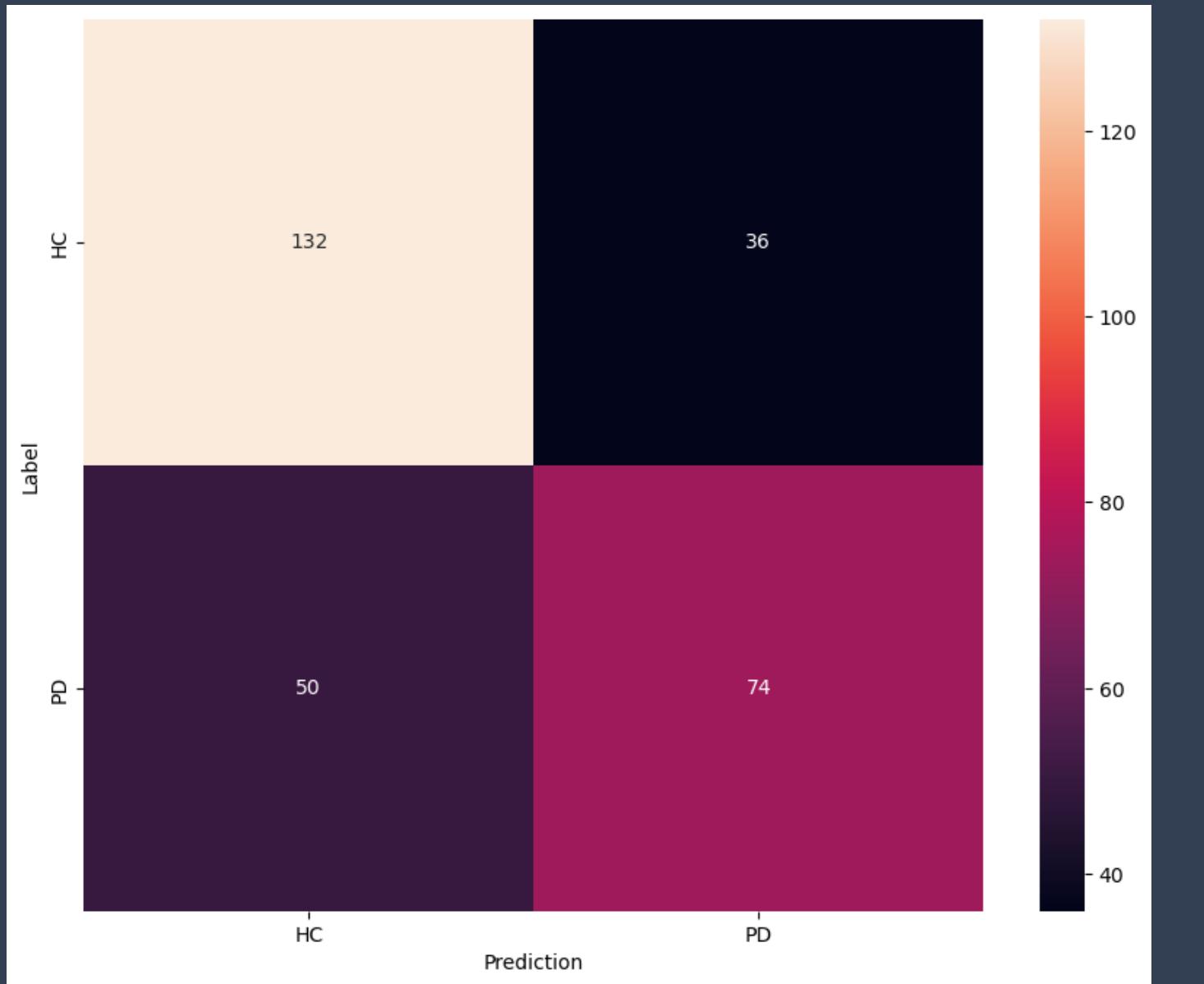
- Entrenamiento con conjuntos de entrenamiento y validación
- 10 épocas de entrenamiento
- Gráfico de la función de pérdida para visualizar la mejora del modelo



