ULPGC
Universidad de
Las Palmas de
Gran Canaria

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

Aplicación de modelos de aprendizaje profundo para el análisis de pacientes con COVID

Titulación: Grado en Ingeniería en Tecnologías de la Telecomunicación

Autor: Sergio Pinto Pérez

Tutores: D. Carlos M. Travieso González

D. Sergio Celada Bernal

Fecha: 20/07/2024

Agradecimientos

En primer lugar, a mis padres, por tener paciencia conmigo, ser un apoyo diario y confiar siempre en mi a lo largo de la carrera, apoyándome en cada decisión que he ido tomando a lo largo de los años. A toda mi familia por estar ahí siempre y sacarme una sonrisa en cada comida, cena de los viernes y demás momentos. A mi hermano por ser la mejor persona que he podido tener a mi lado cada día y por confiar en que este día llegaría desde el primer minuto que decidí meterme en este grado.

A mis compañeros de la carrera, sin ellos este camino habría sido muy aburrido. En especial a Paula, por enseñarme a mantener siempre la calma en todo momento y a Andrés y Juanki, por ser un ejemplo diario de esfuerzo, dedicación y disciplina. Tantos días y noches en la biblioteca al fin han dado sus frutos.

A mis tutores de este trabajo, Carlos Travieso y Sergio Celada, gracias por darme la oportunidad de hacer este TFG y por ayudarme a conseguir cerrar esta bonita etapa.

Resumen

La pandemia de COVID-19, desencadenada por la propagación global del virus SARS-CoV-2 a partir de finales de 2019, tuvo un impacto profundo en todos los aspectos de la sociedad. Esta enfermedad respiratoria altamente contagiosa impactó los sistemas de salud en todo el mundo, abrumando hospitales y generando una urgente necesidad de herramientas eficaces para combatirla. Desde su aparición inicial en la ciudad china de Wuhan, el virus se extendió a más de 200 países y territorios, causando millones de casos confirmados y un impacto devastador en la salud pública.

La complejidad de la COVID-19 residía en su amplio espectro de presentaciones clínicas, que iban desde casos asintomáticos hasta síntomas graves y potencialmente mortales. La variabilidad en la progresión de la enfermedad desafió la capacidad de los profesionales médicos para prever y gestionar sus consecuencias, destacando la necesidad crítica de herramientas de análisis predictivo que pudieran guiar las decisiones clínicas.

En este contexto, surge la importancia de explorar nuevas estrategias para el análisis de datos clínicos relacionados con la COVID-19. Para este estudio, nos apoyaremos en una base de datos proporcionada por el personal sanitario, que contiene información detallada sobre la evolución y características de los pacientes afectados por la enfermedad. La aplicación de técnicas de aprendizaje automático, en particular el aprendizaje profundo, ofrece una oportunidad emocionante para analizar estos datos de manera exhaustiva y descubrir patrones ocultos que puedan mejorar nuestra capacidad de prever la evolución de la enfermedad. El objetivo de este proyecto de fin de grado es utilizar estas técnicas avanzadas para modelar datos de pacientes con COVID-19, proporcionando información predictiva valiosa que pueda complementar la experiencia clínica y mejorar la atención médica.

Abstract

The COVID-19 pandemic, triggered by the global spread of the SARS-CoV-2 virus starting in late 2019, had a profound impact on all aspects of society. This highly contagious respiratory illness affected healthcare systems worldwide, overwhelming hospitals and creating an urgent need for effective tools to combat it. Since its initial appearance in the Chinese city of Wuhan, the virus spread to over 200 countries and territories, causing millions of confirmed cases and a devastating impact on public health.

The complexity of COVID-19 lay in its wide spectrum of clinical presentations, ranging from asymptomatic cases to severe and potentially fatal symptoms. The variability in disease progression challenged healthcare professionals' ability to predict and manage its consequences, underscoring the critical need for predictive analytics tools to guide clinical decisions.

In this context, the importance of exploring new strategies for analyzing clinical data related to COVID-19 emerges. For this study, we will rely on a database provided by healthcare personnel, containing detailed information on the evolution and characteristics of patients affected by the disease. The application of machine learning techniques, particularly deep learning, offers an exciting opportunity to analyze this data comprehensively and uncover hidden patterns that could enhance our ability to predict disease progression. The aim of this undergraduate project is to utilize these advanced techniques to model COVID-19 patient data, providing valuable predictive information that can complement clinical expertise and improve healthcare delivery.

ÍNDICE GENERAL

Parte 1: Memoria

Parte 2: Presupuesto

Parte 3: Pliego de condiciones

ÍNDICE DE FIGURAS

ÍNDICE TABLAS

PARTE 1: MEMORIA

Capítulo 1: Introducción

1.1. Motivación

En España, el año 2020 estuvo marcado por la aparición y propagación del COVID-19, un virus que emergió a finales de 2019 y rápidamente se convirtió en una pandemia mundial. A medida que el virus se extendió por todo el país, las autoridades tomaron medidas drásticas, incluyendo el cierre de escuelas, restricciones de viaje y un estricto confinamiento domiciliario. La primera ola de la pandemia golpeó con fuerza durante la primavera de 2020, llevando a una situación sin precedentes donde los hospitales se vieron desbordados y los recursos médicos fueron escasos [1]. La falta de equipo de protección adecuado para el personal sanitario y la rápida propagación del virus entre los ancianos y otros grupos vulnerables contribuyeron a la crisis.

A lo largo del año, España experimentó múltiples olas de la pandemia, cada una con sus propios desafíos y consecuencias. La situación evolucionó con el tiempo, con períodos de confinamiento intermitente y restricciones continuas para contener la propagación del virus [2]. A pesar de los esfuerzos por controlar la situación, la pandemia tuvo un impacto devastador en la sociedad española, afectando no solo la salud pública, sino también la economía y la vida cotidiana de los ciudadanos [3].

A nivel mundial, los datos actuales muestran la magnitud de la pandemia de COVID-19. Según las cifras de la Universidad Johns Hopkins [4], se han confirmado más de 695 millones de casos y se han reportado casi 6.9 millones de muertes en todo el mundo debido al virus. Estas cifras reflejan el alcance global del impacto de la pandemia y subrayan la urgencia de continuar desarrollando estrategias efectivas para combatir la propagación del virus y proteger la salud pública.

Este Trabajo Fin de Grado (TFG) surge de la iniciativa del equipo de salud de la Unidad de Medicina Intensiva (UMI) del Hospital Insular del Materno Infantil de Las Palmas de Gran Canaria, que busca desarrollar una solución tecnológica innovadora para analizar datos clínicos relacionados con pacientes COVID-19. Debido a las condiciones de alta demanda y presión experimentadas durante la pandemia, ciertos

datos necesarios no pudieron ser recopilados de manera exhaustiva. Agradecemos sinceramente los esfuerzos del Hospital Insular del Materno Infantil de Las Palmas de Gran Canaria por proporcionarnos una gran base de datos, que representa un recurso invaluable para comprender mejor la progresión de la enfermedad y mejorar las estrategias de atención médica. La herramienta propuesta también tiene como objetivo prever con precisión la evolución de los pacientes infectados, permitiendo al personal médico anticiparse a los eventos clínicos y tomar decisiones informadas sobre el nivel de atención necesario, como la posible transferencia a unidades de cuidados intensivos (UCI).

1.2. Trabajos previos

En los últimos años, se han hecho numerosos proyectos para desarrollar herramientas tecnológicas basadas en inteligencia artificial y procesamiento de datos, especialmente en el ámbito de la salud.

Un estudio llevado a cabo en 2021 en la Universidad de Chile [5] tuvo como objetivo desarrollar modelos de *Machine Learning* para predecir el estado de salud de pacientes COVID-19, tanto en seguimiento domiciliario como hospitalizados. Se desarrollaron dos modelos: el primero para predecir el desenlace de pacientes hospitalizados, clasificándolos en alto o bajo riesgo de fallecer, y el segundo para predecir la severidad de la enfermedad en pacientes bajo seguimiento domiciliario, clasificándolos en alto o bajo riesgo. Los resultados mostraron que el Modelo 1 obtuvo un AUC de 0.77, una precisión de 0.96, y una sensibilidad de 0.84, mientras que el Modelo 2 alcanzó un AUC de 0.87, una precisión de 0.12, y una sensibilidad de 0.87.

En el mismo año, un estudio publicado en la RICT Revista de Investigación Científica, Tecnológica e Innovación [6] propuso utilizar modelos de inteligencia artificial para predecir si un paciente con COVID-19 requeriría hospitalización, basado en síntomas. Utilizando 13,757,682 registros de pacientes de México y aplicando algoritmos de regresión logística y redes neuronales, se obtuvieron resultados destacados. Las redes neuronales mostraron una precisión de 0.832 y una tasa de acierto de 0.775.

Otro trabajo significativo de 2021, realizado en la Universidad Politécnica de Valencia, se centró en desarrollar un algoritmo de aprendizaje automático capaz de clasificar a los pacientes hospitalizados por COVID-19 según su gravedad [7]. Este algoritmo, diseñado para situaciones de colapso hospitalario, obtuvo una precisión del 74.8% y una tasa de acierto del 79.7%, demostrando ser una herramienta potencialmente útil para el personal clínico en la gestión de pacientes críticos.

En 2023, un estudio publicado en la Revista Habanera de Ciencias Médicas revisó la literatura sobre modelos de predicción de la severidad en pacientes confirmados de COVID-19 [8].Este estudio identificó que las técnicas más utilizadas fueron la regresión logística, otros algoritmos, redes neuronales y árboles de clasificación, destacando la importancia y diversidad de enfoques en este campo, pero dejando claro que el enfoque de redes neuronales es muy prometedor.

Además, un estudio reciente se centró en la predicción de eventos de hospitalización en UCI y muerte utilizando registros médicos y herramientas de aprendizaje automático [9]. Utilizando un 80% de los datos para entrenamiento y el 20% para prueba, se lograron tasas de precisión notables, especialmente a tres días antes del evento, con una sensibilidad de 0.887 y una precisión de 0.622. En este estudio en concreto se variaron los días, es decir, se hicieron las pruebas con datos de 2, 3 y 5 días antes del suceso, en este caso muerte/ingreso en la UCI o alta. Los resultados de los tres días fueron parecidos, aunque en cuando se acerca el horizonte de predicción (5 días), la sensibilidad llegó a disminuir hasta el 21,6%.

En la Universidad de Las Palmas de Gran Canaria (ULPGC), se realizaron dos trabajos relevantes. El primero, un trabajo de fin de Máster se centró en predecir la evolución de pacientes COVID-19, determinando si pasarían a sala o a la UCI [10]. Para ello, se utilizaron sistemas regresivos y redes neuronales convolucionales, alcanzando una tasa de acierto del 59%, una sensibilidad del 59.5% y una precisión del 89%. El segundo trabajo [11], continuación del primero y predecesor del presente estudio, aplica técnicas de aprendizaje profundo y redes neuronales recurrentes, específicamente LSTM y GRU, con el objetivo de mejorar los resultados previos y lograr una mayor precisión y fiabilidad en las predicciones de la evolución de los pacientes.

En concreto, se obtuvo una precisión del 94,5%, una sensibilidad del 73,8% y una tasa de acierto del 72,0%, siendo estos unos resultados muy prometedores.

1.3. Propuesta

En el marco de este Trabajo Fin de Grado, se emplearán tecnologías innovadoras con el objetivo de facilitar las labores del personal médico y obtener datos esenciales. La fiabilidad de estos datos será fundamental, ya que proporcionarán información valiosa para el personal médico y contribuirán a combatir los efectos de la COVID-19. En este estudio se aplicarán modelos de aprendizaje profundo comparando diferentes arquitecturas de redes neuronales artificiales como son GRU, LSTM, BiLSTM, CNN con LSTM y Transformers para así comprobar cuál de ellas genera unos mejores resultados. Además, se comparan estos resultados con los obtenidos en investigaciones previas relevantes.

Los datos utilizados en esta investigación proceden de registros obtenidos durante la pandemia de COVID-19 por el personal médico del Complejo Hospitalario Universitario Insular Materno Infantil de Las Palmas de Gran Canaria. Agradecemos sinceramente la colaboración de los tutores de este proyecto y el personal médico del Complejo Hospitalario Universitario Insular Materno Infantil de Las Palmas de Gran Canaria por su invaluable apoyo en la obtención de estos datos.

En el transcurso de este trabajo, se garantizará la protección de la privacidad de los pacientes, respetando escrupulosamente los principios éticos establecidos en la Declaración de Helsinki y las normativas sanitarias vigentes aplicables a la investigación clínica con personas enfermas o sanas.

1.4. Objetivo

El propósito principal de este Trabajo de Fin de Grado consiste en desarrollar un sistema que, al utilizar una base de datos de señales adquiridas con información de pacientes afectados por condiciones de COVID, y al examinar las tendencias actuales en modelos de aprendizaje profundo, pueda anticipar la progresión de estos pacientes. Este objetivo se desglosa en metas más específicas:

- O1: Investigar y examinar las últimas técnicas empleadas en la predicción de la afección de la COVID.
- O2: Evaluar y categorizar la información contenida en la base de datos proporcionada.
- O3: Aplicar técnicas de aprendizaje profundo para prever el desarrollo futuro de los pacientes con COVID.

1.5. Peticionario

El peticionario del presente Trabajo Fin de Grado es la Escuela de Ingeniería de Telecomunicación y Electrónica (EITE) de la Universidad de Las Palmas de Gran Canaria (ULPGC), en calidad de institución pública que solicita la realización de dicho trabajo con el fin de superar los requisitos impuestos en obtención del título del Grado de Ingeniería de Telecomunicaciones en la especialidad de Sistemas de Telecomunicación.

1.6. Estructura del documento

El trabajo estará dividido en tres secciones, que serán distribuidas de la siguiente manera: Memoria, Pliego de Condiciones y Presupuesto. La sección de la Memoria estará dividida en 7 capítulos más un apartado adicional que será la Bibliografía. Estos 7 capítulos contendrán lo siguiente:

- **Capítulo 1: Introducción.** En este primer capítulo se expondrán los motivos que han dado lugar a la realización de este Trabajo de Fin de Grado (TFG). Se mencionarán y analizarán algunos trabajos previos relacionados con el tema. Luego se desarrollará la propuesta de este TFG, definiendo los objetivos planteados para su realización. Finalmente, se describirá al solicitante y se presentará la estructura del contenido de este trabajo.
- **Capítulo 2: Materiales.** Se describirán los materiales utilizados en el trabajo, con un enfoque particular en la base de datos proporcionada por el personal médico del C.H.U. Insular-Materno Infantil de Las Palmas de Gran Canaria. Esta base de datos contiene medidas reales tomadas a pacientes con SARS-CoV-2.

Se detallará cómo se han tratado y filtrado estos datos para asegurar su relevancia y calidad en el estudio posterior.

- **Capítulo 3: Métodos.** Aquí se presentarán los conceptos fundamentales de inteligencia artificial, *Machine Learning* y *Deep Learning*. Se profundizará en las arquitecturas utilizadas, tales como GRU, LSTM, BiLSTM, Transformers y CNN con LSTM, dentro del contexto de redes neuronales recurrentes. Se explicarán los enfoques de aprendizaje supervisado, no supervisado y por refuerzo, destacando la importancia de cada uno en la construcción y entrenamiento de modelos predictivos.
- **Capítulo 4: Metodología experimental.** Se explicará la metodología experimental empleada, justificando la elección de MATLAB como herramienta de programación. Además, se describirá qué es una matriz de confusión y cómo se ha utilizado para evaluar el rendimiento de los modelos desarrollados.
- **Capítulo 5: Implementación del modelo.** Este capítulo incluirá el código MATLAB utilizado para generar la matriz de confusión, junto con una explicación detallada. También se presentará y explicará el código de MATLAB de las diferentes redes neuronales aplicadas en el estudio.
- **Capítulo 6: Experimentos y resultados.** Se presentarán los resultados obtenidos de cada red neuronal, destacando los mejores resultados de cada arquitectura. Se seleccionará una arquitectura final con parámetros optimizados, y se compararán estos resultados con los de estudios previos. Además, se incluirá la evaluación de un profesional médico sobre los resultados obtenidos.
- **Capítulo 7: Conclusiones y líneas futuras.** Se resumirán las conclusiones generales del trabajo, verificando el cumplimiento de los objetivos planteados. También se discutirán las posibles mejoras y ampliaciones futuras que podrían derivarse de este estudio.

Capítulo 2: Materiales

2.1. Descripción de la base de datos

La base de datos con la que se va a trabajar en este Trabajo de Fin de Grado consiste en un fichero en formato Excel proporcionado por el personal médico del C.H.U. Insular-Materno Infantil de Las Palmas de Gran Canaria. La aportación de esta base de datos está asociada al proyecto "Evaluación, seguimiento y manejo de síndrome postCOVID tras ingreso en UCI (SPC-UCI). Estudio comparativo con síndrome postUCI", con código CEIm H.U.G.C. Dr. Negrín: 2020-231-1 COVID-19 y fecha 30 de agosto de 2021. En la base de datos original hay 132.870 datos en total, correspondientes a medidas reales de varias pruebas realizadas a pacientes. En la siguiente figura podemos observar un ejemplo de cómo se vería la BDD:

	A	B	C	D	E	F
1	NHC	NOMBRE_INDICADOR	VALOR	NUMERO_SLOT_FINAL	PLANTA_UCI	NUMERO_SLOT_PRUEBA_PACIENTE
2	100179	FC	74,96704298	1	PLANTA	1
3	100179	FC	80	2	PLANTA	2
4	100179	FC	90	3	PLANTA	3
5	100179	FC	76	1	PLANTA	4
6	100179	FC	78	2	PLANTA	5
7	100179	FC	70	3	PLANTA	6
8	100179	FC	83,66020225	1	PLANTA	7
9	100179	FC	84,00345945	2	PLANTA	8
10	100179	FC	76	3	PLANTA	9
11	100179	FC	87	1	PLANTA	10
12	100179	FC	84,04301146	2	PLANTA	11
13	100179	FC	83,84051513	3	PLANTA	12
14	100179	FC	90	1	PLANTA	13
15	100179	FC	90	2	PLANTA	14
16	100179	FC	99	3	PLANTA	15
17	100179	FC	86	1	PLANTA	16
18	100179	FC	82,37757982	2	PLANTA	17
19	100179	FC	82,12482672	3	PLANTA	18
20	100179	FC	64	1	PLANTA	19
21	100179	FC	89	2	PLANTA	20
22	100179	FC	92	3	PLANTA	21
23	100179	FC	72	1	PLANTA	22
24	100179	FC	81,67893566	2	PLANTA	23
25	100179	FC	81,78959804	3	PLANTA	24

Figura 1. Base de datos utilizada

En la imagen se pueden ver varios parámetros organizados en columnas que fueron empleados en la elaboración de este Trabajo de Fin de Máster y se detallan a continuación en la Tabla:

Parámetro	Descripción
NHC	Número de identificación del/la paciente con de objetivo de mantener la privacidad de este.
NOMBRE_INDICADOR	Hace referencia al tipo de prueba realizada por el personal médico.
VALOR	Indica el valor numérico, resultado de la prueba realizada.
NUMERO_SLOT_FINAL	Indica el momento del día en el que se realiza la prueba, siendo 1 la mañana, 2 la tarde y 3 la noche.
PLANTA_UCI	Expresa la situación del/la paciente, en función de si el/la paciente se encuentra en planta o en la UCI.
NUMERO_SLOT_PRUEBA_PACIENTE	Contador del número de veces que se le ha realizado una determinada prueba a un paciente en concreto.

Tabla 1. Detalles de la base de datos utilizada

2.1. Análisis y preparación de los datos

Previo al uso de las redes neuronales habrá que analizar y clasificar la base de datos, con el fin de saber si todos los datos están completos y nos son útiles para el trabajo. Se ha realizado un filtrado de los datos para una obtención más real de resultados. En concreto, se han tenido dos factores en cuenta:

1. El número de pruebas médicas debe ser mayor de 130. Es decir, si una prueba médica se ha realizado en total menos de 130 veces, todas las filas en las que aparezca esa prueba quedarían eliminadas.

2. El número de slots por prueba debe ser mayor o igual a 9. El número de slot de prueba por paciente corresponde al número de veces que se le ha realizado una determinada prueba a un paciente en concreto. Es decir, si una prueba se le realiza menos de 9 veces a un paciente, las filas correspondientes a esa prueba de ese paciente en concreto se eliminan de la base de datos.

Según el personal médico, lo ideal es que se le hagan tres pruebas diarias de una misma prueba (hemoglobina, por ejemplo) a una persona al día. Estas medidas para cada paciente se pueden dividir en tres franjas horarias (entendemos que no se realizan exactamente a la misma hora siempre, es por ello por lo que empleamos franjas horarias), a las que llamaremos slots. Se dividirá como se puede ver a continuación en la figura:

Figura 2. Distribución del sistema de slots

Slot 1 -> Realización de la medida: 00:00 – 8:00

Slot 2 -> Realización de la medida: 8:00 – 16:00

Slot 3 -> Realización de la medida: 16:00 – 24:00

En mi caso esta tabla ya ha sido dividida en slots anteriormente, es por ello por lo que no he tenido que tratar la información para este caso y los datos ya venían divididos. Como se puede visualizar en la figura 1, la tabla tiene 5 columnas y, como se explicó anteriormente, se ha realizado un filtrado en MATLAB siguiendo dos criterios:

1. El número de pruebas médicas debe ser mayor de 130.
2. El número de slots por prueba debe ser mayor o igual a 9.

Este filtrado se llevó a cabo en MATLAB, creando previamente una serie de gráficas para poder visualizar el cambio entre una tabla y otra de manera más clara. Al

final del código se genera un archivo Excel con la nueva tabla que, a continuación, empleamos para las nuevas gráficas. Se ha empleado el siguiente código:

```
archivo_excel = 'BBDD_TFG_SERGIO_PINTO.xlsx';
datos = readtable(archivo_excel);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Grafica 1: número de veces que se ha realizado cada prueba
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Obtener valores únicos en la columna NOMBRE_INDICADOR
valores_pruebas = unique(datos.NOMBRE_INDICADOR);
% Inicializar vector para almacenar el número de veces que se ha realizado
cada prueba
num_veces_prueba = zeros(size(valores_pruebas));
% Iterar sobre cada valor único de prueba
for i = 1:numel(valores_pruebas)
    % Contar cuántas veces aparece el valor de prueba en la columna
NOMBRE_INDICADOR
    num_veces_prueba(i) = sum(strcmp(datos.NOMBRE_INDICADOR,
valores_pruebas{i}));
end
% Crear gráfico de barras
figure;
bar(num_veces_prueba);
hold on;
% Agregar línea roja en y = 130
line([0, numel(valores_pruebas)+1], [130, 130], 'Color', 'red', 'LineStyle',
'--');
%Aspecto de la gráfica
xticks(1:numel(valores_pruebas));
xticklabels(valores_pruebas);
xlabel('Tipo de Prueba');
ylabel('Número de Veces Realizadas');
title('Número de Veces que se ha Realizado Cada Prueba');
% Ajustar el aspecto del gráfico
set(gca, 'XTickLabelRotation', 90);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Grafica 2: Comparativa entre UCI y Planta
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calcular el número total de pacientes únicos en la base de datos
pacientes_totales = numel(unique(datos.NHC));
% Filtrar datos para UCI y contar el número de pacientes únicos
pacientes_uci = unique(datos.NHC(strcmp(datos.PLANTA_UCI, 'UCI')));
num_pacientes_uci = numel(pacientes_uci);
% Filtrar datos para Planta y contar el número de pacientes únicos
pacientes_planta = unique(datos.NHC(strcmp(datos.PLANTA_UCI, 'PLANTA')));
num_pacientes_planta = numel(pacientes_planta);
% Filtrar datos para pacientes que han estado en UCI y Planta
pacientes_uci_planta = intersect(pacientes_uci, pacientes_planta);
num_pacientes_uci_planta = numel(pacientes_uci_planta);
% Crear gráfico de barras
figure;
bar([num_pacientes_uci, num_pacientes_planta, num_pacientes_uci_planta]);
xticks(1:3);
xticklabels({'UCI', 'Planta', 'UCI y Planta'});
ylabel('Número de Pacientes');
title('Comparativa entre UCI, Planta y Ambos');
% Ajustar el aspecto del gráfico
```

```
ylim([0, max([num_pacientes_uci, num_pacientes_planta,
num_pacientes_uci_planta]) * 1.1]);
% Añadir texto con el número total de pacientes
text(1.0, pacientes_totales, ['Pacientes tot. = '
num2str(pacientes_totales)], 'HorizontalAlignment', 'right',
'VerticalAlignment', 'bottom');
```