# Evaluación del modelo

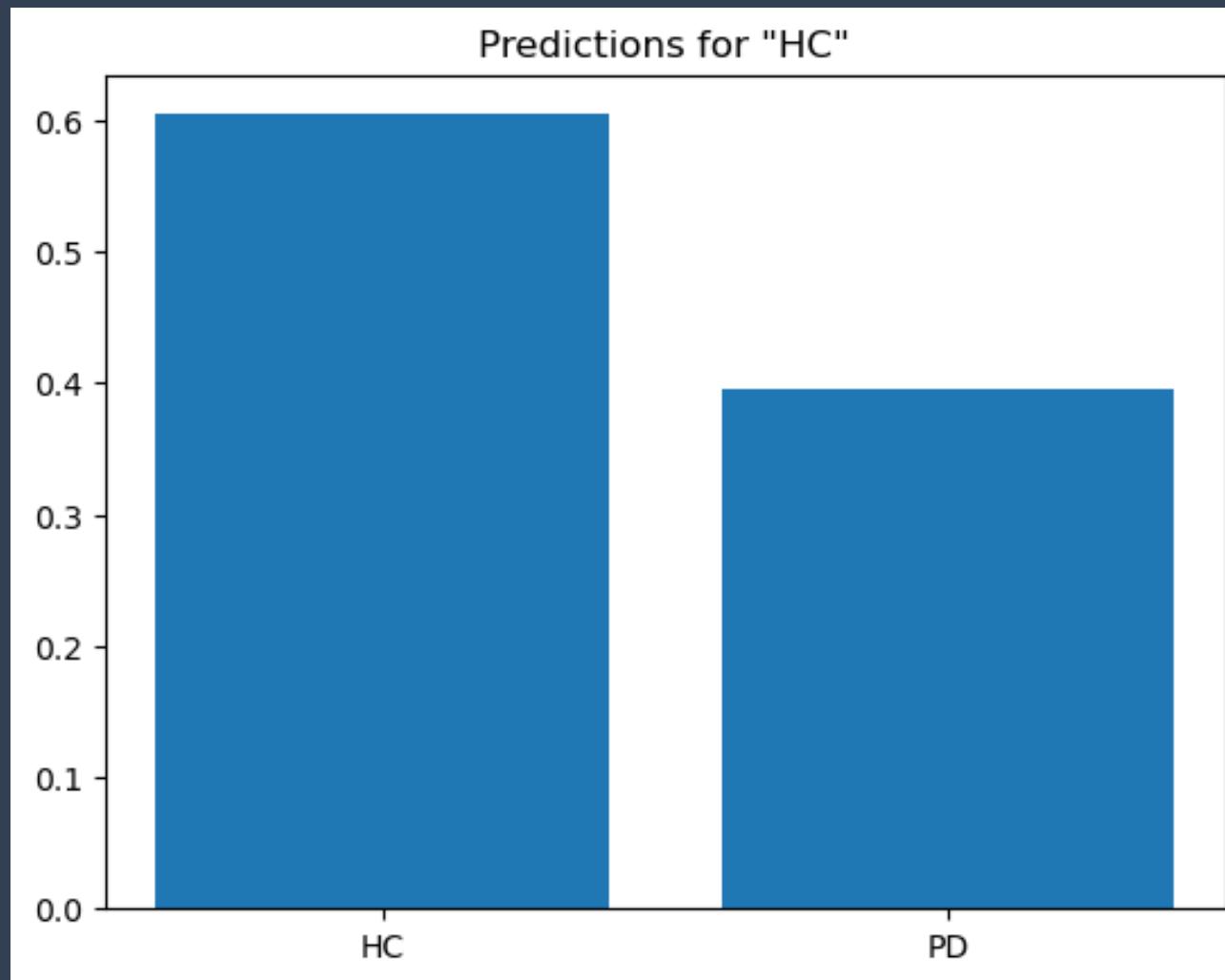
---

- Test accuracy: 71%
- 10 épocas de entrenamiento
- Matriz de confusión
  1. Diagonal principal:  
predichos correctamente
  2. Diagonal secundaria:  
predichos incorrectamente



# Ejecutar inferencia en un archivo de audio

---



# Conclusión

---