Figura 3. Código para la realización de gráficas

El filtrado se realizó con la implementación del siguiente código:

```
% Nombre del archivo Excel original
archivo_excel = 'BBDD_TFG_SERGIO_PINTO.xlsx';

% Leer los datos del archivo Excel
datos = readtable(archivo_excel);

%%%%%%%%%%%%%%%     Condición 1: Número de slots del que hay una misma
prueba del paciente ingresado > 9 slots (3 dias)  %%%%%%%%%%%%%%%%%%%%%%%
% Leer los números de la columna 'NUMERO_SLOT_PRUEBA_PACIENTE'
numeros_slot_prueba = datos.NUMERO_SLOT_PRUEBA_PACIENTE;
% Inicializar variables
filas_a_eliminar = []; % Almacenará las filas a eliminar
indice_inicio = 1; % Índice donde comienza la secuencia de un contador

% Iterar sobre los números de la columna
for i = 1:numel(numeros_slot_prueba)
    % Verificar si el contador se reinicia
    if numeros_slot_prueba(i) == 1 && i > 1
        % Si el contador se reinicia, verificar si no llegó a 10
        if i - indice_inicio <= 9
            % Si no llegó a 10, almacenar las filas anteriores
            filas_a_eliminar = [filas_a_eliminar, indice_inicio:i-1];
        end
        % Actualizar el índice de inicio para la próxima secuencia
        indice_inicio = i;
    end
end

% Eliminar las filas almacenadas en la variable 'filas_a_eliminar'
datos(filas_a_eliminar, :) = [];

%%%%%%%%%%%%%%%Condicion 2: el número total de pruebas tiene que ser mayor
de 130%    %%%%%%%%%%%%%%%%%%%%%%%%
% Contar el número de veces que se realiza cada prueba
ocurrencias_pruebas = tabulate(datos.NOMBRE_INDICADOR);
pruebas_mas_130_ocurrencias =
unique(ocurrencias_pruebas(cell2mat(ocurrencias_pruebas(:, 2)) > 130, 1));

% Filtrar los datos para mantener solo las filas con pruebas que tienen más
de 130 ocurrencias
datos_filtrados = datos(ismember(datos.NOMBRE_INDICADOR,
pruebas_mas_130_ocurrencias), :);

% Contar el número de líneas eliminadas
total_lineas_eliminadas = numel(filas_a_eliminar);
disp(['Total de líneas eliminadas: ' num2str(total_lineas_eliminadas)]);
```

```
% Guardar los datos filtrados en un nuevo archivo Excel
nombre_excel_resultante = 'BBDD_TFG_filtrada.xlsx';
writetable(datos_filtrados, nombre_excel_resultante);

% Obtener el número de líneas del archivo Excel original y del filtrado
[num_lineas_original, ~] = size(readtable('BBDD_TFG_SERGIO_PINTO.xlsx'));
[num_lineas_filtrado, ~] = size(readtable('BBDD_TFG_filtrada.xlsx'));

% Mostrar el número de líneas en la consola
disp(['El archivo Excel original tiene ' num2str(num_lineas_original) '
líneas.']);
disp(['El archivo Excel filtrado tiene ' num2str(num_lineas_filtrado) '
líneas.']);
disp(['Se han guardado los datos filtrados en el archivo: '
nombre_excel_resultante]);
```

Figura 4. Código para el filtrado de la base de datos

Tras el filtrado de datos siguiendo los criterios mencionados, hemos obtenido un total de 126.716 datos, siendo esta la base de datos que emplearemos para nuestro estudio. En las siguientes gráficas podemos observar dicho cambio con mayor claridad:

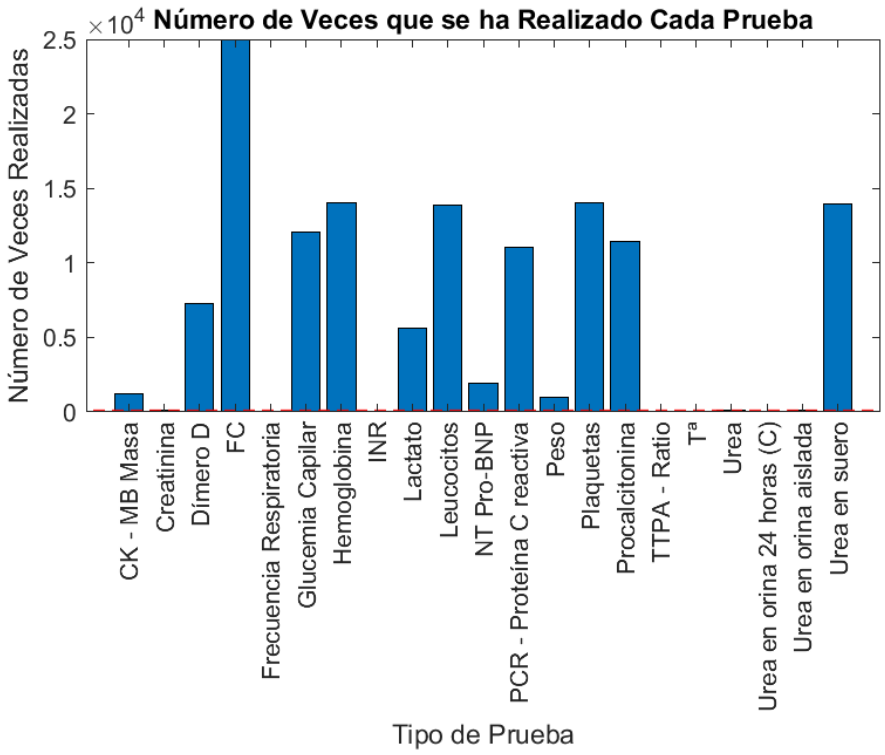

Figura 5. Gráfica Número de veces que se ha realizado una prueba con los datos originales

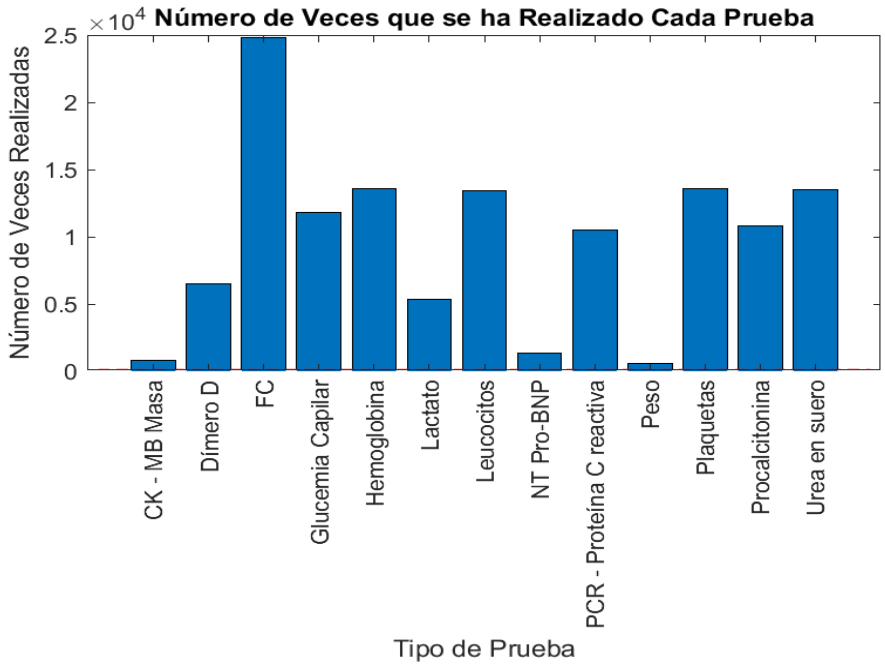

Figura 6. Gráfica Número de veces que se ha realizado una prueba con los datos filtrados

En ambas gráficas se puede observar un interlineado rojo, marcando el valor mínimo necesario para el filtrado, en nuestro caso, 130 pruebas. En la segunda gráfica se puede apreciar el cambio, pasando de 21 pruebas a 13 pruebas totales.

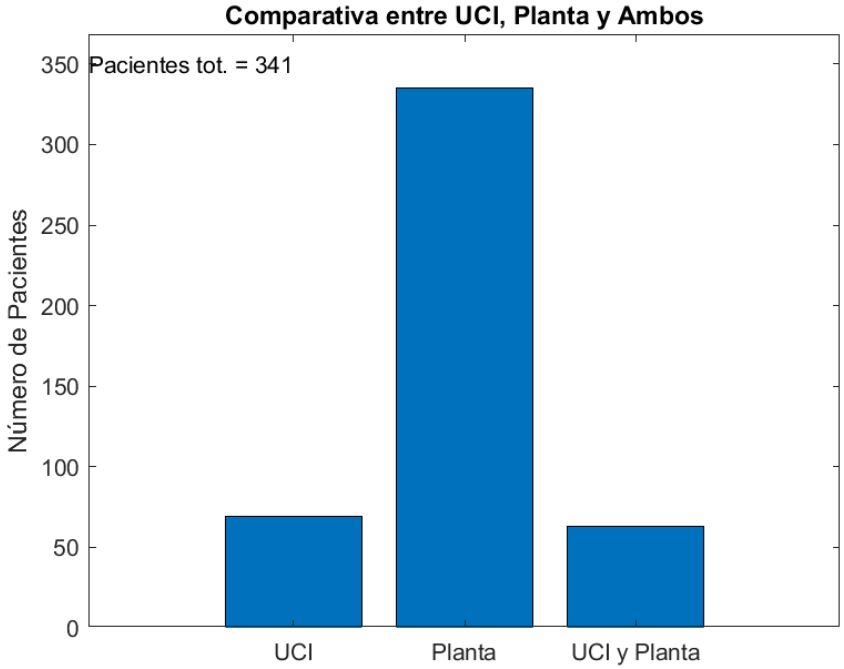

Figura 7. Gráfica Comparativa entre UCI, Planta y ambos con los datos originales

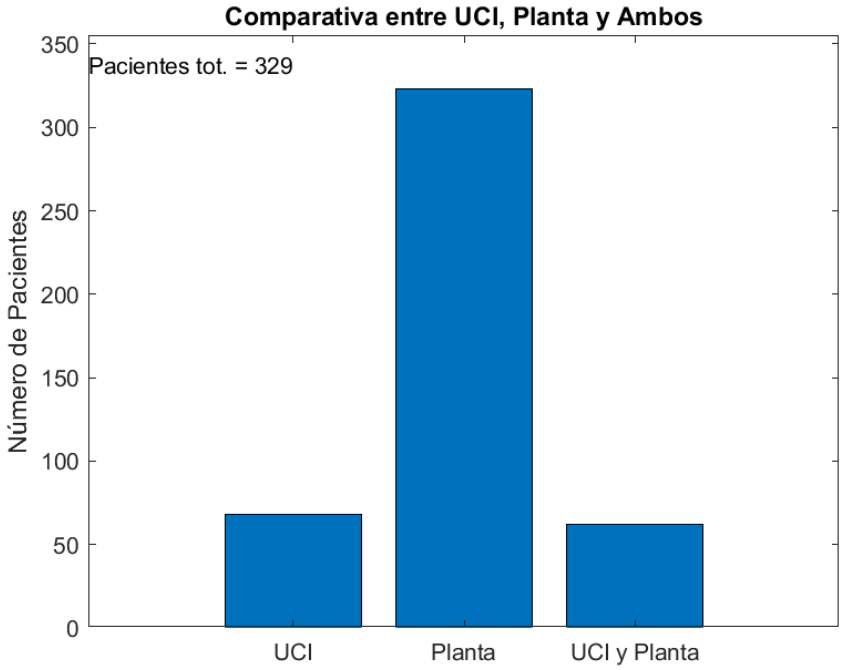

Figura 8. Gráfica Comparativa entre UCI, Planta y ambos con los datos originales

En este segundo par de gráficas se observa la diferencia total de pacientes y su distribución en cuanto a UCI, Planta u ambos (dado que hay casos de pacientes que han estado tanto en UCI como en Planta). Se ve reflejado claramente el cribado en el número de pacientes totales, pasando de 341 a 329 pacientes.

Capítulo 3: Métodos

3.1. Inteligencia Artificial

La inteligencia artificial (IA) se refiere a la capacidad de una máquina para emular funciones cognitivas humanas como percepción, razonamiento, aprendizaje y resolución de problemas [12]. En su esencia, la IA se define como una disciplina en ciencias de la computación que se centra en el desarrollo de algoritmos para software y hardware, con el propósito de ejecutar tareas de manera eficiente en términos de consumo energético y tiempo de ejecución.

Los algoritmos de IA utilizan datos de entrada para clasificar información o hacer predicciones basadas en patrones específicos. Una forma sencilla de entender la IA es imaginarla como un procesador con una gran base de datos, que toma decisiones utilizando estadísticas simples.

La aplicación de la inteligencia artificial abarca una amplia gama de sectores, desde educación y transporte hasta medicina y construcción. En términos técnicos, las máquinas con IA pueden realizar actividades que normalmente requerirían inteligencia humana, como reconocimiento de imágenes, clasificación de datos, mantenimiento predictivo y gestión de contenidos en redes sociales.

Figura 9. Representación de aplicaciones de la Inteligencia Artificial

Ahora que hemos definido claramente qué es la inteligencia artificial, es importante adentrarnos más en esta disciplina para comprender nuestra herramienta de estudio. Dentro del campo de la inteligencia artificial, se encuentra un subcampo conocido como *Machine Learning* (aprendizaje automático), y dentro de este, una técnica más específica denominada *Deep Learning* (aprendizaje profundo). En esencia, podemos visualizarlo como se muestra en la siguiente imagen:

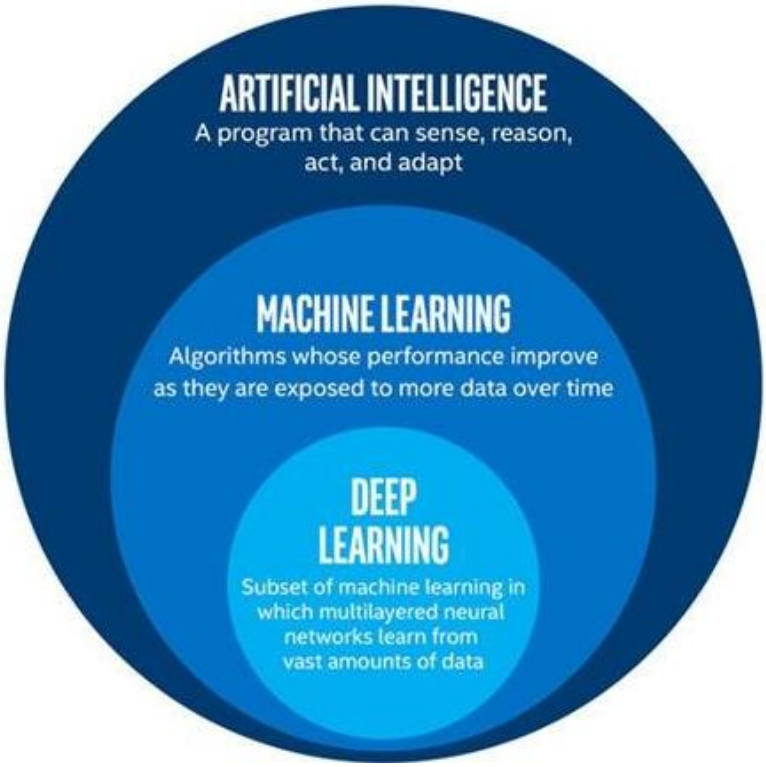

Figura 10. Representación de las capas dentro de la Inteligencia Artificial

3.2. Machine Learning

El *Machine Learning*, o aprendizaje automático, es una subcategoría de la inteligencia artificial que se basa en algoritmos capaces de descubrir patrones recurrentes en conjuntos de datos diversos, como números, palabras, imágenes o estadísticas. La premisa fundamental del *Machine Learning* es permitir que las máquinas aprendan y mejoren su rendimiento en tareas específicas a partir de la detección de estos patrones en los datos.

En el contexto del *Machine Learning*, las máquinas son "entrenadas" por las personas para reconocer estos patrones y hacer predicciones. Este enfoque difiere del funcionamiento tradicional de los programas informáticos, que simplemente ejecutan instrucciones predefinidas. Los algoritmos de inteligencia artificial utilizados en el *Machine Learning* están diseñados para aprender de los datos, corrigiendo errores y ajustándose continuamente a medida que reciben nueva información. La esencia radica en su capacidad para identificar automáticamente características relevantes en los datos y utilizar estas características para realizar predicciones o tomar decisiones. Los algoritmos de *Machine Learning* se entrenan utilizando conjuntos de datos de entrenamiento, donde se les proporciona información previamente etiquetada o estructurada, permitiéndoles aprender y ajustarse para realizar tareas específicas de manera más precisa con datos nuevos.

Esta capacidad de aprendizaje automático tiene un impacto significativo en numerosos aspectos de nuestra vida cotidiana, desde filtros de correo no deseado hasta recomendaciones personalizadas en plataformas como Netflix. El *Machine Learning* aprovecha la vasta cantidad de datos disponibles en la era digital para proporcionar a las empresas perfiles de consumo detallados y mejorar la experiencia del usuario en una variedad de servicios.

Dentro del *Machine Learning* tenemos tres tipos diferentes de aprendizaje: Aprendizaje Supervisado, aprendizaje no supervisado y aprendizaje por refuerzo [13].

3.2.1. Aprendizaje Supervisado

El aprendizaje supervisado es una técnica de aprendizaje automático donde los algoritmos aprenden a partir de datos etiquetados, es decir, datos para los cuales se conoce la salida deseada. En este enfoque, se busca construir un modelo que pueda predecir o clasificar nuevas instancias en función de ejemplos previamente etiquetados. Por ejemplo, en un sistema de detección de spam de correos electrónicos, el algoritmo utiliza un conjunto de datos de correos electrónicos previamente etiquetados como "spam" o "no spam" para identificar patrones

específicos asociados con cada categoría [14]. Estos patrones pueden incluir palabras clave en el asunto, la relación texto/imágenes, o el remitente del correo.

Durante el proceso de aprendizaje supervisado [15], los datos se dividen en un conjunto de entrenamiento y un conjunto de prueba. El conjunto de entrenamiento se utiliza para enseñar al algoritmo a reconocer patrones y relaciones entre las características de entrada y las salidas deseadas. Una vez que el algoritmo ha aprendido estos patrones a partir de los datos de entrenamiento, se evalúa su rendimiento utilizando el conjunto de prueba, donde se comprueba la calidad de las predicciones realizadas con datos no vistos previamente. Los algoritmos de aprendizaje supervisado son fundamentales en problemas de clasificación (donde se asigna una etiqueta a una nueva instancia) y regresión (donde se predice un valor numérico).

Algunos ejemplos populares de algoritmos de aprendizaje supervisado incluyen las Máquinas de Vectores de Soporte (SVM), que buscan encontrar el hiperplano óptimo que separa las clases en un espacio multidimensional; los árboles de decisión, que utilizan estructuras de árbol para tomar decisiones basadas en múltiples características; y los clasificadores bayesianos como *Naïve Bayes*, que aplican el teorema de Bayes para realizar inferencias probabilísticas sobre las clases de las instancias.

3.2.2. Aprendizaje No Supervisado

En contraste con el aprendizaje supervisado, el aprendizaje no supervisado se centra en el estudio de datos no etiquetados, es decir, datos para los cuales no se conoce la salida deseada. Este enfoque permite explorar la estructura subyacente de los datos y descubrir patrones o relaciones ocultas sin la necesidad de intervención humana para proporcionar etiquetas.

Los algoritmos de aprendizaje no supervisado se utilizan comúnmente en tareas como *clustering* (agrupamiento) [16],donde el objetivo es identificar grupos naturales o subconjuntos coherentes dentro de los datos. Por ejemplo, un algoritmo

de *clustering* puede agrupar automáticamente clientes en segmentos similares en función de sus comportamientos de compra.

Otra aplicación importante del aprendizaje no supervisado es la reducción de la dimensionalidad, donde el objetivo es representar los datos en un espacio de menor dimensión manteniendo la información relevante. Esto facilita la visualización y comprensión de los datos, así como la eliminación de características redundantes o irrelevantes.

Algunos algoritmos comunes en aprendizaje no supervisado incluyen *K-Means* para *clustering*, que agrupa datos en k grupos basados en similitudes; y PCA (Análisis de Componentes Principales), que encuentra las direcciones de máxima variabilidad en los datos para proyectarlos en un espacio de menor dimensionalidad. Estos métodos son ampliamente utilizados en análisis exploratorio de datos, reconocimiento de patrones y segmentación de mercado.

3.2.3. Aprendizaje por refuerzo

El aprendizaje por refuerzo [17] es un enfoque en el campo del aprendizaje automático donde un agente aprende a tomar decisiones óptimas a través de la interacción con un entorno. En este proceso, el agente toma una serie de acciones en un entorno dinámico y recibe recompensas o castigos en función de esas acciones. El objetivo del agente es aprender a maximizar las recompensas acumuladas a lo largo del tiempo. En el aprendizaje por refuerzo, el agente no tiene acceso a ejemplos de entrenamiento con respuestas correctas, en cambio, el agente explora el entorno, realiza acciones y aprende de las consecuencias de esas acciones. Cada vez que el agente toma una acción, el entorno responde con una señal de refuerzo que indica si la acción fue beneficiosa o no. El proceso de aprendizaje implica que el agente ajuste su estrategia en función de la retroalimentación recibida. El objetivo final es que el agente aprenda una política de toma de decisiones que maximice la recompensa esperada a lo largo del tiempo, incluso cuando el entorno es desconocido o cambia dinámicamente. Este tipo de aprendizaje se emplea en diversas aplicaciones, como el

reconocimiento facial, el diagnóstico médico y juegos estratégicos como el ajedrez o el *Go*.

Ahora que hemos explorado el *Machine Learning* y sus diversos enfoques de aprendizaje supervisado, no supervisado y por refuerzo, es el momento de sumergirnos en una técnica aún más potente: el *Deep Learning*.

3.3. Deep Learning

Para entender la relación entre la inteligencia artificial, el *Machine Learning* y el *Deep Learning*, es esencial explorar cómo estos conceptos se interrelacionan y diferencian entre sí. El *Machine Learning* se sitúa como un subcampo dentro de la inteligencia artificial, mientras que las redes neuronales representan un subcomponente del *Machine Learning*. El *Deep Learning*, por otro lado, representa un avance significativo en el concepto de redes neuronales, incorporando mayor complejidad y profundidad en su estructura [18].

Una diferencia fundamental entre las redes neuronales y el *Deep Learning* [19] radica en la profundidad de las capas. Mientras que una red neuronal puede tener varias capas, el *Deep Learning* se refiere específicamente a redes neuronales con más de tres capas, incluyendo las capas de entrada y salida. También se incluyen la sofisticación de las redes de *Deep Learning*, que tienen múltiples capas y son más complejas que las redes neuronales tradicionales. Los sistemas de *Deep Learning* tienden a ser más eficientes en la ejecución de tareas en comparación con las redes neuronales convencionales. Sin embargo, la complejidad adicional de las redes de *Deep Learning* requiere tiempos de entrenamiento más largos y recursos computacionales más potentes.

Las estructuras neuronales del aprendizaje buscan emular el funcionamiento del cerebro humano mediante una combinación de entradas de datos, ponderaciones y sesgos. Estos componentes colaboran para identificar, clasificar y describir de manera precisa los objetos presentes en los datos.

Las redes neuronales profundas están compuestas por múltiples capas de nodos interconectados, cada uno de ellos construido sobre la capa precedente para mejorar y refinar las predicciones o clasificaciones. Las capas de entrada y salida de una red neuronal profunda son conocidas como capas visibles. La capa de entrada es donde el modelo de aprendizaje profundo toma los datos para su procesamiento, mientras que la capa de salida es donde se genera la predicción o clasificación final.

Lo anterior describe la estructura básica de una red neuronal profunda en términos simplificados. Sin embargo, los algoritmos de aprendizaje profundo son altamente complejos y existen diversos tipos de redes neuronales diseñadas para resolver problemas específicos o manejar conjuntos de datos particulares. En este trabajo se va a trabajar concretamente con 5 tipos de arquitecturas diferentes, que explicaré a continuación:

3.3.1. Redes Neuronales Recurrentes

Las redes neuronales recurrentes (RNN) [20] son la arquitectura base sobre la que se implementan el resto. Una red neuronal recurrente no tiene una estructura de capas definida, sino que permiten conexiones arbitrarias entre las neuronas, incluso pudiendo crear ciclos, con esto se consigue crear la temporalidad, permitiendo que la red tenga memoria. En este trabajo vamos a trabajar con tres redes recurrentes diferentes: GRU, LSTM y BiLSTM

3.3.2. LSTM (Long Short-Term Memory)

Las redes LSTM (*Long Short-Term Memory*) [21] están compuestas por unidades LSTM. La unidad de memoria LSTM contiene tres puertas que controlan el modo en que la información fluye dentro o fuera de la unidad.

- Puerta de **entrada** controla cuando la información nueva puede entrar en la memoria.
- Puerta del **olvido** controla cuando se olvida una parte de la información, lo que permite a la celda discriminar entre datos importantes y superfluos, dejando así sitio para nuevos datos.

- Puerta de **salida** controla cuando se utiliza en el resultado de los recuerdos almacenados en la celda.

La celda dispone de un mecanismo de optimización de las ponderaciones basado en el error de salida de la red resultante, que controla cada puerta.

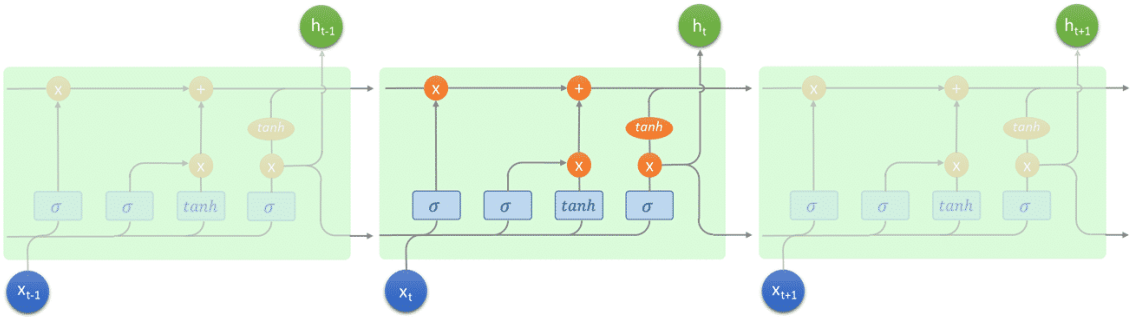

Figura 11. Ejemplo de una arquitectura LSTM

3.3.3. BiLSTM

Existe una variedad de esta arquitectura, llamada BiLSTM. Una capa de LSTM bidireccional (BiLSTM) es una capa de RNN que aprende dependencias bidireccionales a largo plazo entre unidades de tiempo de series de tiempo y datos secuenciales. Estas dependencias pueden resultar útiles cuando quiera que la RNN aprenda de series de tiempo completas en cada unidad de tiempo [22]. En términos sencillos, una red neuronal BiLSTM es una arquitectura diseñada para procesar secuencias en ambas direcciones: de adelante hacia atrás y de atrás hacia adelante. Incorpora unidades LSTM en dos direcciones para capturar información contextual del pasado y del futuro en una secuencia.

3.3.4. GRU (*Gated Recurrent Unit*)

Las redes GRU (*Gated Recurrent Unit*) [23] están compuestas por unidades GRU. La unidad de memoria GRU contiene dos puertas que controlan el modo en que la información fluye dentro o fuera de la unidad.

- Puerta de **actualización** y una puerta de reajuste. La puerta de actualización indica cuánto del contenido de las celdas anteriores hay que mantener.

- Puerta de **reajuste** define cómo incorporar la nueva entrada con los contenidos anteriores de la celda.

Las redes GRU se caracterizan por su simplicidad en comparación con las LSTM. En primer lugar, tienen menos parámetros en su arquitectura, lo que las hace más ligeras y eficientes en términos de almacenamiento y computación. Además, las GRU carecen de una puerta de salida como la que tienen las LSTM, lo que simplifica su estructura y puede contribuir a una mejor generalización, que se traduce en un entrenamiento más rápido. Dependiendo del volumen de datos proporcionado, una red puede ser más efectiva que otra.

3.3.5. CNN con LSTM

La arquitectura CNN (*Convolutional Neural Network*) con LSTM es una combinación potente para abordar problemas donde los datos tienen tanto características estructurales como secuenciales. En el caso de los datos numéricos o matriciales, la CNN se especializa en la extracción de características locales relevantes dentro de estos datos. Las capas convolucionales de una CNN aplican filtros para detectar patrones locales en los datos de entrada, como tendencias o anomalías en los datos numéricos, que son cruciales para comprender el comportamiento subyacente de los datos.

La estructura híbrida CNN-LSTM es particularmente útil en contextos donde los datos no solo tienen una estructura espacial o estructural, sino también una dimensión temporal. Mientras que la CNN se encarga de capturar las características locales en cada instante de tiempo de los datos, la LSTM modela la secuencia temporal de estas características extraídas. Esto es especialmente útil en tareas como la predicción de series temporales complejas, el análisis de secuencias de datos numéricos, o el procesamiento de texto donde es crucial capturar tanto la estructura de las palabras o frases como la evolución del contexto a lo largo de una secuencia.

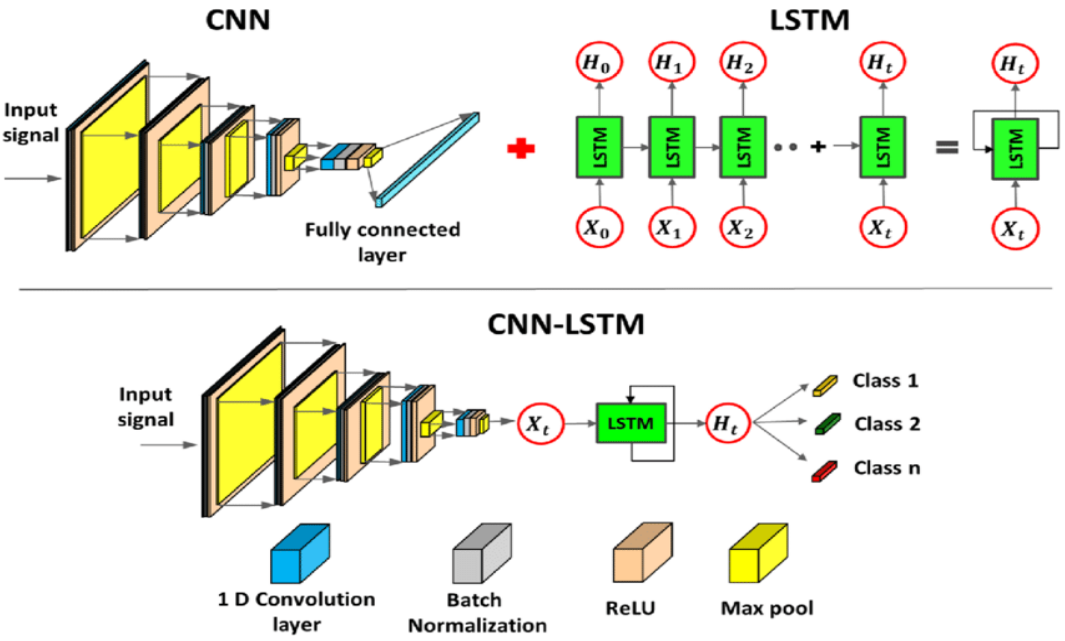

Figura 12. Ejemplo de una arquitectura CNN con LSTM

3.3.5. Transformers

La red neuronal *Transformer* representa una innovadora arquitectura de aprendizaje automático que surgió como una alternativa eficaz a las redes neuronales recurrentes (RNN) para tareas de procesamiento de lenguaje natural y secuencial. A diferencia de los modelos secuenciales anteriores, los Transformers eliminan la necesidad de recurrencia al permitir que todas las palabras (tokens) de una secuencia sean procesadas simultáneamente mediante un mecanismo de autoatención [24]. La arquitectura *Transformer* se divide en dos componentes esenciales: el *Encoder* y el *Decoder*. El *Encoder* toma una secuencia de entrada y la transforma en una serie de representaciones latentes utilizando capas de autoatención y redes *feed-forward*. Cada palabra de la entrada se convierte en un vector de características mediante un proceso de *embedding*, al cual se añade una codificación posicional para mantener el orden relativo de las palabras en la secuencia. Luego, múltiples capas de autoatención en paralelo se encargan de procesar la información de manera eficiente, identificando las relaciones entre las diferentes palabras y extrayendo características importantes. La salida final del *Encoder* es una representación contextualizada de la entrada, que captura las dependencias y el contexto de la secuencia.

Por otro lado, el *Decoder* recibe esta representación latente generada por el *Encoder* y la utiliza para producir una secuencia de salida. Similar al *Encoder*, el *Decoder* emplea capas de autoatención y redes *feed-forward*, pero además incluye una capa de atención cruzada que permite al modelo acceder a la información del *Encoder* para realizar la decodificación. Durante el proceso de decodificación, el modelo genera una salida palabra por palabra, tomando como entrada las representaciones contextuales previamente calculadas y las salidas generadas anteriormente. La arquitectura del *Transformer* permite un entrenamiento y una inferencia altamente paralelos, lo que mejora significativamente la eficiencia y la escalabilidad del modelo en comparación con las RNN tradicionales.

La figura adjunta muestra de manera esquemática la arquitectura general de un *Transformer*, destacando los bloques de *Encoder* y *Decoder*, así como las capas de autoatención y redes *feed-forward* que los componen. Esta visualización proporciona una comprensión clara de cómo funciona el modelo, capturando las interacciones entre las palabras dentro de una secuencia y facilitando la generación de secuencias de salida basadas en el contexto aprendido durante el entrenamiento.

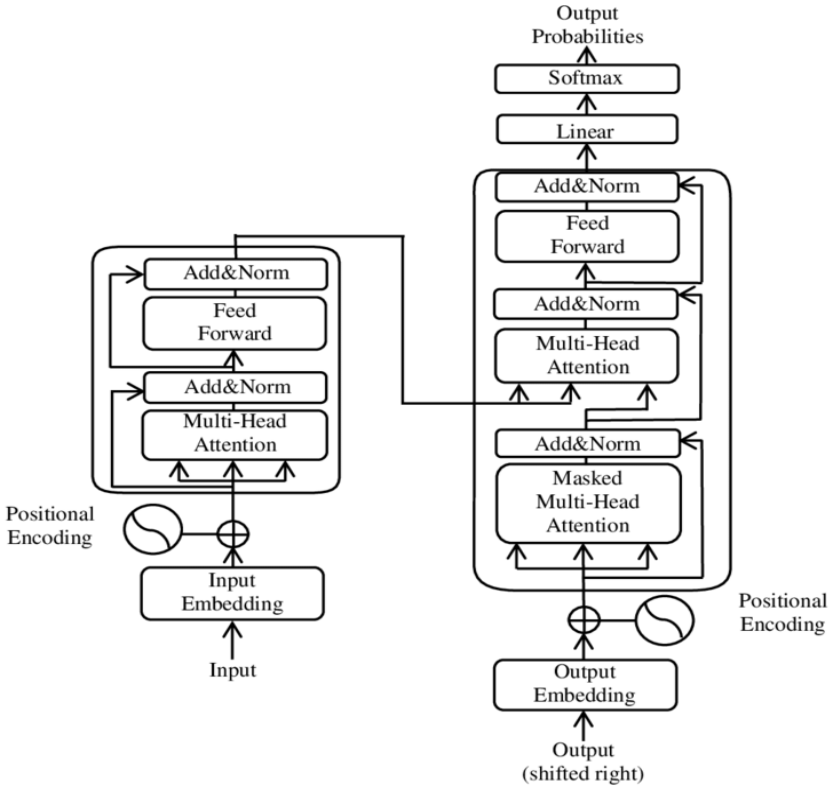

Figura 13. Ejemplo de una arquitectura Transformers

3.4. Explicación del ensamble realizado

Al realizar diferentes experimentos con un modelo de una sola red neuronal, los resultados pueden variar considerablemente de un experimento a otro debido a que no todos los modelos capturan la misma información contenida en los datos de entrenamiento. Esto subraya la limitación de depender únicamente de un modelo de una sola red neuronal, ya que podría omitir información relevante para la predicción precisa.

Para abordar esta variabilidad, se emplea el método de conjunto. Este enfoque implica la aplicación simultánea de múltiples modelos con una sola red neuronal de forma independiente, es decir, varios modelos que operan en paralelo. Posteriormente, se calcula el promedio aritmético de los resultados de todos los modelos. Este conjunto resultante mejora la precisión al integrar las predicciones de cada modelo, favoreciendo aquellos que se acercan más a la media de las predicciones [25].

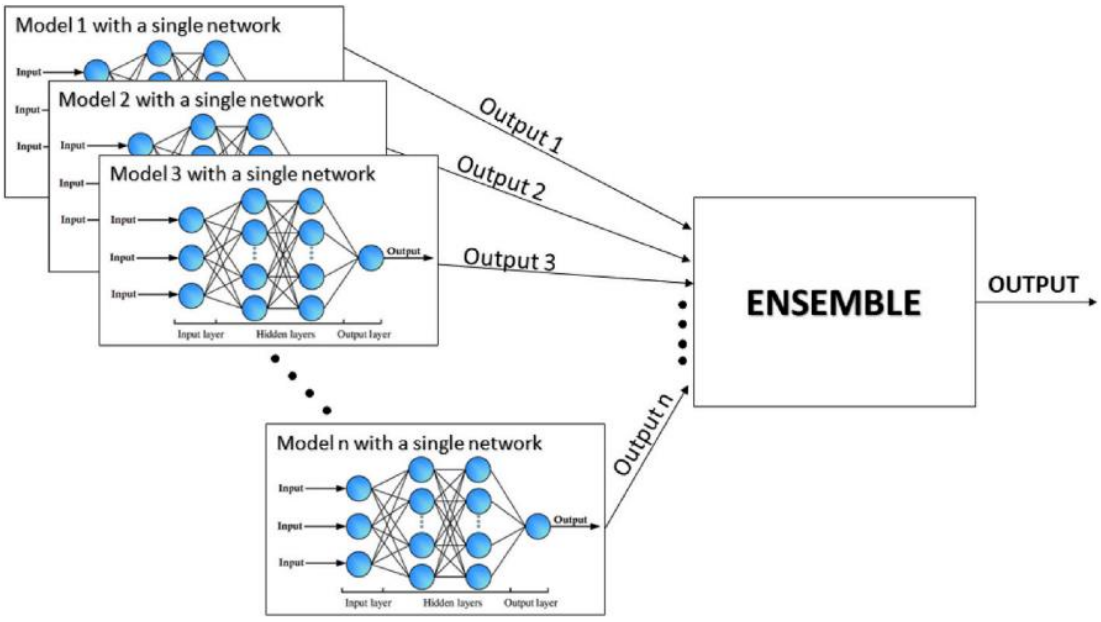

Figura 14. Ejemplo del ensamble realizado

En el caso de mi Trabajo de Fin de Grado, se ha implementado un ensamble de 10 redes neuronales para estabilizar los resultados. Aunque la fiabilidad aumenta con más redes en paralelo, también se incrementa la demanda computacional. Dado el

balance entre recursos y resultados, optar por 10 redes ha demostrado ser suficiente para estabilizar los resultados con un margen de error reducido, evitando la sobrecarga computacional que implicaría utilizar un número mucho mayor de modelos.

Capítulo 4: Metodología experimental

4.1. Metodología experimental

Para llevar a cabo este Trabajo de Fin de Grado, se empleará una metodología experimental [26] reconocida por su capacidad para evaluar hipótesis y teorías mediante la observación y análisis sistemático de fenómenos. Esta metodología comienza con la formulación de hipótesis, donde se establecen las preguntas clave que guiarán la investigación. Las hipótesis, siendo específicas y verificables, orientan la siguiente fase crucial del análisis de datos y diseño experimental. Aquí se planifica minuciosamente cómo se ejecutarán los estudios, incluyendo la definición precisa de variables, métodos de recolección de datos y protocolos experimentales.

Finalmente, en la Interpretación de Resultados, se evalúa la coherencia de los hallazgos con las hipótesis iniciales y se exploran las implicaciones teóricas y prácticas de los resultados obtenidos. Es en este último paso donde se desarrollará el documento final del proyecto.

La metodología seguida para el desarrollo de este TFG se puede ver en la Figura XX y cada una de las tareas realizadas para llevarla a cabo se explican a continuación:

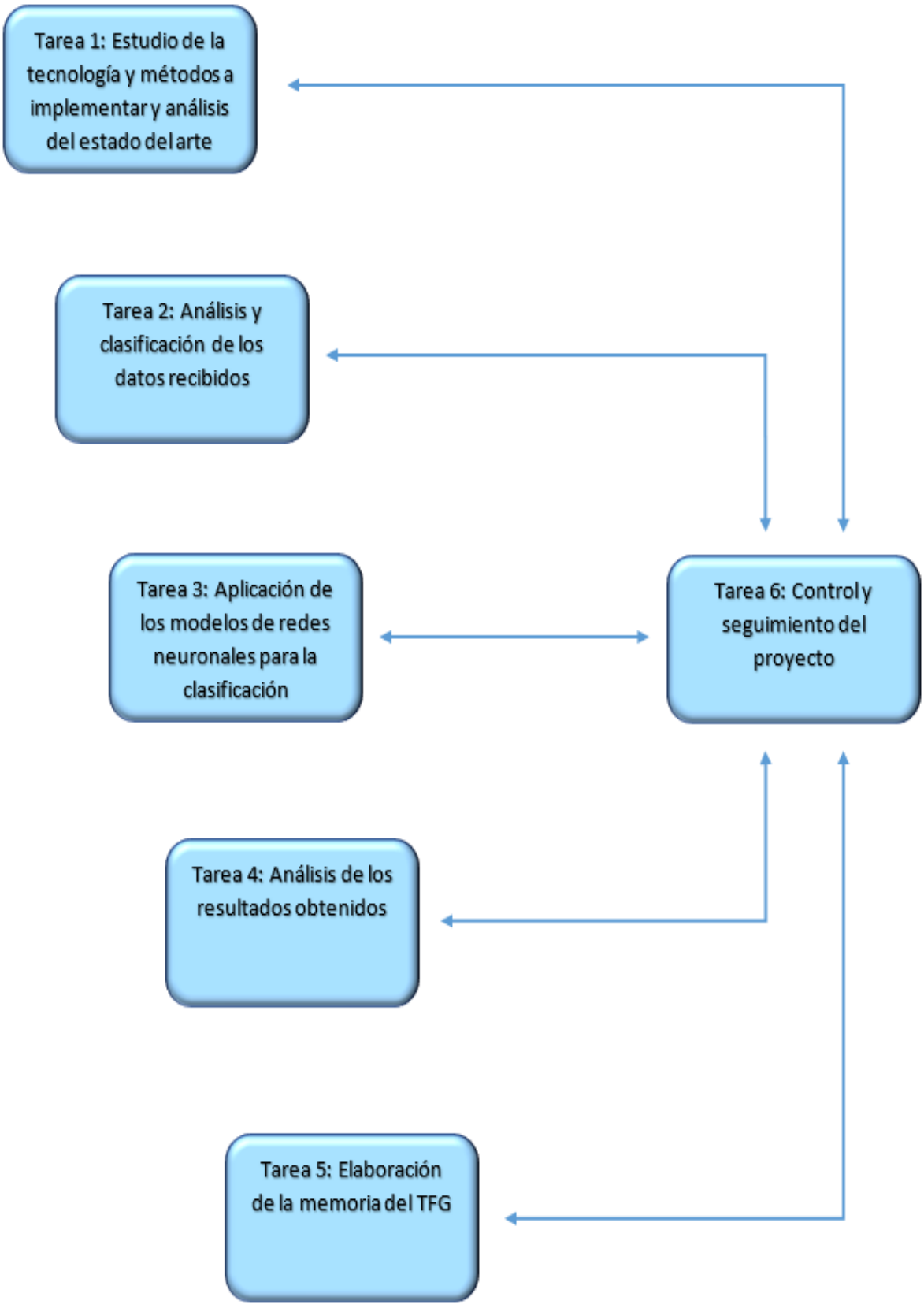

Figura 15. Tareas llevadas a cabo en el trabajo

Tarea 1: Estudio de la tecnología y métodos a implementar y análisis del estado del arte

En esta fase inicial, se realizó un estudio del estado del arte en relación con la COVID-19, inteligencia artificial (IA) y tanto *Machine* como *Deep Learning*. Se indagó específicamente en el área de *Deep Learning*, explorando diferentes arquitecturas como GRU, LSTM, BiLSTM y Transformers. El objetivo fue identificar y comprender las características y aplicaciones de cada arquitectura dentro del contexto del procesamiento de datos relacionados con la pandemia.

Tarea 2: Análisis y clasificación de los datos recibidos

Los datos utilizados en este estudio fueron proporcionados por el Hospital Materno de Las Palmas de Gran Canaria y consisten en medidas reales tomadas a pacientes con el virus SARS-CoV-2. Se realizó un análisis de estos datos, identificando variables relevantes y filtrando las observaciones según criterios mencionados anteriormente en el documento.

Tarea 3: Aplicación de los modelos de redes neuronales para la clasificación

En este bloque se aplicarán diversas arquitecturas de redes neuronales, incluyendo GRU, LSTM, BiLSTM y Transformers, para la clasificación de los datos según criterios médicos relevantes. Cada arquitectura se implementará con diferentes configuraciones y parámetros para evaluar su desempeño en la tarea de clasificación. Los resultados obtenidos serán evaluados utilizando una matriz de confusión, que proporciona métricas como especificidad, sensibilidad, precisión y exactitud. La elección final de la arquitectura óptima dependerá de la interpretación de estos resultados, considerando factores como la eficacia, los costos computacionales y el tiempo de procesamiento.

Tarea 4: Análisis de los resultados obtenidos

En esta etapa crucial, se llevará a cabo un análisis exhaustivo de los resultados obtenidos de las diferentes arquitecturas de redes neuronales aplicadas en el Bloque 3 para la clasificación de datos relacionados con la COVID-19.

El objetivo principal es identificar el modelo final más adecuado para la clasificación de pacientes en categorías relevantes como "UCI" o "Planta". Se compararán y contrastarán los resultados de cada arquitectura, teniendo en cuenta aspectos como la eficacia en la identificación de casos críticos y la capacidad para minimizar falsos positivos o negativos.

Se realizará un análisis detallado de las fortalezas y debilidades de cada modelo, considerando también el tiempo de procesamiento y los recursos computacionales requeridos. La decisión sobre el modelo final se tomará en base a una evaluación integral de estos factores, con ayuda de mi tutor Sergio Celada, garantizando la selección de la arquitectura óptima que maximice la precisión diagnóstica y la eficiencia operativa en el contexto médico.

Tarea 5: Elaboración de la memoria del TFG

En este apartado se realizó el documento presentado acerca de la investigación del proyecto.

Tarea 6: Control y seguimiento del proyecto

Se implementará un sistema de control y seguimiento que incluirá la planificación temporal del proyecto, reuniones periódicas semanales con mi tutor, con una duración de 1 hora, donde se revisará el estado de la planificación, las tareas realizadas y las tareas en curso, en las cuales se presentará la documentación correspondiente. Este sistema permitirá mantener un avance organizado y ajustar estrategias según sea necesario, garantizando que el proyecto se desarrolle de manera eficiente y cumpla con los objetivos planteados.

4.2. Lenguaje de programación empleado

En el presente proyecto se ha optado por implementar el desarrollo en Matlab, un entorno de programación ampliamente reconocido y utilizado en el ámbito académico y de investigación [27].

Matlab es una plataforma potente y versátil que permite la implementación de una amplia gama de algoritmos y modelos, incluyendo redes neuronales. Su capacidad para manejar matrices y realizar cálculos numéricos complejos de manera eficiente lo convierte en una opción ideal para el desarrollo de modelos de aprendizaje profundo, como las incluidas en este proyecto. Además, Matlab cuenta con una amplia biblioteca de funciones y herramientas específicas para el diseño, entrenamiento y validación de redes neuronales, lo que facilita enormemente el proceso de desarrollo. Además, está disponible de manera gratuita para los estudiantes de nuestra universidad, lo que representa una ventaja significativa en términos de accesibilidad y costo.

El lenguaje de Matlab, aunque puede considerarse complejo para quienes no tienen experiencia previa, resulta accesible y muy eficaz para quienes tienen conocimientos básicos. Con una pequeña formación previa, como es mi caso, se proporciona conocimientos suficientes para desarrollar y comprender algoritmos en Matlab con solvencia.

Por último, Matlab es ampliamente utilizado en la comunidad académica y de investigación. Existen numerosos trabajos y proyectos de investigación realizados en Matlab [28], lo que proporciona una rica fuente de referencia y apoyo. La extensa documentación y la activa comunidad de usuarios de Matlab facilitan la resolución de problemas y la implementación de soluciones eficientes.

4.3. Matriz de confusión

En este estudio, utilizaremos un modelo con dos opciones para clasificar los pacientes según su ubicación: en UCI o en planta. Para la elección de la mejor red neuronal con los mejores parámetros (que explicaré más adelante), es necesario realizar múltiples procesados, con el objetivo encontrar la óptima. Este criterio lo

vamos a obtener gracias a la generación de una matriz de confusión cada vez que se entrena a la red neuronal.

Para evaluar los resultados de las pruebas, utilizaremos la conocida como tabla de contingencia, matriz de error o matriz de confusión [29] (nos referiremos de esta manera a la misma en este documento). Esta herramienta permite visualizar el desempeño de un algoritmo de aprendizaje supervisado, proporcionando información sobre el porcentaje de aciertos y errores del modelo al procesar la base de datos. Cada columna de la matriz representa las predicciones del modelo para cada clase, mientras que cada fila corresponde a las instancias de la clase real. Así, la matriz de confusión nos ofrece una visión clara de los tipos de aciertos y errores que presenta nuestro modelo durante el proceso de aprendizaje [30]. En la siguiente tabla se muestra gráficamente la estructura de la matriz de confusión.

MATRIZ DE CONFUSIÓN		**Predicción**	
		Positivo	Negativo
Reales	Positivo	Verdadero Positivo True Positive (TP)	Falso Negativo False Negative (FN)
	Negativo	Falso Positivo False Positive (FP)	Verdadero Negativo True Negative (TN)

Figura 16. Ejemplo estructura de una matriz de confusión

- **Verdadero positivo:** El valor real es positivo y la prueba predijo también que era positivo.
- **Verdadero negativo:** El valor real es negativo y la prueba predijo también que el resultado era negativo.
- **Falso negativo:** El valor real es positivo, y la prueba predijo que el resultado es negativo.

- **Falso positivo:** El valor real es negativo, y la prueba predijo que el resultado es positivo.

A continuación, se presenta un resumen de los indicadores [23] que se obtienen a partir de una matriz de confusión, fundamentales para evaluar el desempeño de un modelo de clasificación:

		Predicción		
		Planta	UCI	
Valor Real	Planta	El valor es Planta y se predijo Planta (TP)	El valor e Planta y se predijo UCI (FN)	Sensibilidad
	UCI	EL valor es UCI y se predijo Planta (FP)	El valor es UCI y se predijo UCI (TN)	Especifidad
		Precisión	Tasa de verdaderos negativos	Tasa de acierto (Accuracy)

Figura 17. Matriz de confusión aplicada a mis datos

- Tasa de acierto (*Accuracy*): Representa el porcentaje de predicciones correctas realizadas por el modelo de clasificación. Se calcula mediante la fórmula:

$$Tasa\ de\ acierto = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precisión: Indica de todas las predicciones que fueron etiquetadas como PLANTA, cuántas fueron correctas. Su fórmula es:

$$Precisión = \frac{TP}{TP + FP}$$

- Especificidad: Mide de todos los valores que son realmente UCI, cuántos se predicen correctamente como UCI. La fórmula es:

$$Especifidad = \frac{TN}{TN + FP}$$

- Sensibilidad (*Recall*): Indica de todos los valores que son realmente PLANTA, cuántos se predicen correctamente como PLANTA. Se calcula mediante:

$$Sensibilidad = \frac{TP}{TP + FN}$$

- Tasa de verdaderos negativos *(True Negative Rate)*: Mide, de todos los valores que no son PLANTA, cuántos se predicen correctamente como no PLANTA. La fórmula es:

$$Tasa\ de\ verdaderos\ negativos = \frac{TN}{TN + FN}$$

Capítulo 5: Implementación del modelo

5.1. Generación de la Matriz de Confusión

En primer lugar, se presentará el código utilizado para la generación de la matriz de confusión, el cual es aplicable a todas las arquitecturas implementadas. Posteriormente, se proporcionará el código específico de cada modelo, subrayando las diferencias en aquellos casos donde las variaciones no sean evidentemente claras. Cabe destacar que todo el proceso de codificación, entrenamiento, generación de matrices y demás procedimientos se ha llevado a cabo utilizando el software MATLAB.

```
%Este apartado tiene como objetivo presentar la generación y análisis de una
matriz de confusión.

      vector_validacionTotal2=mean(vector_validacionTotal);
      vector_predicionTotal2=mean(vector_predicionTotal);

      vector_validacionTotal3(vector_validacionTotal2 > 0.5) = 1;
      vector_validacionTotal3(vector_validacionTotal2 <= 0.5) = 0;

      vector_predicionTotal3(vector_predicionTotal2 > 0.5) = 1;
      vector_predicionTotal3(vector_predicionTotal2 <= 0.5) = 0;

      vector_predicionTotal4=zeros(2, length(vector_predicionTotal3));
      vector_predicionTotal4(1, :) = vector_predicionTotal3 == 1;
      vector_predicionTotal4(2, :) = ~vector_predicionTotal4(1, :);

      vector_validacionTotal4=zeros(2, length(vector_validacionTotal3));
      vector_validacionTotal4(1, :) = vector_validacionTotal3 == 1;
      vector_validacionTotal4(2, :) = ~vector_validacionTotal4(1, :);

figure
plotconfusion(vector_validacionTotal4, vector_predicionTotal4)
```

Figura 18. Código generación de matriz de confusión

El código presentado tiene como objetivo generar y analizar una matriz de confusión para evaluar el rendimiento de un modelo de clasificación. Inicialmente, se calculan los valores promedio de los vectores de validación y predicción, los cuales se binarían utilizando un umbral de 0.5: los valores superiores a 0.5 se asignan a 1, y los valores iguales o inferiores a 0.5 se asignan a 0. Posteriormente, se crean matrices binarias para las predicciones y validaciones, estructuradas en dos filas donde la primera fila contiene los valores iguales a 1 y la segunda fila contiene los valores

complementarios. Finalmente, se utiliza la función *plotconfusion* para generar y visualizar la matriz de confusión.

5.2. Implementación red neuronal GRU

```
%Se carga la base de datos
load T.mat
tic
    % Identificar filas con valores NaN en la columna "VALOR"
    nan_rows = isnan(cell2mat(T.VALOR));
    % Eliminar filas con valores NaN de la tabla T
    T = T(~nan_rows, :);
     % Extraer columnas "VALOR" y "PLANTA_UCI" de la tabla T
    VALOR = T.VALOR;
    PLANTA_UCI = categorical(T.PLANTA_UCI);
    % Dividir datos en conjuntos de entrenamiento y validación
    numObservations = numel(VALOR);
    numTrain = floor(0.6 * numObservations);

    XTrain = VALOR(1:numTrain);
    YTrain = PLANTA_UCI(1:numTrain);

    XValidation = VALOR(numTrain+1:end);
    YValidation = PLANTA_UCI(numTrain+1:end);

    %%
for i=1:10,
        % Crear arquitectura de red GRU
        inputSize = 1;
        numHiddenUnits = 100;
        numClasses = numel(categories(PLANTA_UCI));
        layers = [ ...
            sequenceInputLayer(inputSize)
            gruLayer(numHiddenUnits,'OutputMode','last')
            fullyConnectedLayer(numClasses)
            softmaxLayer
            classificationLayer];
        % Especificar opciones de entrenamiento
        options = trainingOptions('adam', ...
            'MaxEpochs',5, ...
            'InitialLearnRate',0.001, ...
            'GradientThreshold',1, ...
            'Shuffle','every-epoch', ...
            'ValidationData',{XValidation,YValidation}, ...
            'Plots','none', ...
            'Verbose',false);
        % Entrenar la red GRU
        net = trainNetwork(XTrain,YTrain,layers,options);
        % Realizar predicciones
        YPred = classify(net, XValidation);

        vector_predicion = string(cellstr(YPred'));
        vector_validacion = string(cellstr(YValidation'));

        vector_validacionTotal(i, strcmpi('PLANTA', vector_validacion)) = 1;
        vector_validacionTotal(i, ~strcmpi('PLANTA', vector_validacion)) = 0;
```

```
        vector_predicionTotal(i, strcmpi('PLANTA', vector_predicion)) = 1;
        vector_predicionTotal(i, ~strcmpi('PLANTA', vector_predicion)) = 0;

end
```

Figura 19. Código generación red neuronal GRU

El código en MATLAB tiene como propósito cargar una base de datos, preparar los datos y entrenar múltiples redes neuronales, en este caso GRU, en paralelo para estabilizar los resultados y realizar predicciones. Inicialmente, se carga la base de datos desde un archivo denominado *T.mat*, que se trata de la base de datos cambiada a una matriz mencionada anteriormente. La primera operación sería eliminar las filas que contienen valores *NaN* en la columna "VALOR" para asegurar la integridad de los datos. Posteriormente, se extraen las columnas "VALOR" y "PLANTA_UCI", convirtiendo esta última a un tipo categórico. Los datos se dividen en conjuntos de entrenamiento y validación, utilizando para este ejemplo el 60% de las observaciones para el entrenamiento y el 40% restante para la validación.

El entrenamiento de las redes GRU se realiza en un bucle que se ejecuta diez veces. Cada iteración configura una red GRU con una capa de entrada secuencial, una capa GRU con 100 unidades ocultas (para este ejemplo), una capa completamente conectada, una capa *softmax* y una capa de clasificación. Las opciones de entrenamiento se establecen utilizando el optimizador Adam, que ha sido el que mejores resultados no ha dado con todas las redes neuronales, por lo que lo hemos establecido como optimizador base, con parámetros específicos para el número de épocas, la tasa de aprendizaje y otros aspectos relacionados con la validación y el procesamiento de datos. La red se entrena con los datos de entrenamiento y realiza una clasificación sobre los datos de validación para obtener las predicciones, repitiendo el proceso diez veces (ensamblado) para estabilizar los resultados y asegurar la robustez del modelo. Más adelante, se generaría la matriz de confusión, explicada en el punto anterior.

5.2. Implementación red neuronal LSTM

```
%Se carga la base de datos
load T.mat
tic
```

```
    % Puede ser que algunas tablas tengan valores NaN. Lo primero de todo
    % sería eliminar esas filas y se hace así:

    % Identificar filas con valores NaN en la columna "VALOR"
    nan_rows = isnan(cell2mat(T.VALOR));
    % Eliminar filas con valores NaN de la tabla T
    T = T(~nan_rows, :);

%%
       % Extraer columnas "VALOR" y "PLANTA_UCI" de la tabla T
    VALOR = T.VALOR;
    PLANTA_UCI = categorical(T.PLANTA_UCI);

    % Dividir datos en conjuntos de entrenamiento y validación
    numObservations = numel(VALOR);
    numTrain = floor(0.55 * numObservations);

    XTrain = VALOR(1:numTrain);
    YTrain = PLANTA_UCI(1:numTrain);

    XValidation = VALOR(numTrain+1:end);
    YValidation = PLANTA_UCI(numTrain+1:end);
    %%

%Se montan 10 redes en paralelo para estabilizar los resultados
for i=1:10,
        % Crear arquitectura de red LSTM
        inputSize = 1;
        numHiddenUnits = 75;
        numClasses = numel(categories(PLANTA_UCI));
        layers = [ ...
            sequenceInputLayer(inputSize)
            lstmLayer(numHiddenUnits,'OutputMode','last')
            fullyConnectedLayer(numClasses)
            softmaxLayer
            classificationLayer];

        % Especificar opciones de entrenamiento
        options = trainingOptions('adam', ...
            'MaxEpochs',5, ...
            'InitialLearnRate',0.001, ...
            'GradientThreshold',0.9, ...
            'Shuffle','every-epoch', ...
            'ValidationData',{XValidation,YValidation}, ...
            'Plots','none', ...
            'Verbose',false);

        % Entrenar la red LSTM
        net = trainNetwork(XTrain,YTrain,layers,options);

        % Realizar predicciones
        YPred = classify(net, XValidation);

        vector_predicion = string(cellstr(YPred'));
        vector_validacion = string(cellstr(YValidation'));

        vector_validacionTotal(i, strcmpi('PLANTA', vector_validacion)) = 1;
        vector_validacionTotal(i, ~strcmpi('PLANTA', vector_validacion)) = 0;
```

```
        vector_predicionTotal(i, strcmpi('PLANTA', vector_predicion)) = 1;
        vector_predicionTotal(i, ~strcmpi('PLANTA', vector_predicion)) = 0;

end
```

Figura 20. Código generación red neuronal LSTM

5.3. Implementación red neuronal BiLSTM

```
%Se carga la base de datos
load T.mat
tic
    % Puede ser que algunas tablas tengan valores NaN. Lo primero de todo
    % sería eliminar esas filas y se hace así:

    % Identificar filas con valores NaN en la columna "VALOR"
    nan_rows = isnan(cell2mat(T.VALOR));
    % Eliminar filas con valores NaN de la tabla T
    T = T(~nan_rows, :);

    % Extraer columnas "VALOR" y "PLANTA_UCI" de la tabla T
    VALOR = T.VALOR;
    PLANTA_UCI = categorical(T.PLANTA_UCI);

    % Dividir datos en conjuntos de entrenamiento y validación
    numObservations = numel(VALOR);
    numTrain = floor(0.6 * numObservations);

    XTrain = VALOR(1:numTrain);
    YTrain = PLANTA_UCI(1:numTrain);

    XValidation = VALOR(numTrain+1:end);
    YValidation = PLANTA_UCI(numTrain+1:end);

%Se montan 10 redes en paralelo para estabilizar los resultados
for i=1:10,

        % Crear arquitectura de red LSTM
        inputSize = 1;
        numHiddenUnits = 125;
        numClasses = numel(categories(PLANTA_UCI));
        layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(numHiddenUnits,'OutputMode','last')
            fullyConnectedLayer(numClasses)
            softmaxLayer
            classificationLayer];

        % Especificar opciones de entrenamiento
        options = trainingOptions('adam', ...
            'MaxEpochs',5, ...
            'InitialLearnRate',0.001, ...
            'GradientThreshold',1, ...
            'Shuffle','every-epoch', ...
            'ValidationData',{XValidation,YValidation}, ...
            'Plots','none', ...
            'Verbose',false);
```

```
        % Entrenar la red BiLSTM
        net = trainNetwork(XTrain,YTrain,layers,options);

        % Realizar predicciones
        YPred = classify(net, XValidation);

        vector_predicion = string(cellstr(YPred'));
        vector_validacion = string(cellstr(YValidation'));

        vector_validacionTotal(i, strcmpi('PLANTA', vector_validacion)) = 1;
        vector_validacionTotal(i, ~strcmpi('PLANTA', vector_validacion)) = 0;

        vector_predicionTotal(i, strcmpi('PLANTA', vector_predicion)) = 1;
        vector_predicionTotal(i, ~strcmpi('PLANTA', vector_predicion)) = 0;

end
```

Figura 21. Código generación red neuronal BiLSTM

5.4. Implementación red neuronal CNN con LSTM

```
% Cargar la base de datos
%load T.mat
tic

% Extraer columnas "VALOR" y "PLANTA_UCI" de la tabla T
VALOR = num2cell(T.VALOR);
PLANTA_UCI = categorical(T.PLANTA_UCI);

% Dividir datos en conjuntos de entrenamiento y validación
numObservations = numel(VALOR);
numTrain = floor(0.6 * numObservations);

XTrain = VALOR(1:numTrain);
YTrain = PLANTA_UCI(1:numTrain);

XValidation = VALOR(numTrain+1:end);
YValidation = PLANTA_UCI(numTrain+1:end);

% Montar 10 redes en paralelo para estabilizar los resultados
for i=1:10
    % Crear arquitectura de red CNN+LSTM
    inputSize = 1;
    numHiddenUnits = 125;
    numFilters = 16;
    filterSize = 3;
    numClasses = numel(categories(PLANTA_UCI));

    layers = [ ...
        sequenceInputLayer(inputSize)
        convolution2dLayer(filterSize, numFilters, 'Padding', 'same')
        reluLayer
        maxPooling2dLayer(2, 'Stride', 2)
        lstmLayer(numHiddenUnits, 'OutputMode', 'last')
        fullyConnectedLayer(numClasses)
        softmaxLayer
        classificationLayer];
```

```
    % Especificar opciones de entrenamiento
    options = trainingOptions('adam', ...
        'MaxEpochs', 5, ...
        'InitialLearnRate', 0.001, ...
        'GradientThreshold', 1, ...
        'Shuffle', 'every-epoch', ...
        'ValidationData', {XValidation, YValidation}, ...
        'Plots', 'none', ...
        'Verbose', false);

    % Entrenar la red CNN+LSTM
    net = trainNetwork(XTrain, YTrain, layers, options);

    % Realizar predicciones
    YPred = classify(net, XValidation);

    vector_predicion = string(cellstr(YPred'));
    vector_validacion = string(cellstr(YValidation'));

    vector_validacionTotal(i, strcmpi('PLANTA', vector_validacion)) = 1;
    vector_validacionTotal(i, ~strcmpi('PLANTA', vector_validacion)) = 0;

    vector_predicionTotal(i, strcmpi('PLANTA', vector_predicion)) = 1;
    vector_predicionTotal(i, ~strcmpi('PLANTA', vector_predicion)) = 0;
end
```

Figura 22. Código generación red neuronal CNN con LSTM

5.5. Implementación red neuronal Transformers

```
% Cargar la base de datos
%load T.mat
tic

% Extraer columnas "VALOR" y "PLANTA_UCI" de la tabla T
VALOR = num2cell(T.VALOR);
PLANTA_UCI = categorical(T.PLANTA_UCI);

% Dividir datos en conjuntos de entrenamiento y validación
numObservations = numel(VALOR);
numTrain = floor(0.6 * numObservations);

XTrain = VALOR(1:numTrain);
YTrain = PLANTA_UCI(1:numTrain);

XValidation = VALOR(numTrain+1:end);
YValidation = PLANTA_UCI(numTrain+1:end);

% Preprocesar datos para que sean adecuados para la red Transformer
% Convertir a formato adecuado
XTrain = cellfun(@(x) reshape(x, [1, 1, numel(x)]), XTrain, 'UniformOutput',
false);
XValidation = cellfun(@(x) reshape(x, [1, 1, numel(x)]), XValidation,
'UniformOutput', false);

% Montar 10 redes en paralelo para estabilizar los resultados
for i=1:10
```

```
    % Crear arquitectura de red basada en Transformers
    inputSize = 1;
    numHeads = 8;
    numHiddenUnits = 125;
    numClasses = numel(categories(PLANTA_UCI));
    numLayers = 6;
    dropoutRate = 0.1;

    layers = [ ...
        sequenceInputLayer(inputSize)
        transformerEncoderLayer(numHiddenUnits, numHeads, 'Dropout',
dropoutRate)
        fullyConnectedLayer(numClasses)
        softmaxLayer
        classificationLayer];

    % Especificar opciones de entrenamiento
    options = trainingOptions('adam', ...
        'MaxEpochs', 5, ...
        'InitialLearnRate', 0.001, ...
        'GradientThreshold', 1, ...
        'Shuffle', 'every-epoch', ...
        'ValidationData', {XValidation, YValidation}, ...
        'Plots', 'none', ...
        'Verbose', false);

    % Entrenar la red basada en Transformers
    net = trainNetwork(XTrain, YTrain, layers, options);

    % Realizar predicciones
    YPred = classify(net, XValidation);

    vector_predicion = string(cellstr(YPred'));
    vector_validacion = string(cellstr(YValidation'));

    vector_validacionTotal(i, strcmpi('PLANTA', vector_validacion)) = 1;
    vector_validacionTotal(i, ~strcmpi('PLANTA', vector_validacion)) = 0;

    vector_predicionTotal(i, strcmpi('PLANTA', vector_predicion)) = 1;
    vector_predicionTotal(i, ~strcmpi('PLANTA', vector_predicion)) = 0;
end
```

Figura 17. Código generación red neuronal Transformers

5.6. Variación entre modelos

Como se puede observar, tanto la red GRU, la LSTM y la BiLSTM presentan códigos casi idénticos, con la excepción de la línea de código donde se especifica la red a emplear. A continuación, se explican los principales hiperparámetros utilizados en estos códigos:

- ***Input size***: Define el tamaño de la entrada de la red neuronal, que en este caso es 1. Esto indica que cada dato de entrada será un valor escalar. En

problemas con datos multivariantes, el tamaño de entrada podría ser mayor para acomodar múltiples características de entrada simultáneamente.

- ***numHiddenUnits***: Se define el número de unidades ocultas en la capa recurrente (GRU, LSTM o BiLSTM). Las unidades ocultas, o neuronas, en la capa oculta son responsables de aprender representaciones complejas de los datos de entrada. Un mayor número de unidades ocultas puede aumentar la capacidad de aprendizaje de la red, pero también incrementa la complejidad computacional y el riesgo de sobreajuste.
- ***MaxEpochs***: Define el número máximo de épocas de entrenamiento, donde una época es un ciclo completo a través del conjunto de datos. Un mayor número de épocas permite que la red aprenda mejor los patrones en los datos, pero también aumenta el tiempo de entrenamiento y puede llevar al sobreajuste si se entrena durante demasiadas épocas.
- ***InitialLearnRate***: Establece la tasa de aprendizaje inicial para el optimizador. La tasa de aprendizaje determina el tamaño de los pasos que da el algoritmo de optimización en cada iteración para minimizar la función de pérdida. Una tasa de aprendizaje demasiado alta puede hacer que el entrenamiento sea inestable, mientras que una tasa demasiado baja puede hacer que el entrenamiento sea muy lento.
- ***GradientThreshold***: Define el umbral de gradiente para evitar gradientes explosivos. Durante el entrenamiento de redes neuronales recurrentes, los gradientes pueden crecer exponencialmente y causar inestabilidad en el proceso de aprendizaje. El umbral de gradiente establece un límite máximo para los gradientes, recortándolos si superan este límite para mantener la estabilidad del entrenamiento.

En cuanto a las otras dos arquitecturas restantes, podemos observar que hay alguna diferencia más. En el caso de CNN con LSTM, se añaden dos hiperparámetros adicionales:

- ***numFilters***: Número de filtros en la capa de convolución.

- ***filterSize***: Tamaño del filtro de la capa de convolución.

A la hora de definir las capas, en los primeros tres casos solo se define una capa (GRU, LSTM o BiLSTM). Sin embargo, en este caso se define primero la capa convolucional y, después, la capa LSTM (la cual se define de la misma manera que en los casos anteriores).

En la arquitectura basada en Transformers, se observa que primero se realiza una conversión de datos para que el formato sea adecuado para la arquitectura. A continuación, se continúa con la formación de las 10 redes como en los casos anteriores. Sin embargo, se introducen nuevos hiperparámetros:

- ***numHeads***: Número de cabezales en la capa de atención múltiple, que permite al modelo enfocarse en diferentes partes de la secuencia de entrada simultáneamente.
- ***dropoutRate***: Tasa de abandono, que es una técnica utilizada para prevenir el sobreajuste durante el entrenamiento al desactivar aleatoriamente una fracción de las unidades de la red.

La creación de las redes en este caso es más parecida a los tres primeros casos debido a que solo se define una capa, a diferencia de las dos capas en el caso de CNN con LSTM.

Capítulo 6: Experimentos y resultados

6.1. Representación de resultados

En este apartado se mostrarán los resultados del trabajo, en el que se ha trabajado con varias redes neuronales: GRU, LSTM, BiLSTM, CNN con LSTM y Transformers. A lo largo del proyecto, se han realizado numerosos experimentos para evaluar y comparar el rendimiento de estas arquitecturas. No obstante, solo se presentarán aquí los resultados más destacados y los que mejor se ajustan a la estructura del trabajo, aunque se hayan hecho más experimentos que no se incluirán en este documento. Primero, se mostrarán los resultados de cada arquitectura, escogiendo la mejor de cada una y, al final, se seleccionará un resultado final con una arquitectura específica y los parámetros más adecuados.

Al llevar a cabo las pruebas de las redes neuronales, existen hiperparámetros que pueden ser controlados y ajustados, como se explicó anteriormente. El objetivo es variar estos hiperparámetros en cada red neuronal para determinar la mejor arquitectura. Para lograrlo, he optado por un método basado en ensayo y error.

Aunque sería posible realizar un número muy elevado de pruebas para explorar todas las combinaciones de hiperparámetros, basándome en estudios anteriores y algunas pruebas realizadas por mí, se indica que, una vez que ciertos hiperparámetros exceden un rango específico, los resultados tienden a deteriorarse. Por lo tanto, he limitado mis experimentos a rangos de valores considerados óptimos. El rango de valores de los hiperparámetros utilizados en mis experimentos se presenta en la "TABLA X". Esta tabla resume los límites dentro de los cuales he variado los hiperparámetros para cada tipo de arquitectura, asegurando que las pruebas sean eficientes y que los resultados obtenidos sean los mejores posibles sin necesidad de realizar un número excesivo de experimentos.

Hiperparámetro	Rango
Test Porcentage	0.5 – 0.6
Initial Learn Rate	0.01 – 0.001
Max Epochs	5 - 7

Gradient Threshold	0.9 - 1
Number of Hidden Units	75 - 125

Tabla 2. Rango de valores establecidos

Como se muestra a continuación, se mostrarán los resultados en una matriz de confusión, anotando los parámetros de calidad en una tabla adjunta. Los parámetros generales de cada prueba se adjuntarán en una tabla. Por otro lado, se mostrará un recuadro debajo, indicando el parámetro que hemos variado para esa prueba en concreto. Al final de cada conjunto de pruebas, se mostrará una gráfica para poder visualizar la diferencia en los resultados de una forma más clara y sencilla.

Hiperparámetro	Valor
Train Ratio	X
Initial Learn Rate	X
Max Epochs	X
Gradient Threshold	X
Number of Hidden Units	X

Tabla 3. Ejemplo visualización de hiperparámetros de cada prueba

Matriz de confusión:

	PLANTA	UCI
PLANTA	**XX** **X%**	**XX** **X%**
UCI	**X** **X%**	**XX** **X%**

Figura 18: Matriz de confusión ejemplo prueba

Parámetros de calidad:

Precisión	X%
Especificidad	X%
Sensibilidad	X%
Tasa de acierto	X%

Tabla 4. Ejemplo parámetros de calidad

Parámetro por cambiar
X

6.2. Resultados obtenidos

6.2.1. GRU

Prueba 1:

En esta primera prueba se pretende ilustrar la importancia de mantener los valores dentro de los rangos establecidos. Se ha comenzado variando la proporción de

entrenamiento ("Train Ratio") fuera de los rangos establecidos para observar cómo afectan estos cambios a los resultados.

Prueba 1.1.:

Hiperparámetro	Valor
Train Ratio	0.55
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	100

Tabla 5. Prueba 1.1. Hiperparámetros (GRU)

Matriz de confusión:

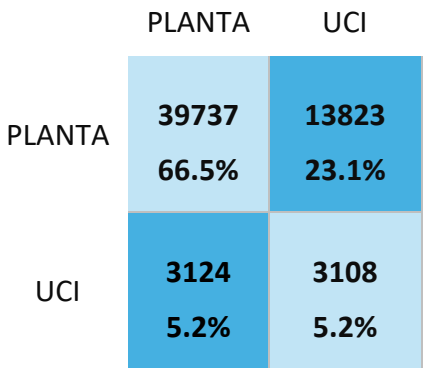

Figura 19: Matriz de confusión prueba 1.1 (GRU)

Parámetros de calidad:

Precisión	92.7%
Especificidad	49.9%
Sensibilidad	74.2%
Tasa de acierto	71.7%

Tabla 6. Parámetros de calidad Prueba 1.1 (GRU)

Train ratio
0.55

Prueba 1.2.:

Hiperparámetro	Valor
Train Ratio	0.65
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	100

Tabla 7. Prueba 1.2. Hiperparámetros (GRU)

Matriz de confusión:

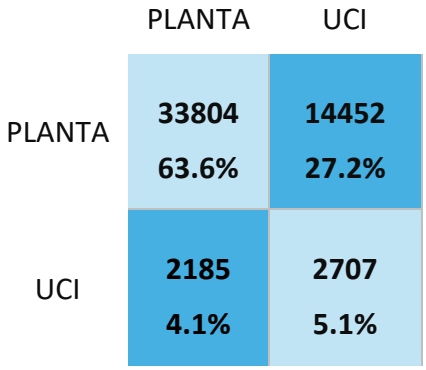

Figura 20: Matriz de confusión prueba 1.2 (GRU)

Parámetros de calidad:

Precisión	93.9%
Especificidad	55.3%
Sensibilidad	70.1%
Tasa de acierto	68.7%

Tabla 8. Parámetros de calidad Prueba 1.2 (GRU)

Train ratio
0.65

Prueba 1.3.:

Hiperparámetro	Valor
Train Ratio	0.75
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	100

Tabla 9. Prueba 1.3. Hiperparámetros (GRU)

Matriz de confusión:

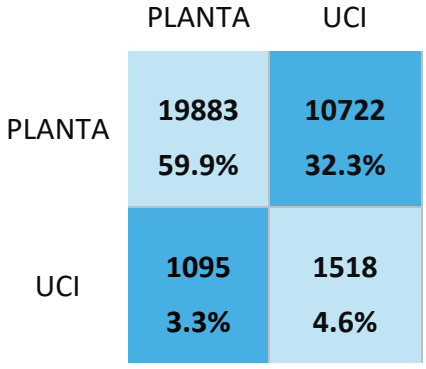

Figura 21: Matriz de confusión prueba 1.3 (GRU)

Parámetros de calidad:

Precisión	92.7%
Especificidad	56.4%
Sensibilidad	66.8%
Tasa de acierto	65.7%

Tabla 10. Parámetros de calidad Prueba 1.3 (GRU)

Train ratio
0.75

Prueba 1.4.:

Hiperparámetro	Valor
Train Ratio	0.8
Initial Learn Rate	0.01
Max Epochs	5

Gradient Threshold	1
Number of Hidden Units	100

Tabla 11. Prueba 1.4. Hiperparámetros (GRU)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**35682** **59.7%**	**17699** **29.6%**
UCI	**2798** **4.7%**	**3613** **6.0%**

Figura 22: Matriz de confusión prueba 1.4 (GRU)

Parámetros de calidad:

Precisión	94.8%
Especificidad	58.1%
Sensibilidad	65.0%
Tasa de acierto	64.4%

Tabla 12. Parámetros de calidad Prueba 1.4 (GRU)

Train ratio
0.8

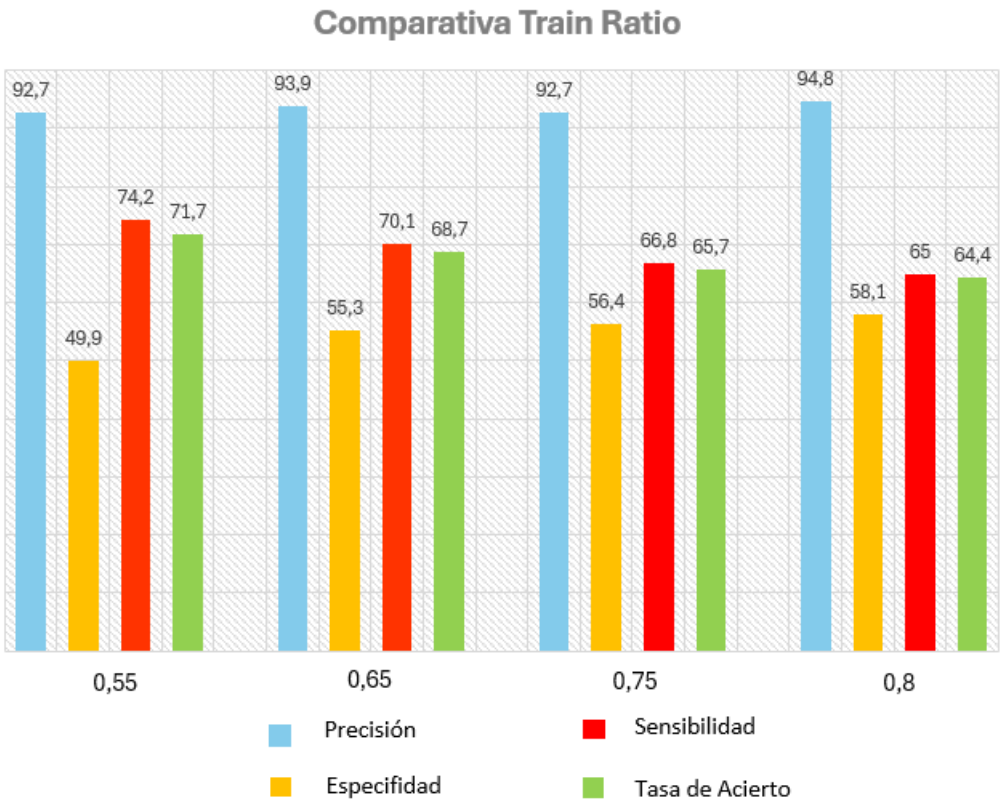

Figura 23: Comparativa Prueba 1 (GRU)

En esta primera prueba se pretende ilustrar la importancia de mantener los valores dentro de los rangos establecidos. Se ha comenzado variando la proporción de

entrenamiento ("Train Ratio") fuera de los rangos establecidos para observar cómo afectan estos cambios a los resultados.

Prueba 2:

En esta segunda prueba se ha variado la tasa de aprendizaje inicial (*Initial Learn Rate*), comenzando con valores dentro de los rangos establecidos.

Prueba 2.1.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 13. Prueba 2.1. Hiperparámetros (GRU)

Matriz de confusión:

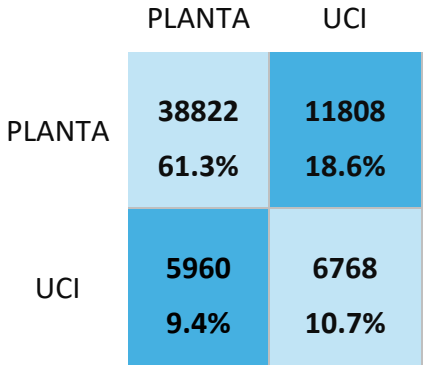

Figura 24: Matriz de confusión prueba 2.1 (GRU)

Parámetros de calidad:

Precisión	86.7%
Especificidad	53.2%
Sensibilidad	76.7%
Tasa de acierto	72.0%

Tabla 14. Parámetros de calidad Prueba 2.1 (GRU)

Initial Learn Rate
0.001

Prueba 2.2.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.005
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 15. Prueba 2.2. Hiperparámetros (GRU)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**37593** **59.3%**	**11902** **18.2%**
UCI	**6760** **10.7%**	**7103** **11.2%**

Figura 25: Matriz de confusión prueba 2.2 (GRU)

Parámetros de calidad:

Precisión	84.8%
Especificidad	51.2%
Sensibilidad	76.0%
Tasa de acierto	70.5%

Tabla 16. Parámetros de calidad Prueba 2.2 (GRU)

Initial Learn Rate
0.005

Prueba 2.3.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 17. Prueba 2.3. Hiperparámetros (GRU)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**40491** **63.9%**	**13909** **22.0%**
UCI	**4291** **6.8%**	**4667** **7.4%**

Figura 26: Matriz de confusión prueba 2.3 (GRU)

Parámetros de calidad:

Precisión	90.4%
Especificidad	52.1%
Sensibilidad	74.4%
Tasa de acierto	71.3%

Tabla 18. Parámetros de calidad Prueba 2.3. (GRU)

Initial Learn Rate
0.01

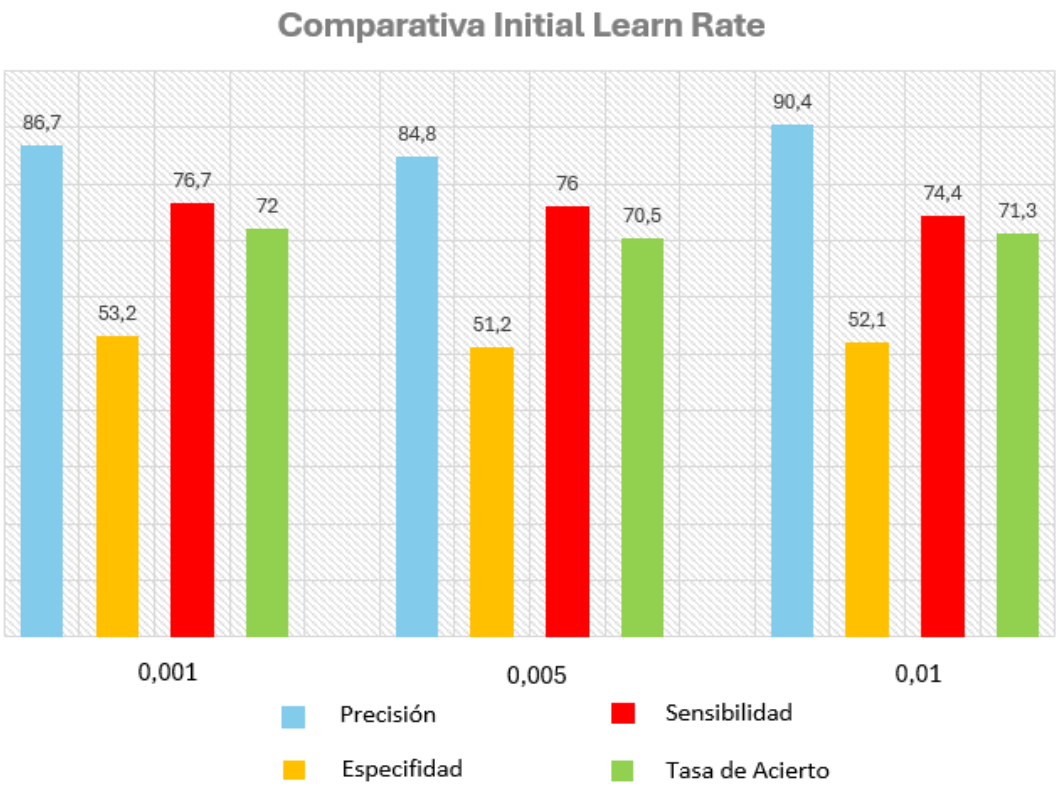

Figura 27: Comparativa Prueba 2 (GRU)

Se observa una clara tendencia a la reducción de la sensibilidad a medida que aumenta la tasa de aprendizaje inicial. Este modelo muestra una tasa de acierto del 72%, superando ampliamente los valores de la prueba 1, lo que sugiere que la configuración óptima se encuentra en torno a estos valores. Además, en la prueba 3, la precisión supera el 90%, pero la sensibilidad se ve muy reducida, lo que indica que buscamos un modelo lo más equilibrado posible.

Prueba 3:

Para esta prueba se han variado dos hiperparámetros: el gradiente y las épocas, cambiando ahora a un ratio de entrenamiento del 60% para observar la variación tanto de estos valores como de los valores de entrenamiento con respecto a las pruebas anteriores.

Prueba 3.1.:

Hiperparámetro	Valor
Train Ratio	0.6

Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	125

Tabla 19. Prueba 3.1. Hiperparámetros (GRU)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**29945** **59.1%**	**8671** **17.1%**
UCI	**6009** **11.9%**	**6062** **12.0%**

Figura 28: Matriz de confusión prueba 3.1 (GRU)

Parámetros de calidad:

Precisión	83.3%
Especificidad	50.2%
Sensibilidad	77.5%
Tasa de acierto	71.0%

Tabla 20. Parámetros de calidad Prueba 3.1 (GRU)

Gradient Threshold & Max Epochs
0.9 & 5

Prueba 3.2.:

Hiperparámetro	**Valor**
Train Ratio	0.6
Initial Learn Rate	0.001
Max Epochs	7
Gradient Threshold	0.9
Number of Hidden Units	125

Tabla 21. Prueba 3.2. Hiperparámetros (GRU)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**30610** **60.4%**	**9087** **17.9%**
UCI	**5344** **10.5%**	**5646** **11.1%**

Figura 29: Matriz de confusión prueba 3.2 (GRU)

Parámetros de calidad:

Precisión	85.1%
Especificidad	51.4%
Sensibilidad	77.1%
Tasa de acierto	71.5%

Tabla 22. Parámetros de calidad Prueba 3.2. (GRU)

Gradient Threshold & Max Epochs
0.9 & 7

Prueba 3.3.:

Hiperparámetro	Valor
Train Ratio	0.6
Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	125

Tabla 23. Prueba 3.3. Hiperparámetros (GRU)

Matriz de confusión:

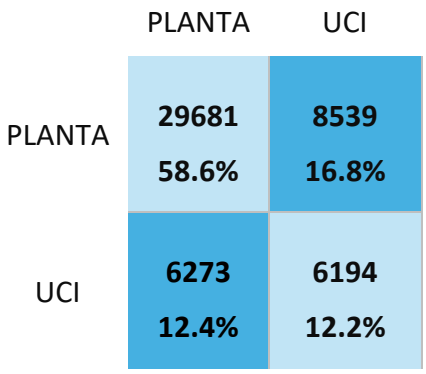

Figura 30: Matriz de confusión prueba 3.3 (GRU)

Parámetros de calidad:

Precisión	82.6%
Especificidad	49.7%
Sensibilidad	77.7%
Tasa de acierto	70.8%

Tabla 24. Parámetros de calidad Prueba 3.3. (GRU)

Gradient Threshold & Max Epochs
1 & 5

Prueba 3.4.:

Hiperparámetro	Valor
Train Ratio	0.6
Initial Learn Rate	0.001
Max Epochs	7
Gradient Threshold	1
Number of Hidden Units	125

Tabla 25. Prueba 3.4. Hiperparámetros (GRU)

Matriz de confusión:

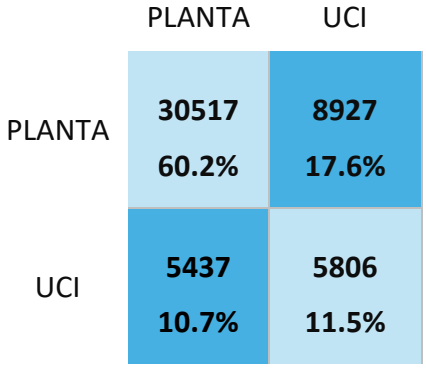

Figura 31: Matriz de confusión prueba 3.4 (GRU)

Parámetros de calidad:

Precisión	84.9%
Especificidad	51.6%
Sensibilidad	77.4%
Tasa de acierto	71.7%

Tabla 26. Parámetros de calidad Prueba 3.4. (GRU)

Gradient Threshold & Max Epochs
1 & 7

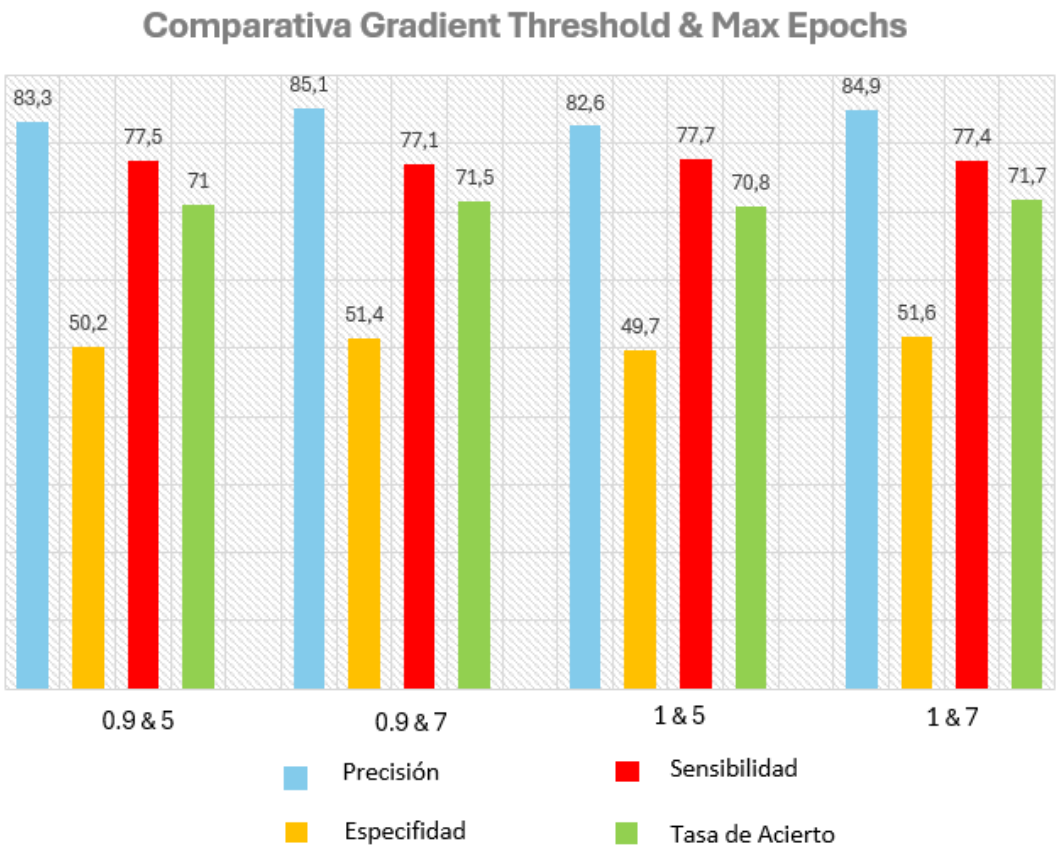

Figura 32: Comparativa Prueba 3 (GRU)

La variación del gradiente y las épocas no ha proporcionado resultados significativos. Se observa que la especificidad aumenta con el número de épocas. Por otro lado, la tasa de acierto mejora en el último caso; sin embargo, no es un resultado destacable, probablemente debido al "Train Ratio" del 60% común en estos ejemplos, lo que en general disminuye la calidad de los resultados.

Prueba 4:

En esta última prueba se ha variado el número de unidades ocultas *(Hidden Units*), manteniendo la proporción de entrenamiento en un 50%, ya que es donde se han obtenido los mejores resultados en general.

Prueba 4.1.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001

Max Epochs	7
Gradient Threshold	1
Number of Hidden Units	75

Tabla 27. Prueba 4.1. Hiperparámetros (GRU)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**38552** **60.8%**	**11420** **18.0%**
UCI	**6230** **9.8%**	**7156** **11.3%**

Figura 33: Matriz de confusión prueba 4.1 (GRU)

Parámetros de calidad:

Precisión	86.1%
Especificidad	53.5%
Sensibilidad	77.1%
Tasa de acierto	72.1%

Tabla 28. Parámetros de calidad Prueba 4.1 (GRU)

Number of Hidden Units
75

Prueba 4.2.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	7
Gradient Threshold	1
Number of Hidden Units	100

Tabla 29. Prueba 4.2. Hiperparámetros (GRU)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**38219** **60.3%**	**11231** **17.7%**
UCI	**6563** **10.4%**	**7345** **11.6%**

Figura 34: Matriz de confusión prueba 4.2 (GRU)

Parámetros de calidad:

Precisión	85.3%
Especificidad	52.8%
Sensibilidad	77.3%
Tasa de acierto	71.9%

Tabla 30. Parámetros de calidad Prueba 4.2. (GRU)

Number of Hidden Units
100

Prueba 4.3.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	7
Gradient Threshold	1
Number of Hidden Units	125

Tabla 31. Prueba 4.3. Hiperparámetros (GRU)

Matriz de confusión:

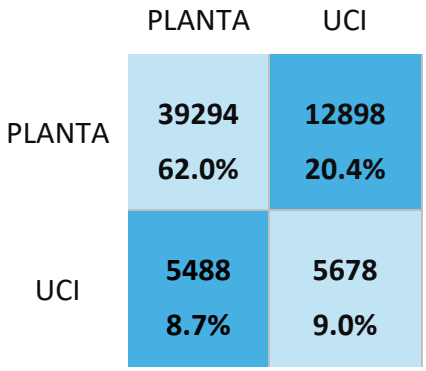

Figura 35: Matriz de confusión prueba 4.3 (GRU)

Parámetros de calidad:

Precisión	87.7%
Especificidad	50.9%
Sensibilidad	75.3%
Tasa de acierto	71.0%

Tabla 32. Parámetros de calidad Prueba 4.3. (GRU)

Number of Hidden Units
125

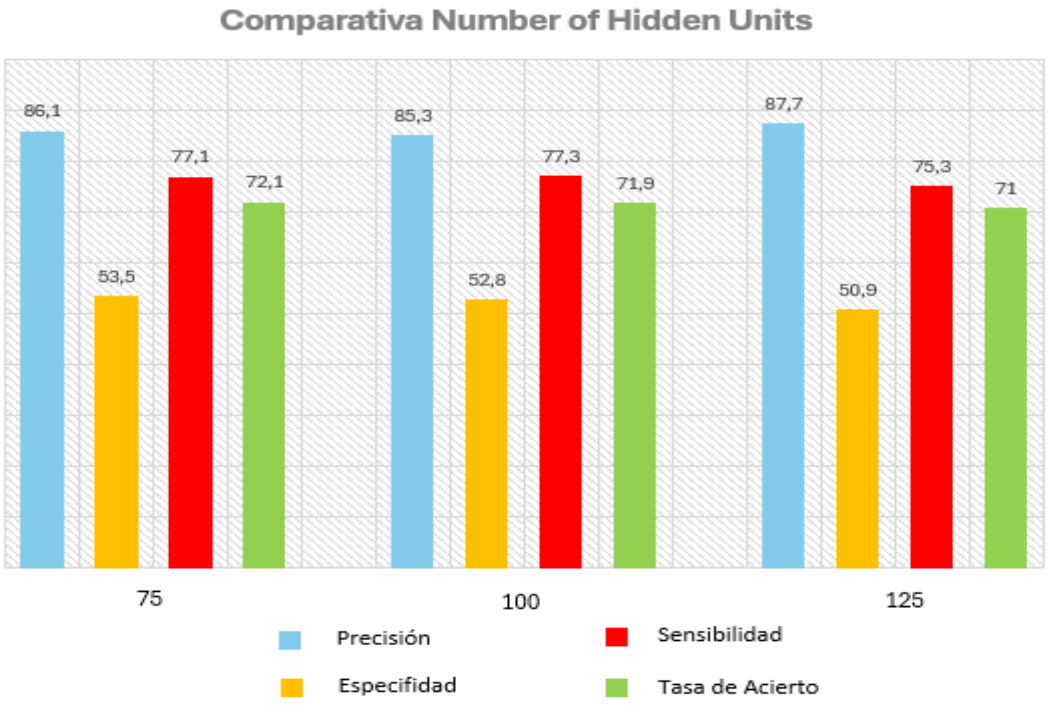

Figura 36: Comparativa Prueba 4 (GRU)

En esta prueba final se observa el mejor valor de la tasa de acierto, alcanzando el 72.1%. Cuantos menores han sido los valores de los hiperparámetros (dentro de los rangos establecidos), mejores han sido los resultados en general. Este modelo es el más estable de todos los probados y, por lo tanto, se seleccionará como el mejor modelo. Aunque la precisión en este caso se ve comprometida, la sensibilidad y otros valores mejoran, resultando en un modelo más equilibrado.

6.2.2. LSTM

Prueba 1:

Para esta primera prueba volveremos a probar con valores un tanto aleatorios dentro del rango establecido, variando solo el número de unidades ocultas. Mantendremos la tendencia de tener "Train Ratios" más bajos, buscando resultados más estables.

Prueba 1.1.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	75

Tabla 33. Prueba 1.1. Hiperparámetros (LSTM)

Matriz de confusión:

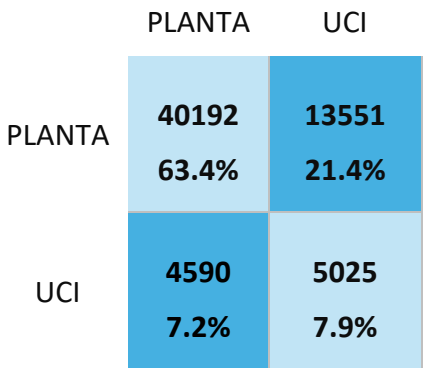

Figura 37: Matriz de confusión prueba 1.1. (LSTM)

Parámetros de calidad:

Precisión	89.8%
Especificidad	52.3%
Sensibilidad	74.8%
Tasa de acierto	71.4%

Tabla 34. Parámetros de calidad Prueba 1.1. (LSTM)

Number of Hidden Units
75

Prueba 1.2.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	100

Tabla 35. Prueba 1.2. Hiperparámetros (LSTM)

Matriz de confusión:

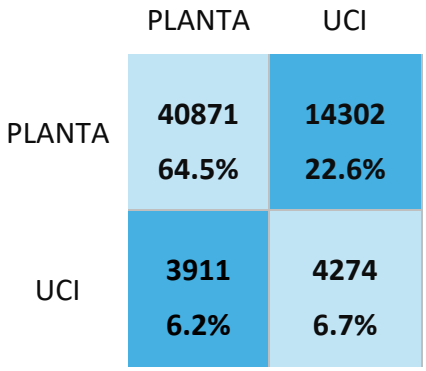

Figura 38: Matriz de confusión prueba 1.2. (LSTM)

Parámetros de calidad:

Precisión	91.3%
Especificidad	52.2%
Sensibilidad	74.1%
Tasa de acierto	71.3%

Tabla 36. Parámetros de calidad Prueba 1.2. (LSTM)

Number of Hidden Units
100

Prueba 1.3.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	125

Tabla 37. Prueba 1.3. Hiperparámetros (LSTM)

Matriz de confusión:

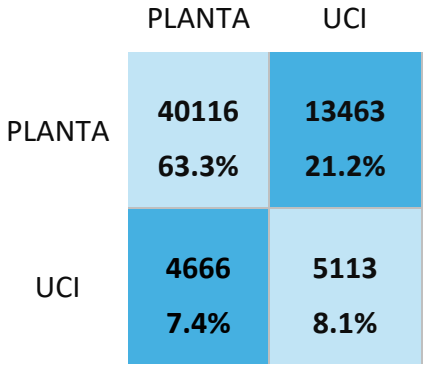

Figura 39: Matriz de confusión prueba 1.3. (LSTM)

Parámetros de calidad:

Precisión	89.6%
Especificidad	52.3%
Sensibilidad	74.9%
Tasa de acierto	71.4%

Tabla 38. Parámetros de calidad Prueba 1.3. (LSTM)

Number of Hidden Units
125

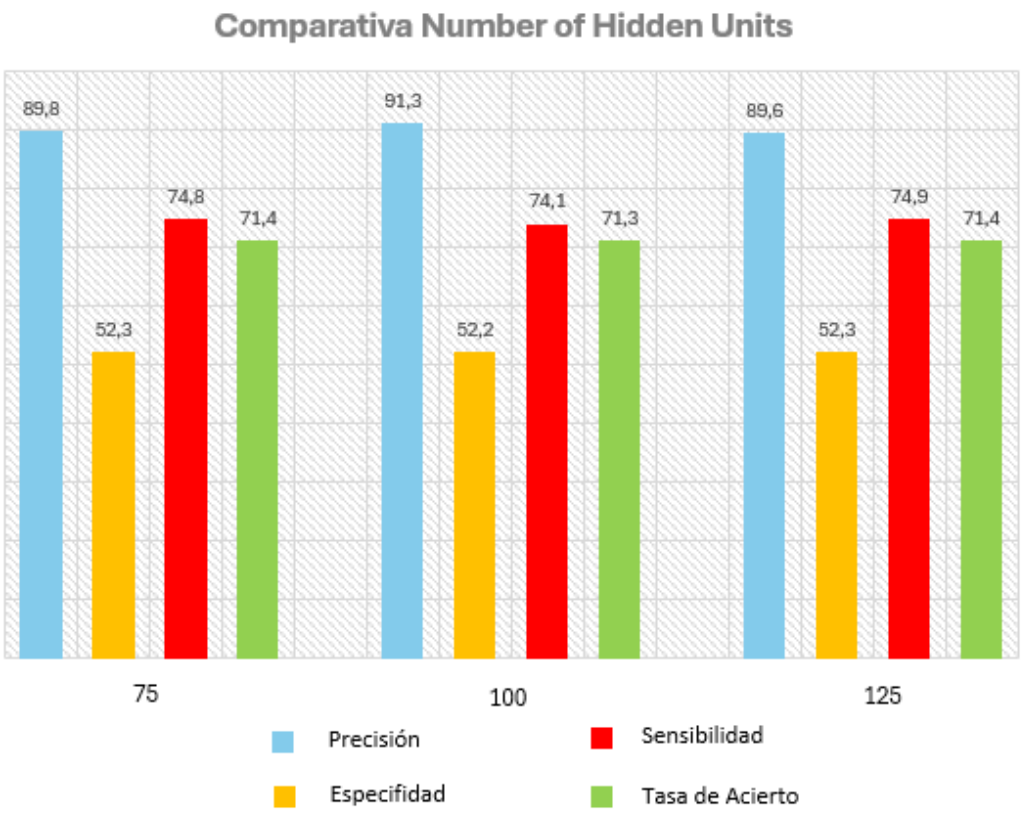

Figura 40: Comparativa Prueba 1 (LSTM)

Observamos que los valores no cambian mucho, observando el único cambio en la precisión, siendo la segunda prueba la mejor parada. Sin embargo, ha obtenido la peor tasa de acierto, que es el valor que mejor indica el comportamiento general de una arquitectura.

Prueba 2:

Para esta prueba hemos optado por cambiar la ratio de entrenamiento a un 55%, en busca de unos resultados mejores. He dejado la ratio de entrenamiento inicial en 0.01, manteniendo el número de unidades ocultas en un término medio con 100. He ido cambiando los valores de *Gradient Threshold* y *Max Epochs* para observar la variación.

Prueba 2.1.:

Hiperparámetro	Valor
Train Ratio	0.55
Initial Learn Rate	0.01

Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	100

Tabla 39. Prueba 2.1. Hiperparámetros (LSTM)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**35870** **62.9%**	**12099** **21.2%**
UCI	**4443** **7.8%**	**4611** **8.1%**

Figura 41: Matriz de confusión prueba 2.1. (LSTM)

Parámetros de calidad:

Precisión	88.0%
Especificidad	50.9%
Sensibilidad	74.8%
Tasa de acierto	71.0%

Tabla 40. Parámetros de calidad Prueba 2.1. (LSTM)

Gradient Threshold & Max Epochs
0.9 & 5

Prueba 2.2.:

Hiperparámetro	Valor
Train Ratio	0.55
Initial Learn Rate	0.01
Max Epochs	7
Gradient Threshold	0.9
Number of Hidden Units	100

Tabla 41. Prueba 2.2. Hiperparámetros (LSTM)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**35930** **63.0%**	**12161** **21.3%**
UCI	**4383** **7.7%**	**4549** **8.0%**

Figura 42: Matriz de confusión prueba 2.2. (LSTM)

Parámetros de calidad:

Precisión	89.1%
Especificidad	50.9%
Sensibilidad	74.7%
Tasa de acierto	71.0%

Tabla 42. Parámetros de calidad Prueba 2.2. (LSTM)

Gradient Threshold & Max Epochs
0.9 & 7

Prueba 2.3.:

Hiperparámetro	Valor
Train Ratio	0.55
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	100

Tabla 43. Prueba 2.3. Hiperparámetros (LSTM)

Matriz de confusión:

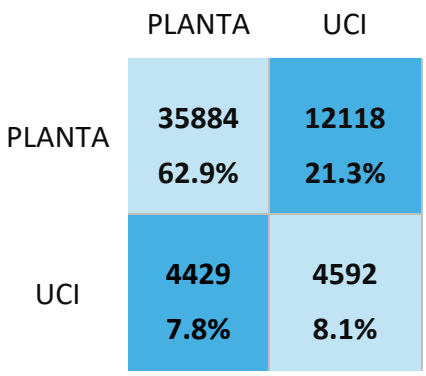

Figura 43: Matriz de confusión prueba 2.3. (LSTM)

Parámetros de calidad:

Precisión	89.0%
Especificidad	50.9%
Sensibilidad	74.8%
Tasa de acierto	71.0%

Tabla 44. Parámetros de calidad Prueba 2.3. (LSTM)

Gradient Threshold & Max Epochs
0.9 & 5

Prueba 2.4.:

Hiperparámetro	Valor
Train Ratio	0.55
Initial Learn Rate	0.01
Max Epochs	7
Gradient Threshold	1
Number of Hidden Units	100

Tabla 45. Prueba 2.4. Hiperparámetros (LSTM)

Matriz de confusión:

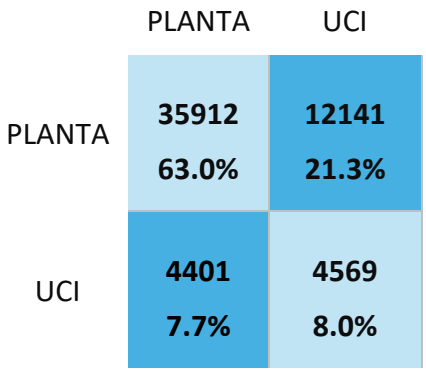

Figura 44: Matriz de confusión prueba 2.4. (LSTM)

Parámetros de calidad:

Precisión	89.1%
Especificidad	50.9%
Sensibilidad	74.7%
Tasa de acierto	71.0%

Tabla 46. Parámetros de calidad Prueba 2.4. (LSTM)

Gradient Threshold & Max Epochs
1 & 7

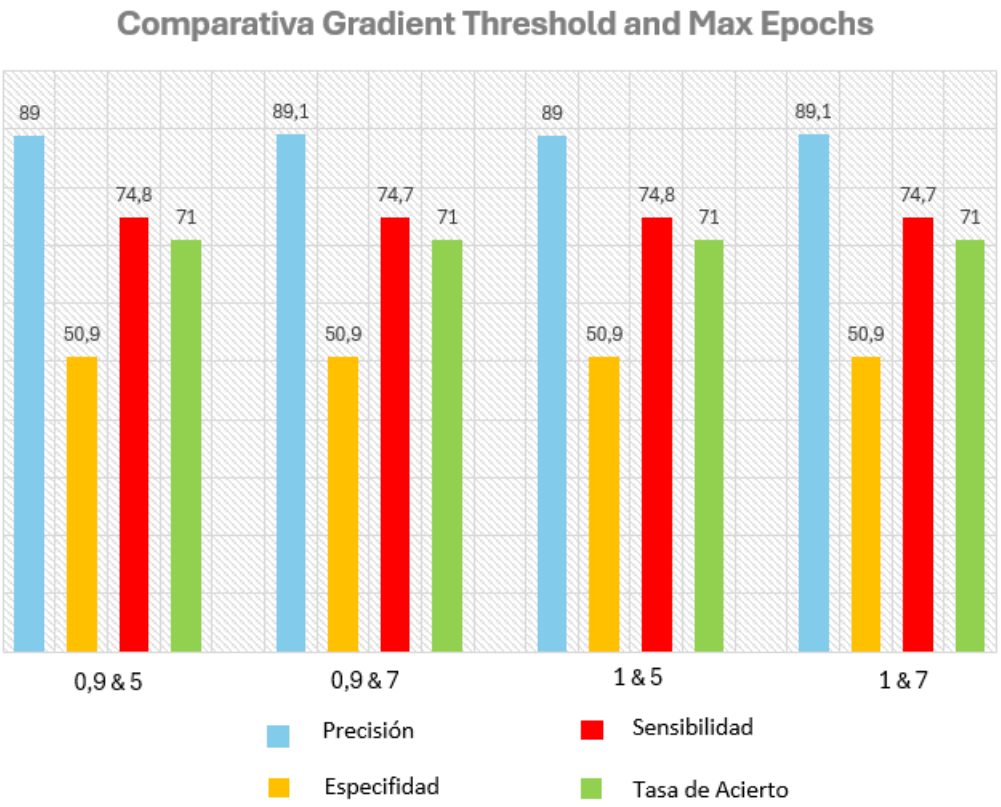

Figura 45: Comparativa Prueba 2 (LSTM)

Esta ronda de pruebas no nos ha mostrado grandes resultados, en general no se ve reflejado un cambio notorio entre las pruebas y, para empeorar, hemos obtenido peores resultados que los anteriores. Son en general resultados algo pobres, que nos hace indicar que los mejores resultados seguramente no estén en torno al 55% de entrenamiento, sino más hacia un 50%.

Prueba 3:

Para esta prueba se ha variado el *Initial Learn Rate*, comenzando con el menor valor dentro del rango y terminando con el valor mayor.

Prueba 3.1.:

Hiperparámetro	Valor
Train Ratio	0.55
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	100

Tabla 47. Prueba 3.1. Hiperparámetros (LSTM)

Matriz de confusión:

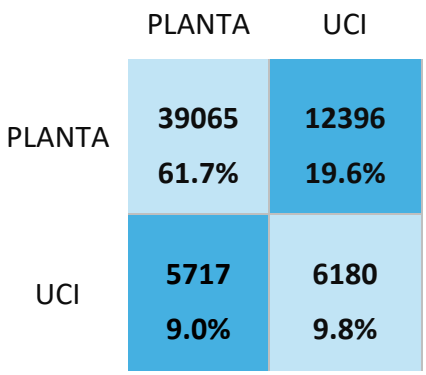

Figura 46: Matriz de confusión prueba 3.1. (LSTM)

Parámetros de calidad:

Precisión	87.2%
Especificidad	51.9%
Sensibilidad	75.9%
Tasa de acierto	71.4%

Tabla 48. Parámetros de calidad Prueba 3.1. (LSTM)

Initial Learn Rate
0.001

Prueba 3.2.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.005
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 49. Prueba 3.2. Hiperparámetros (LSTM)

Matriz de confusión:

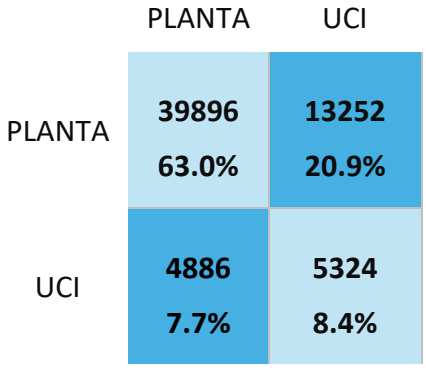

Figura 47: Matriz de confusión prueba 3.2. (LSTM)

Parámetros de calidad:

Precisión	89.1%
Especificidad	52.1%
Sensibilidad	75.1%
Tasa de acierto	71.4%

Tabla 50. Parámetros de calidad Prueba 3.2. (LSTM)

Initial Learn Rate
0.005

Prueba 3.3.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.005
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 51. Prueba 3.3. Hiperparámetros (LSTM)

Matriz de confusión:

	PLANTA	UCI
PLANTA	40821 64.4%	14264 22.5%
UCI	3961 6.3%	4312 6.8%

Figura 48: Matriz de confusión prueba 3.3. (LSTM)

Parámetros de calidad:

Precisión	91.2%
Especificidad	52.1%
Sensibilidad	74.1%
Tasa de acierto	71.2%

Tabla 52. Parámetros de calidad Prueba 3.3. (LSTM)

Initial Learn Rate
0.01

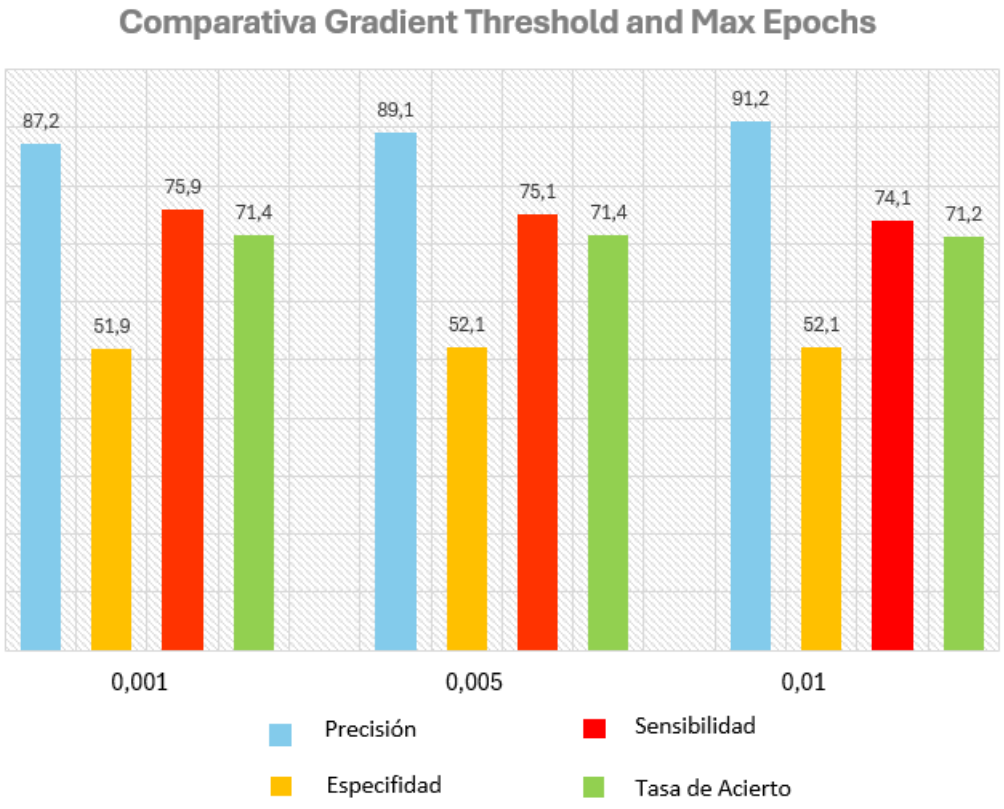

Figura 49: Comparativa Prueba 3 (LSTM)

En esta última prueba se ha intentado depurar y sacar un resultado algo mejor, sin embargo, los resultados generales de esta arquitectura no son muy buenos. La mayor tasa de acierto obtenida es de un 71,4%, muy por debajo de lo obtenido en la arquitectura GRU. Es difícil escoger un resultado final por encima del resto debido a lo parejos que son todos los resultados, sin embargo, voy a decantarme por la prueba 3.2. debido a que es el resultado más estable en cuanto a resultados se refiere. Es

cierto que la precisión se ve afectada negativamente por la elección, pero la sensibilidad se ve afectada positivamente.

6.2.3. BiLSTM

Prueba 1:

Para esta arquitectura, se ha decidido comenzar variando el porcentaje de entrenamiento, manteniéndolo dentro de los rangos establecidos y utilizando valores menores de los hiperparámetros dentro de su rango, siguiendo así la línea de los mejores resultados obtenidos en arquitecturas anteriores.

Prueba 1.1.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 53. Prueba 1.1. Hiperparámetros (BiLSTM)

Matriz de confusión:

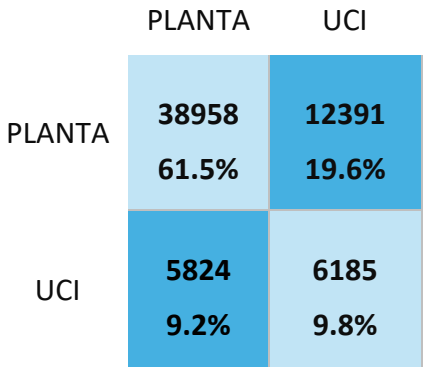

Figura 50: Matriz de confusión prueba 1.1. (BiLSTM)

Parámetros de calidad:

Precisión	87.0%
Especificidad	51.5%
Sensibilidad	75.9%
Tasa de acierto	71.3%

Tabla 54. Parámetros de calidad Prueba 1.1. (BiLSTM)

Train Ratio
0.5

Prueba 1.2.:

Hiperparámetro	Valor
Train Ratio	0.55
Initial Learn Rate	0.001
Max Epochs	5

Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 55. Prueba 1.2. Hiperparámetros (BiLSTM)

Matriz de confusión:

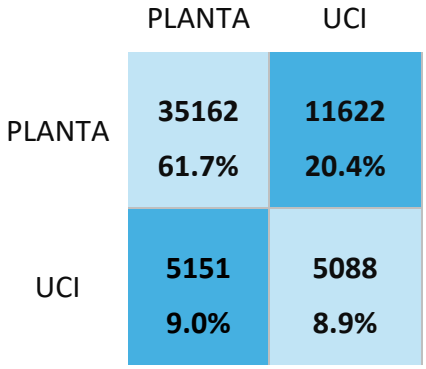

Figura 51: Matriz de confusión prueba 1.2. (BiLSTM)

Parámetros de calidad:

Precisión	87.2%
Especificidad	49.7%
Sensibilidad	75.2%
Tasa de acierto	70.6%

Tabla 56. Parámetros de calidad Prueba 1.2. (BiLSTM)

Train Ratio
0.55

Prueba 1.3.:

Hiperparámetro	Valor
Train Ratio	0.6
Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 57. Prueba 1.3. Hiperparámetros (BiLSTM)

Matriz de confusión:

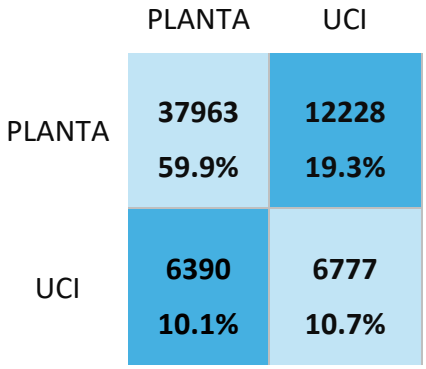

Figura 52: Matriz de confusión prueba 1.3. (BiLSTM)

Parámetros de calidad:

Precisión	85.6%
Especificidad	51.5%
Sensibilidad	75.6%
Tasa de acierto	70.6%

Tabla 58. Parámetros de calidad Prueba 1.3. (BiLSTM)

Train Ratio
0.6

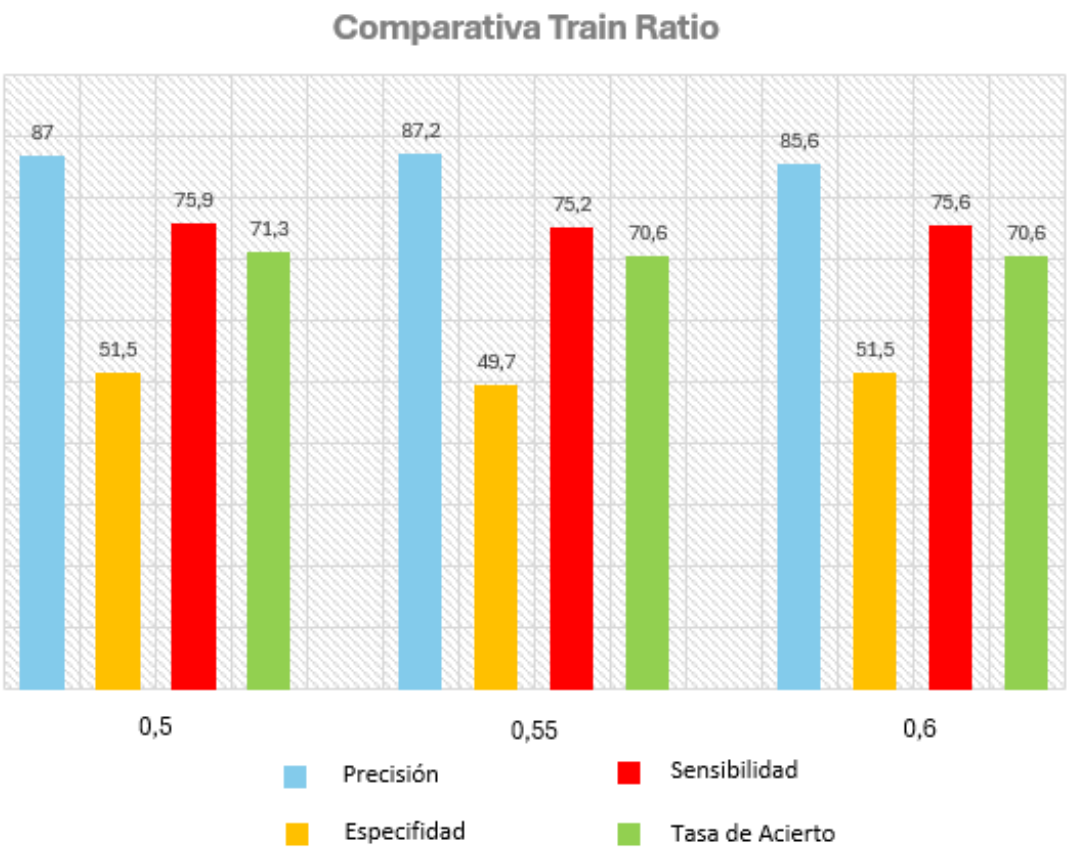

Figura 53: Comparativa Prueba 1 (BiLSTM)

Como se puede observar, los resultados son mejores con proporciones de entrenamiento ("Train Ratios") inferiores. Aunque es cierto que los valores de precisión suelen ser mayores en otros casos, se busca la mayor estabilidad del modelo, siendo además la sensibilidad un valor muy importante.

Prueba 2:

En esta segunda prueba, se ha optado por un porcentaje de entrenamiento mayor, con valores menores para las épocas y el gradiente, y variando el número de unidades ocultas (*Hidden Units*).

Prueba 2.1.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 59. Prueba 2.1. Hiperparámetros (BiLSTM)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**40106** **63.3%**	**13457** **21.2%**
UCI	**4676** **7.4%**	**5119** **8.1%**

Figura 54: Matriz de confusión prueba 1.2. (BiLSTM)

Parámetros de calidad:

Precisión	89.6%
Especificidad	52.3%
Sensibilidad	74.9%
Tasa de acierto	71.4%

Tabla 60. Parámetros de calidad Prueba 2.1. (BiLSTM)

Number of Hidden Units
75

Prueba 2.2.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	100

Tabla 61. Prueba 2.2. Hiperparámetros (BiLSTM)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**41489** **65.5%**	**14837** **23.4%**
UCI	**3293** **5.2%**	**3739** **5.9%**

Figura 55: Matriz de confusión prueba 2.2. (BiLSTM)

Parámetros de calidad:

Precisión	92.6%
Especificidad	53.2%
Sensibilidad	73.7%
Tasa de acierto	71.4%

Tabla 62. Parámetros de calidad Prueba 2.2. (BiLSTM)

Number of Hidden Units
100

Prueba 2.3.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	125

Tabla 63. Prueba 2.3. Hiperparámetros (BiLSTM)

Matriz de confusión:

	PLANTA	UCI
PLANTA	40486 63.9%	13937 22.0%
UCI	4296 6.8%	4639 7.3%

Figura 56: Matriz de confusión prueba 2.3. (BiLSTM)

Parámetros de calidad:

Precisión	90.4%
Especificidad	51.9%
Sensibilidad	74.4%
Tasa de acierto	71.2%

Tabla 64. Parámetros de calidad Prueba 1.1. (BiLSTM)

Number of Hidden Units
125

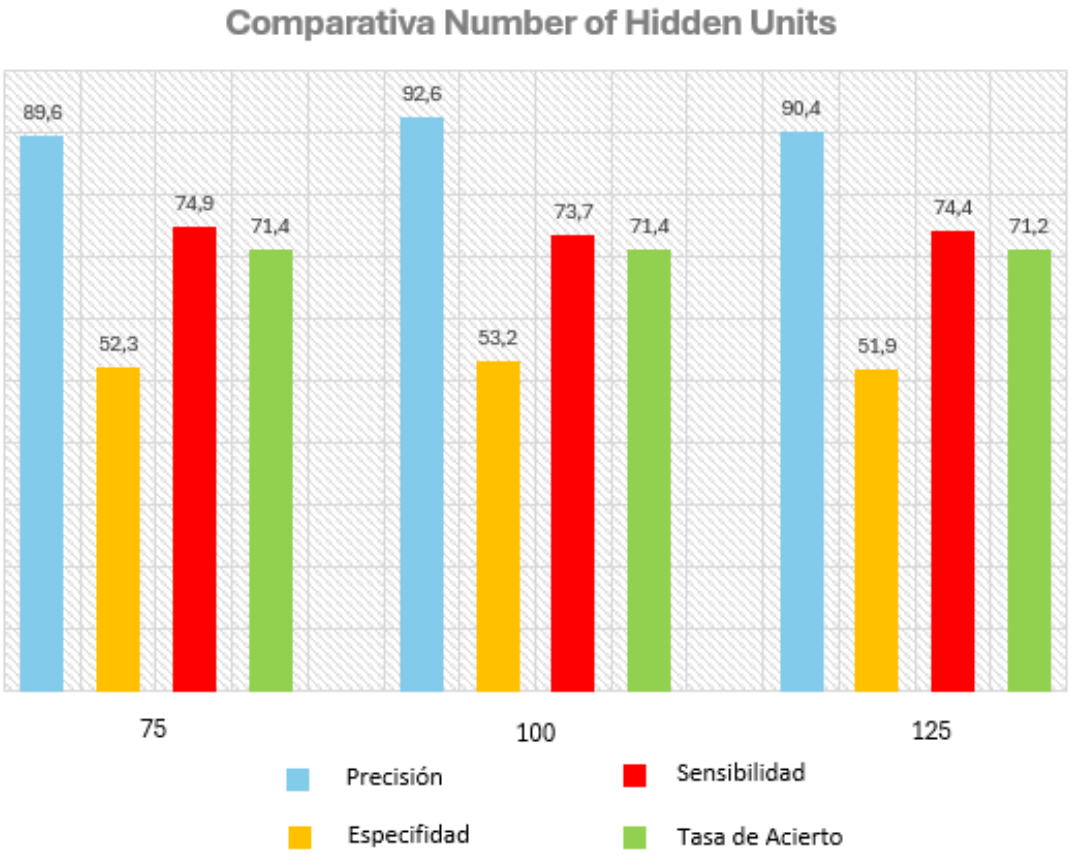

Figura 57: Comparativa Prueba 2 (BiLSTM)

Se puede observar que los resultados son mejores en los dos primeros casos. El primer caso presenta el resultado más estable, mientras que el segundo caso tiene un valor de precisión muy bueno. Sin embargo, el valor de la sensibilidad se ve afectado significativamente.

Prueba 3:

En esta última prueba de la arquitectura BiLSTM, se han elegido valores más bajos de porcentaje de entrenamiento y un valor medio de 100 para el número de unidades ocultas, buscando obtener una mayor precisión, como en el caso anterior.

Prueba 3.1.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	100

Tabla 65. Prueba 3.1. Hiperparámetros (BiLSTM)

Matriz de confusión:

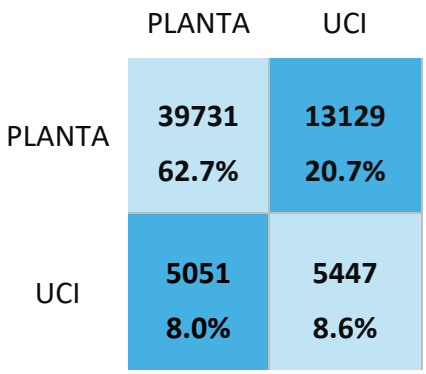

Figura 58: Matriz de confusión prueba 3.1. (BiLSTM)

Parámetros de calidad:

Precisión	88.7%
Especificidad	51.9%
Sensibilidad	75.2%
Tasa de acierto	71.3%

Tabla 66. Parámetros de calidad Prueba 3.1. (BiLSTM)

Gradient Threshold & Max Epochs
0.9 & 5

Prueba 3.2.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	7
Gradient Threshold	0.9
Number of Hidden Units	100

Tabla 67. Prueba 3.2. Hiperparámetros (BiLSTM)

Matriz de confusión:

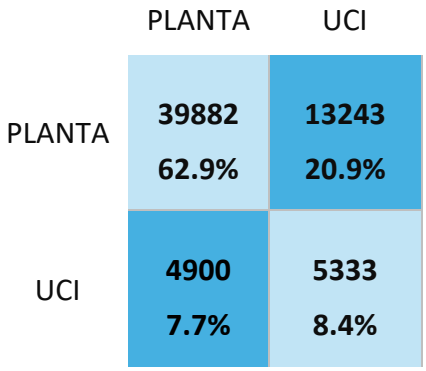

Figura 59: Matriz de confusión prueba 3.2. (BiLSTM)

Parámetros de calidad:

Precisión	89.1%
Especificidad	52.1%
Sensibilidad	75.1%
Tasa de acierto	71.4%

Tabla 68. Parámetros de calidad Prueba 3.2. (BiLSTM)

Gradient Threshold & Max Epochs
0.9 & 7

Prueba 3.3.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	100

Tabla 69. Prueba 3.3. Hiperparámetros (BiLSTM)

Matriz de confusión:

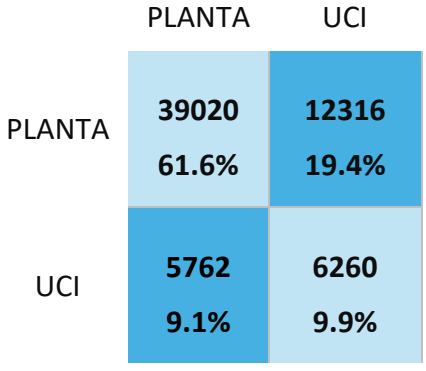

Figura 60: Matriz de confusión prueba 3.3. (BiLSTM)

Parámetros de calidad:

Precisión	87.1%
Especificidad	52.1%
Sensibilidad	76.0%
Tasa de acierto	71.5%

Tabla 70. Parámetros de calidad Prueba 3.3. (BiLSTM)

Gradient Threshold & Max Epochs
1 & 5

Prueba 3.4.:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	7

Gradient Threshold	1
Number of Hidden Units	100

Tabla 71. Prueba 3.4. Hiperparámetros (BiLSTM)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**39753** **62.7%**	**13159** **20.8%**
UCI	**5029** **7.9%**	**5417** **8.5%**

Figura 61: Matriz de confusión prueba 3.3. (BiLSTM)

Parámetros de calidad:

Precisión	88.8%
Especificidad	51.9%
Sensibilidad	75.1%
Tasa de acierto	71.3%

Tabla 72. Parámetros de calidad Prueba 3.4. (BiLSTM)

Gradient Threshold & Max Epochs
1 & 7

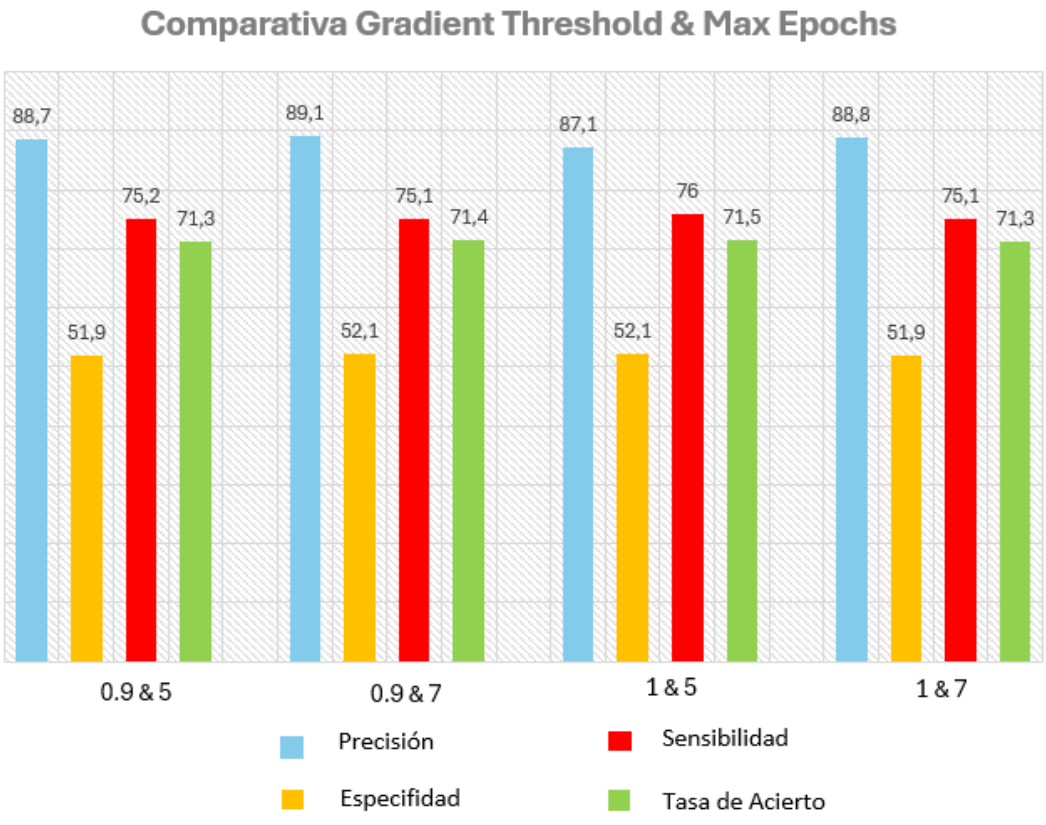

Figura 62: Comparativa Prueba 3 (BiLSTM)

Se observa que se ha obtenido el mayor valor de tasa de acierto para esta arquitectura, con un valor del 71.5%. Por otro lado, es el que tiene menor precisión, pero el valor de la sensibilidad es mucho mejor que en los demás casos. Aunque se

desearía tener una mayor precisión, este es el mejor modelo dentro de los resultados obtenidos para esta arquitectura.

6.2.4. CNN con LSTM

Debido a la exigencia computacional y a la imposibilidad de disponer diariamente de un ordenador con las capacidades necesarias, no se han podido realizar un gran número de pruebas para esta arquitectura. La elección de modelos se ha hecho en base a los resultados obtenidos en las arquitecturas anteriores. Dicho esto, se ha intentado variar un poco los valores para encontrar la versión óptima para esta arquitectura y poder también estudiar la variación en cada resultado.

Prueba 1:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 73. Prueba 1. Hiperparámetros (CNN con LSTM)

Matriz de confusión:

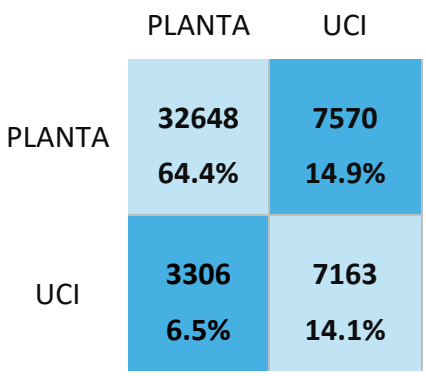

Figura 63: Matriz de confusión prueba 1 (CNN con LSTM)

Parámetros de calidad:

Precisión	90.8%
Especificidad	68.4%
Sensibilidad	81.2%
Tasa de acierto	78.5%

Tabla 74. Parámetros de calidad Prueba 1 (CNN con LSTM)

Prueba 2:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	7

Gradient Threshold	1
Number of Hidden Units	75

Tabla 75. Prueba 2. Hiperparámetros (CNN con LSTM)

Matriz de confusión:

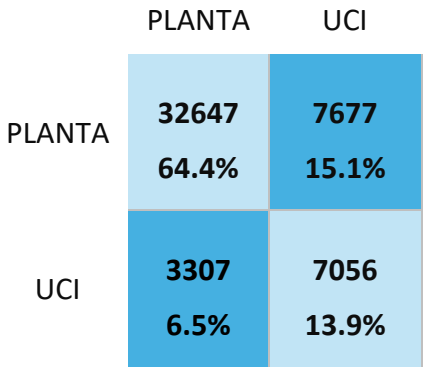

Figura 64: Matriz de confusión prueba 2 (CNN con LSTM)

Parámetros de calidad:

Precisión	90.8%
Especificidad	68.1%
Sensibilidad	81.0%
Tasa de acierto	78.3%

Tabla 76. Parámetros de calidad Prueba 2 (CNN con LSTM)

Prueba 3:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	100

Tabla 77. Prueba 2. Hiperparámetros (CNN con LSTM)

Matriz de confusión:

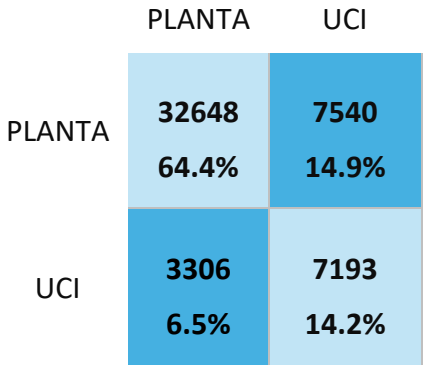

Figura 65: Matriz de confusión prueba 3 (CNN con LSTM)

Parámetros de calidad:

Precisión	90.8%
Especificidad	68.5%
Sensibilidad	81.2%
Tasa de acierto	78.6%

Tabla 78. Parámetros de calidad Prueba 3 (CNN con LSTM)

Prueba 4:

Hiperparámetro	Valor
Train Ratio	0.6
Initial Learn Rate	0.001

Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	125

Tabla 79. Prueba 4. Hiperparámetros (CNN con LSTM)

Matriz de confusión:

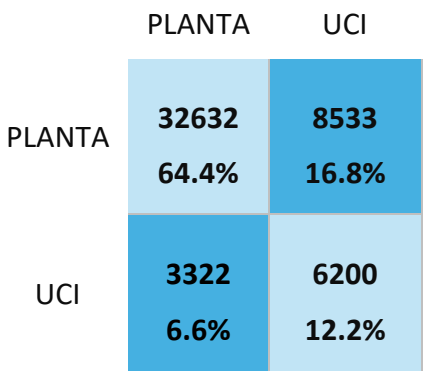

Figura 66: Matriz de confusión prueba 4 (CNN con LSTM)

Parámetros de calidad:

Precisión	90.8%
Especificidad	65.1%
Sensibilidad	79.3%
Tasa de acierto	76.6%

Tabla 80. Parámetros de calidad Prueba 4 (CNN con LSTM)

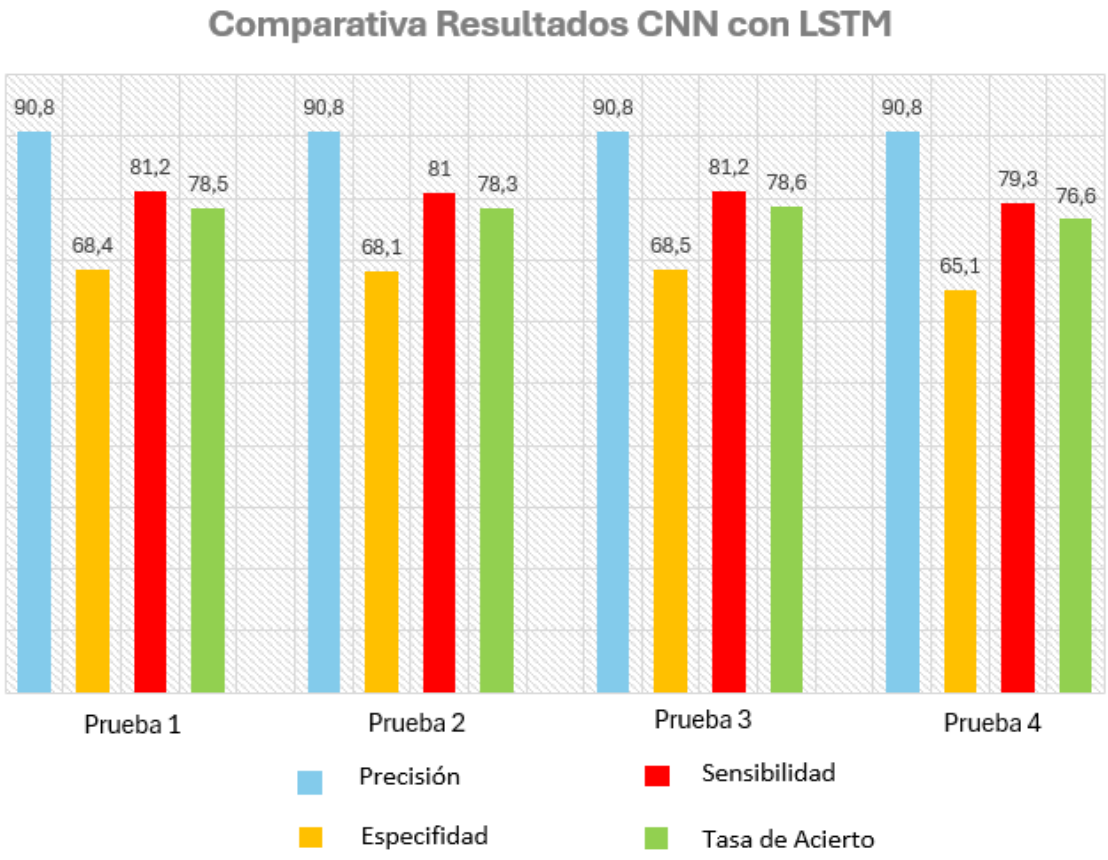

Figura 67: Comparativa Prueba 3 (CNN con LSTM)

En esta arquitectura hemos visto una notable mejoría en cuanto a los resultados se refiere. Nuestro mejor resultado anterior tenía una Tasa de acierto del 72.1%, por otro lado, ahora tenemos un valor del 78,6% en este valor. Los valores de la precisión se han mantenido estables en los cuatro casos, arrojando un resultado idéntico del 90.8%. En los cuatro casos los valores de la sensibilidad se han mantenido parecidos, exceptuando la última prueba que se han bajado unos 2 puntos con respecto al resto. Cabe destacar que el valor de la Especificidad se ha visto aumentado en gran medida con respecto a los resultados de las demás arquitecturas, rondando anteriormente el 52%, salvo en la red GRU que se llegó al 58%, y estando actualmente con esta arquitectura en un 68,5%. Podemos concluir que la prueba 3, hasta ahora, sería la mejor arquitectura.

6.2.5. Transformers

Prueba 1:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Tabla 81. Prueba 1. Hiperparámetros (Transformers)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**33024** **65.2%**	**5635** **11.1%**
UCI	**2930** **5.8%**	**9098** **17.9%**

Figura 68: Matriz de confusión prueba 1 (Transformers)

Parámetros de calidad:

Precisión	91.9%
Especificidad	75.6%
Sensibilidad	85.4%
Tasa de acierto	83.1%

Tabla 82. Parámetros de calidad Prueba 1 (Transformers)

Prueba 2:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	125

Tabla 83. Prueba 2. Hiperparámetros (Transformers)

Matriz de confusión:

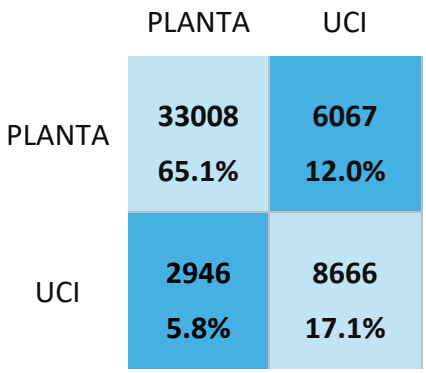

Figura 69: Matriz de confusión prueba 2 (Transformers)

Parámetros de calidad:

Precisión	91.8%
Especificidad	74.6%
Sensibilidad	84.5%
Tasa de acierto	82.2%

Tabla 84. Parámetros de calidad Prueba 2 (Transformers)

Prueba 3:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	7
Gradient Threshold	1
Number of Hidden Units	75

Tabla 85. Prueba 3. Hiperparámetros (Transformers)

Matriz de confusión:

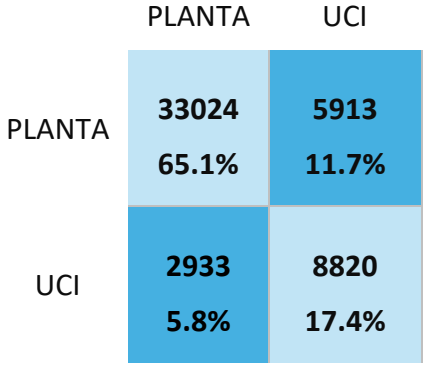

Figura 70: Matriz de confusión prueba 3 (Transformers)

Parámetros de calidad:

Precisión	91.8%
Especificidad	75.0%
Sensibilidad	84.8%
Tasa de acierto	82.5%

Tabla 86. Parámetros de calidad Prueba 3 (Transformers)

Prueba 4:

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.01
Max Epochs	5
Gradient Threshold	1
Number of Hidden Units	100

Tabla 87. Prueba 4. Hiperparámetros (Transformers)

Matriz de confusión:

	PLANTA	UCI
PLANTA	**33024** **65.2%**	**5772** **11.4%**
UCI	**2930** **5.8%**	**8961** **17.7%**

Figura 71: Matriz de confusión prueba 4 (Transformers)

Parámetros de calidad:

Precisión	91.9%
Especificidad	75.4%
Sensibilidad	85.1%
Tasa de acierto	82.8%

Tabla 88. Parámetros de calidad Prueba 4 (Transformers)

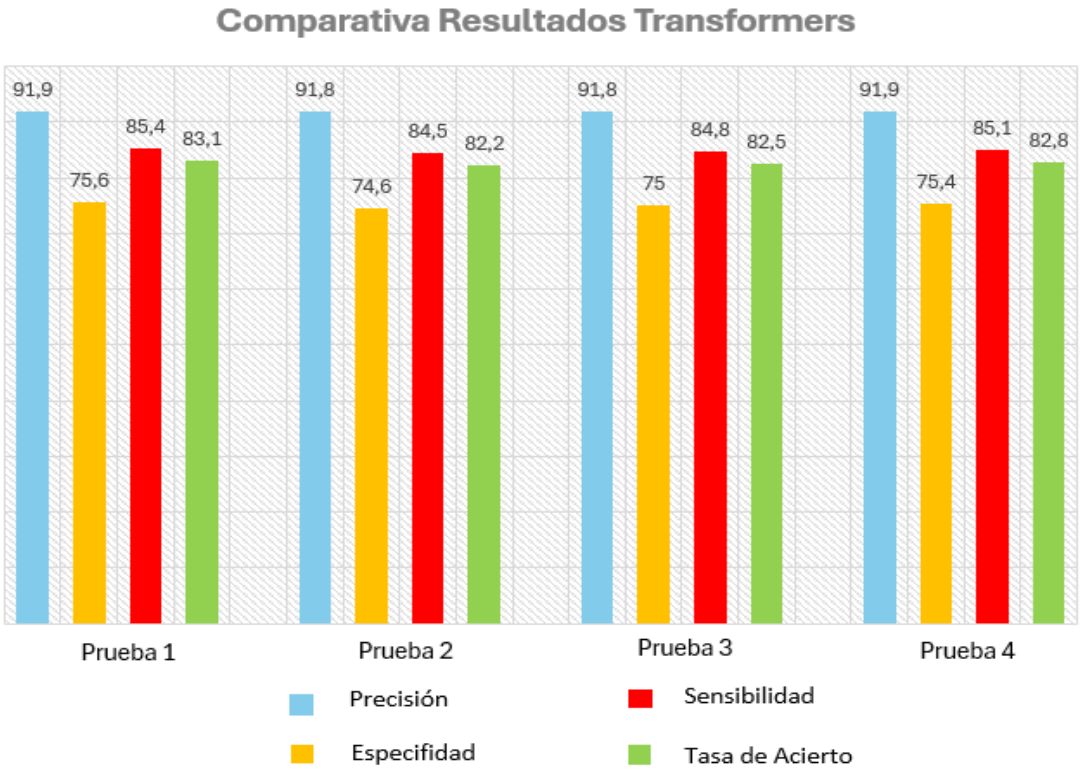

Figura 72: Comparativa Prueba 4 (Transformers)

Para la última arquitectura hemos podido observar los mejores resultados. De nuevo hemos tenido una estabilidad en los resultados de la precisión, siendo los 4 casi idénticos, con un valor muy elevado de casi el 92%. Por otro lado, el valor de la especificidad ha aumentado con respecto a la última. Comenzamos las pruebas con los mejores valores en torno al 52% y llegando al 58%, y ahora, estaríamos en un valor alrededor del 75,5%. Por último, la sensibilidad se ha visto muy estabilizada, subiendo a más del 85%. Quiero destacar la mejora de resultados en cuanto a los falsos negativos se refiere, pasando de tener valores alrededor del 20% en los mejores resultados, a tener un 11,1%. Podemos concluir que la prueba 1 sería la mejor opción hasta ahora.

6.3. Discusión

6.3.1. Elección del modelo

Tras analizar cinco arquitecturas diferentes, hemos obtenido resultados significativamente distintos. En las primeras tres arquitecturas se ha presentado un mayor número de resultados en comparación con las últimas dos, debido al considerable costo computacional que implica reproducir los resultados de cada una. Para las primeras tres arquitecturas, el tiempo de procesamiento fue similar, aproximadamente 25 minutos por prueba. Sin embargo, para la arquitectura CNN con LSTM, los tiempos de procesamiento aumentaron considerablemente, alcanzando los 90 minutos, y para Transformers, los 120 minutos. Este factor de tiempo de procesamiento es crucial a la hora de seleccionar un modelo específico.

La arquitectura de Transformers ha demostrado ser la más eficaz, con la Prueba 1 alcanzando una tasa de acierto del 83.1% y una precisión de casi el 92%. Además, con una sensibilidad del 84.5%, el número de falsos negativos es muy reducido, lo cual es crítico en situaciones como la pandemia, donde las UCI se vieron desbordadas. Esta arquitectura resulta ser la más adecuada para predecir la clasificación correcta, minimizando los falsos negativos y, por ende, es una opción preferida para situaciones similares.

Por otro lado, la arquitectura CNN con LSTM también ha mostrado resultados prometedores. Aunque requiere menos tiempo y recursos, sus resultados no superan a los de la arquitectura Transformers, pero sí son superiores a los obtenidos con GRU, LSTM y BiLSTM. Es notable su alta precisión, alcanzando casi el 91%, manteniendo además buenos valores en otros parámetros.

Finalmente, entre las arquitecturas GRU, LSTM y BiLSTM, que demandan un menor costo computacional, la GRU destaca por sus resultados superiores en términos de tasa de acierto y sensibilidad. En escenarios donde los recursos computacionales son limitados, la arquitectura GRU se presenta como una excelente alternativa, sobresaliendo frente a LSTM y BiLSTM.

Es importante destacar que un porcentaje de entrenamiento inferior al 50% o superior al 60% en cualquier arquitectura produce resultados subóptimos. Generalmente, los mejores resultados se obtienen con porcentajes de entrenamiento cercanos al 50% y con valores de hiperparámetros bajos, dentro de los rangos establecidos inicialmente.

En conclusión, dado que el objetivo de este estudio es encontrar la manera más eficiente de clasificar datos, optamos por la arquitectura de Transformers y los valores de la Prueba 1 como la mejor opción.

Prueba 1: **Transformers**

Hiperparámetro	Valor
Train Ratio	0.5
Initial Learn Rate	0.001
Max Epochs	5
Gradient Threshold	0.9
Number of Hidden Units	75

Matriz de confusión:

	PLANTA	UCI
PLANTA	**33024** **65.2%**	**5635** **11.1%**
UCI	**2930** **5.8%**	**9098** **17.9%**

Parámetros de calidad:

Precisión	91.9%
Especificidad	75.6%
Sensibilidad	85.4%
Tasa de acierto	83.1%

6.3.2. Comparativa con estudios previos

Tras analizar los datos obtenidos en este trabajo, resulta imprescindible compararlos con estudios previos para evaluar si realmente se han mejorado los modelos existentes hasta la fecha.

En primer lugar, se consideran los artículos que utilizan técnicas de *Machine Learning* [5][8]. En el estudio publicado por la Universidad de Chile, se presentaron dos modelos diferentes, diferenciados por la fuente de datos utilizada. En uno de estos modelos se logró una precisión del 96%, un dato altamente prometedor. No obstante, los demás indicadores mostraron resultados inferiores, alcanzando una tasa de acierto máxima del 74%.

Por otro lado, en el estudio realizado en Baleares [9], se propusieron distintos resultados según el día en que se tomaron las muestras (2, 3 o 5 días antes del evento de fallecimiento o alta). En uno de los casos, se obtuvo una sensibilidad del 88.7%, superior a la nuestra, aunque en ese mismo caso la precisión disminuyó a un 62.2%. Los demás resultados de ese estudio no superaron los nuestros en otros aspectos, lo que nos lleva a concluir que, aunque los algoritmos de *Machine Learning* ofrecen resultados prometedores, carecen de estabilidad, haciendo que no sean la mejor opción en este contexto específico.

En los estudios de la Universidad Politécnica de Valencia [7] y de la Revista de Investigación Científica, Tecnológica e Innovación [6], se emplearon técnicas de regresión y de *Deep Learning*. El estudio de la Universidad Politécnica de Valencia se centró en etiquetar a los pacientes según el desenlace del fallecimiento (sí/no). El predictor propuesto alcanzó una precisión del 74.8% y una tasa de acierto del 79.7% en su mejor modelo, ubicándose por debajo de nuestros resultados.

En el caso de la Revista de Investigación Científica, Tecnológica e Innovación, se utilizaron técnicas de regresión logística y una red neuronal artificial profunda, utilizando una base de datos de 13,757,682 registros de pacientes. El modelo de regresión logística alcanzó una tasa de acierto del 77.1%, con una precisión del 81.1% y una sensibilidad por debajo del 70%. Por su parte, el modelo de red neuronal logró una precisión del 83.2%, con una sensibilidad cercana al 70% y una tasa de acierto del 77.5%. En ambos casos, nuestro modelo con la arquitectura Transformers superó estos resultados.

Finalmente, al comparar con los trabajos realizados en esta universidad [10][11], el primer estudio alcanzó una tasa de acierto y una sensibilidad cercanas al 60%, con una precisión del 89%. En el segundo estudio, se obtuvo una precisión del 94.5%, una sensibilidad del 73.8% y una tasa de acierto del 72.0%. En este último caso se utilizó GRU para construir el modelo, mostrando resultados similares a los nuestros para esta arquitectura. Como se mencionó anteriormente, tanto CNN con LSTM como Transformers superaron estos resultados, aunque con un mayor gasto computacional.

6.3.3. Opinión médica

Se han consultado los resultados de este estudio con el personal médico del Complejo Hospitalario Universitario Insular Materno Infantil de Las Palmas de Gran Canaria encargado de proporcionar la base de datos y con el que hemos podido colaborar, realizando el siguiente análisis a continuación:

Los resultados obtenidos del modelo son de gran relevancia desde el punto de vista médico y de gestión de recursos hospitalarios. A continuación, se analiza la

importancia de estos resultados en relación con las pautas establecidas por los médicos:

Predicción de Necesidades de Ingreso:

El modelo permite identificar a los pacientes con características específicas que requerirán ingreso en la planta o en la UCI. Esta capacidad predictiva es fundamental para anticipar las necesidades de tratamiento y los recursos que serán necesarios.

La alta precisión del modelo (91.9%) para predecir ingresos en UCI asegura que la mayoría de los pacientes identificados realmente necesitarán cuidados intensivos, lo que permite una planificación más precisa y efectiva.

Optimización de la Gestión de Recursos:

La capacidad del modelo para predecir de manera precisa y anticipada qué pacientes necesitarán ingreso en UCI es crucial para la gestión de recursos hospitalarios. Permite reducir los tiempos de espera y optimizar el tiempo de tratamiento, lo cual puede ser más efectivo en algunas unidades.

La identificación temprana de los pacientes que requerirán cuidados intensivos permite anticipar el tratamiento y preparar los recursos necesarios, mejorando la eficiencia operativa del hospital.

Atención a Pacientes Potencialmente Más Graves:

La predicción precisa de los pacientes que necesitarán ingreso en UCI también ayuda a identificar a aquellos que potencialmente son más graves y que requerirán un mayor número de pruebas médicas y tratamientos específicos disponibles solo en la UCI. Esto permite un enfoque más dirigido y personalizado en la atención médica, asegurando que los pacientes más graves reciban la atención que necesitan en el momento adecuado.

Optimización y Organización de Recursos:

La implementación del modelo contribuye a una mayor optimización de los recursos hospitalarios. Permite una organización y programación más eficiente de los recursos según los requerimientos de los pacientes, basándose en el destino previsto (UCI o planta).

Esta optimización se traduce en una mejor utilización de los recursos disponibles, reduciendo el estrés en el sistema hospitalario y mejorando la calidad del cuidado que se proporciona a los pacientes.

Conclusiones

El modelo predictivo desarrollado ofrece una herramienta valiosa para la gestión de pacientes hospitalizados por COVID-19, permitiendo una mejor identificación y manejo de los casos que requieren ingreso en UCI. Su alta precisión y exactitud aseguran que los pacientes que necesitan cuidados intensivos sean identificados y atendidos de manera oportuna.

El uso del modelo no solo mejora la eficiencia en la gestión de recursos hospitalarios, sino que también optimiza el cuidado del paciente, reduciendo tiempos de espera y mejorando la efectividad de los tratamientos. La capacidad de predecir y planificar de manera anticipada los requerimientos de los pacientes contribuye significativamente a la calidad y eficiencia del sistema de salud.

Capítulo 7: Conclusiones y líneas futuras

7.1. Conclusiones

En este estudio se han desarrollado y evaluado diversos modelos de predicción. Todos basados en redes neuronales, se han empleado métodos más y menos complejos, con aspectos tanto positivos como negativos. Hagamos un pequeño resumen.

Tras analizar cinco arquitecturas diferentes, hemos dejado claro que, si dispones de bajos recurso computacionales, la elección de GRU es una muy buena opción. Como se reflejó anteriormente, con tasas de acierto superiores al 72%, te aseguras de tener tanto fiabilidad como resultados estables, según la prioridad propia claro. Sin embargo, como el objetivo de este estudio es encontrar la mejor manera dentro de los recursos disponibles, la mejor arquitectura es sin duda Transformers. Siendo el mejor resultado una tasa de acierto de casi el 83%, se puede comprobar que se ha avanzado mucho en el objetivo de mejorar los sistemas de clasificación para estos casos.

En el apartado 1.4 de este trabajo se expusieron los objetivos para este Trabajo de Fin de Grado, siendo estos los siguientes:

- O1: Investigar y examinar las últimas técnicas empleadas en la predicción de la afección de la COVID.
- O2: Evaluar y categorizar la información contenida en la base de datos proporcionada.
- O3: Aplicar técnicas de aprendizaje profundo para prever el desarrollo futuro de los pacientes con COVID.

El primer objetivo se ha cumplido, quedando reflejado en el apartado 1.2 de trabajos previos. Más adelante, en el capítulo 3, se ha querido dar más profundidad a los conceptos y métodos empleados en trabajos anteriores, explicando conceptos desde inteligencia artificial hasta redes neuronales profundas.

El segundo objetivo también se ha cumplido, exponiendo la categorización de los datos y la visualización de estos, tanto previa como posteriormente a los cambios realizados.

Se ha tratado de analizar y clasificar los datos, dejando a un lado datos que nos podrían dar falsos resultados o no tan específicos y reales como nos gustaría.

El último objetivo también se ha cumplido, llegando a aplicar hasta cinco técnicas de clasificación diferentes para poder encontrar así el resultado óptimo. Tanto en el capítulo 3, donde he explicado los conceptos de las arquitecturas que he empleado en el trabajo, como en el capítulo 5, donde se ve reflejada la implementación de cada arquitectura, realizando un análisis individual de cada arquitectura por separado

Además, el desarrollo de este proyecto ha permitido al autor profundizar en el conocimiento sobre inteligencia artificial, *Machine Learning* y *Deep Learning*. Como resultado, se ha conseguido crear una herramienta altamente fiable que facilita la toma de decisiones del personal sanitario, especialmente en situaciones de desbordamiento de las UCI. Este estudio no solo busca aliviar la carga de los profesionales de la salud, sino también ha sido impulsado por una gran motivación personal para contribuir positivamente en la gestión de crisis sanitarias. La implicación en este proyecto ha sido una experiencia enriquecedora, proporcionando tanto un crecimiento académico como, espero sea también, una buena aportación al ámbito de la medicina.

7.2. Líneas futuras

Para continuar con la evolución y mejora del presente trabajo, se pueden considerar varias líneas de acción que permitirán optimizar los resultados y ampliar el impacto del proyecto en la práctica clínica.

En primer lugar, se sugiere incrementar el número de etiquetas utilizadas en la clasificación. En esta investigación se emplearon únicamente las etiquetas "UCI" y "Planta". Sin embargo, la incorporación de etiquetas adicionales como "Alta" y "Fallecimiento" podría proporcionar un marco más completo y realista para la predicción de la evolución de los pacientes. Estas etiquetas adicionales permitirán a los modelos de predicción ajustarse más precisamente a los posibles desenlaces clínicos, mejorando así la exactitud y utilidad de las predicciones.

Además, se recomienda expandir la base de datos no solo en términos del número de pacientes, sino también en cuanto a la variedad y cantidad de datos recopilados. Incluir información detallada sobre los síntomas presentados por los pacientes al ingreso y su evolución durante el tratamiento podría enriquecer los modelos predictivos. Esta información adicional permitiría una evaluación más completa de los factores que influyen en la evolución de la enfermedad y, por ende, mejorar la precisión de las predicciones.

Un enfoque colaborativo y alineado con las necesidades del personal sanitario es esencial. Por ello, sería beneficioso mantener reuniones periódicas, especialmente al inicio del proyecto, para asegurar un enfoque común. Estas reuniones permitirán ajustar las metodologías y objetivos del proyecto de acuerdo con las necesidades y expectativas de los profesionales de la salud, asegurando que los resultados obtenidos sean de máxima relevancia y aplicabilidad.

Asimismo, para manejar los crecientes volúmenes de datos y la complejidad de los modelos de predicción, es fundamental contar con un hardware más potente. Utilizar ordenadores con mayor capacidad de procesamiento permitirá ejecutar modelos más complejos y entrenarlos de manera más eficiente, reduciendo los tiempos de procesamiento y aumentando la capacidad para manejar grandes conjuntos de datos.

Por último, una profundización en las arquitecturas de redes neuronales es crucial. Este campo ha avanzado significativamente en los últimos años y presenta una gran proyección de futuro. Explorar y aplicar las últimas innovaciones en arquitecturas de redes neuronales podría llevar a mejoras sustanciales en la precisión y fiabilidad de los modelos predictivos. La investigación y desarrollo continuos en este ámbito asegurarán que el proyecto se mantenga a la vanguardia de la tecnología en inteligencia artificial aplicada a la salud.

Referencias

[1] D. E. G. Agustín Julián-Jiménez, «Acerca de cómo los servicios de urgencias españoles hicieron frente a la primera oleada de pacientes durante la pandemia COVID-19,» 2020.

[2] A. M. D. C. ASISTENCIAL, «PROTOCOLO PARA LA GESTIÓN HOSPITALARIA DE LA PANDEMIA DE SARS-CoV-2 (COVID-19),» 2020.

[3] A. C. A. Carmen Caballero Dominguez, «Problemas de salud mental en la sociedad un acercamiento desde el impacto del COVID 19 y de la cuarentena,» Septiembre 2020.

[4] J.H.U.o. Medicine, «COVID-19 Dashboard,» 2023. https://coronavirus.jhu.edu/map.html

[5] J. A. Sepúlveda Rodríguez, «Modelos de predicción para la evolución de pacientes Covid-19,» *Universidad de Chile*, 2021.

[6] A. Bautista Loaiza y F. J. Ávila Camacho, «Predicción de la condición de hospitalización para pacientes Covid-19 utilizando modelos de clasificación », *RICT CCAI*, vol. 2, n.º 3, pp. 1–5, abr. 2024.

[7] B. B. Viciano, «Desarrollo de un algoritmo automático para la predicción de evolución de pacientes hospitalizados por COVID-19,», *Universidad Politécnica de Valencia*, 2021.

[8] M. E. S.-V. L. R.-G. E. L.-D. A. Monzón-Pérez, «Estado en la investigación sobre modelos de predicción de la severidad en confirmados de la Covid-19,» *Revista*

Habanera de Ciencias Médicas, Febrero 2023.

[9] J. A. P. G. C. R. M. R. V. Tarun Khajuria, «COVID-19: Predicción de eventos de hospitalización en UCI /muerte usando registros médicos de atención y herramientas de aprendizaje automático,» 2020.

[10] S. C. Bernal, «Análisis de datos COVID-19 para predecir su evolución,» *Universidad de las Palmas de Gran Canaria,* Enero 2022.

[11] A. S. B. Padrón, «EVOLUCIÓN PACIENTES COVID-19 MEDIANTE APRENDIZAJE PROFUNDO,» *Universidad de las Palmas de Gran canaria,* Julio 2023.

[12] A. T. S. F. S. Q. A. [. Julio Cesar Ponce Gallegos, Inteligencia Artificial, Iniciativa Latinoamericana de Libros de Texto Abiertos.

[13] S. S. L. J. Escuela Especializada en Ingeniería (ITCA-FEPADE), Algoritmos de aprendizaje automático para análisis y predicción de datos, ITCA Editores.

[14] P. a. S. K. Teja Nallamothu, «Machine Learning for SPAM Detection,» *Asian Journal of Advances in Research,* Enero 2024.

[15] A. F. Pineda Martínez, «Construcción de modelos de machine learning con aprendizaje supervisado para determinar la deserción académica en estudiantes universitarios,» Repositorio Institucional, Febrero 2022.

[16] J. L. Fernández Genaro, «Modelos de Machine Learning para la Ciencia de Datos,» Universidad de Salamanca, 2023.

[17] M. L. Errecalde, «Marcos teóricos del aprendizaje por refuerzo multiagente,» Red de Universidades con Carreras en Informática (RedUNCI), 2001.

[18] IMB, «¿Qué es el deep learning?,» https://www.ibm.com/es-es/topics/deep-learning.

[19] E. Mancilla, «IA vs. machine learning vs. deep learning vs. redes neuronales: ¿Cuál es la diferencia?,» https://blog.invgate.com/es/ia-vs-machine-learning-vs-deep-learning-vs-redes-neuronales, Abril 2023.

[20] C. Arana, «Redes neuronales recurrentes: Análisis de los modelos especializados en datos secuenciales,» *Econstor in colaboration with University of CEMA, Buenos Aires,* nº 797, 2021.

[21] C. A. Cortés, «Herramientas modernas en redes neuronales: la librería Keras,» UAM. Departamento de Ingeniería Informática, Enero, 2017.

[22] MathWorks, «Redes neuronales de memoria de corto-largo plazo,» https://es.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html.

[23] J. López Segura, «Análisis de las redes neuronales recurrentes: enfoque en las LSTM y GRU para predicción,» Universidad Complutense de Madrid, 2023.

[24] A. F. Nasimba Tipan, "Attention is all you need". Arquitectura Transformers: descripción y aplicaciones, Universidad Miguel Hernández de Elche, Junio, 2023.

[25] S. C.-B. A. L. F. C.-Q. Carlos M. Travieso, «Analysis of variables to determine their influence on renewable energy forecasting using ensemble methods,» Abril, 2022.

[26] P. D. L. Castelló, «Metodología experimental,» de *Metodología de la investigación*.

[27] S. Calixto Aldama, «ESTUDIO COMPARATIVO DE HERRAMIENTAS PARA REDES

NEURONALES ARTIFICIALES (RNA): WEKA, MATLAB Y NEUROSOLUTIONS,» Universidad Autónoma del Estado de México, Julio, 2017.

[28] I. Zumárraga Eguidazu, «REDES NEURONALES EN MATLAB,» Universidad del País Vasco, 2019.

[29] "La matriz de confusión y sus métricas – Inteligencia Artificial –." https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/ (accessed Jan. 26, 2022).

[30] "Cómo interpretar la matriz de confusión: ejemplo práctico - Think Big Empresas." https://empresas.blogthinkbig.com/como-interpretar-la-matriz-de-confusion-ejemplo-practico/ (accessed Jan. 26, 2022).

PARTE 2: PRESUPUESTO

Desglose del presupuesto

Para la elaboración del presupuesto de este Trabajo de Fin de Grado se han considerado los precios del mercado vigente, así como las directrices proporcionadas por el Colegio Oficial de Graduados e Ingenieros Técnicos de Telecomunicación (COITT). Este informe incluye una estimación detallada de los gastos incurridos durante su realización, basándose en los conceptos que se describen a continuación:

1. Recursos Materiales
2. Trabajo tarifado por tiempo empleado
3. Costes asociados a la redacción del Trabajo Fin de Grado
4. Derechos del visado del COITT
5. Gastos de tramitación y envíos
6. Gastos derivados de los impuestos

1. Recursos materiales

Entre los recursos materiales empleados para la ejecución y desarrollo de este Trabajo de Fin de Grado se consideran tanto los recursos hardware como los recursos software, los cuales pueden implicar costes asociados a la adquisición de licencias para su utilización. Para estimar el coste de amortización, se establece un período de 4 años, utilizando un método de amortización lineal, donde los activos fijos se deprecian de manera uniforme a lo largo del período considerado. Para efectuar este cálculo, se aplica la siguiente fórmula:

$$Coste\ de\ amortización = \frac{Coste\ de\ adquisición - Valor\ residual}{Años\ de\ vida\ útil}$$

Se entiende como valor residual el coste que tendrá el material después de que haya pasado su vida útil. Como el presente proyecto tiene una duración de 300 horas distribuidas de manera aproximada en 4 meses, y este período es inferior a los 5 años del coste estipulado de amortización, dicho coste será el derivado de los 4 meses en los que se desarrolla el proyecto.

1.1. Recursos de hardware

Entre los recursos de *hardware* empleados para la realización de este TFG se encuentra:

- Ordenador portátil HP Pavilion 15-cs0xxx

Recurso	**Valor de adquisición (€)**	**Valor residual (€)**	**Coste de amortización (€)**
Ordenador Portátil	800	265	66.88€

El coste de amortización libre de impuesto correspondiente a la parte de hardware será de **sesenta y seis euros con ochenta y ocho céntimos (66,88€).**

1.2. Recursos de Software

Para este trabajo de Fin de Grado se emplearán los siguientes recursos de Software:

- MATLAB
- Microsoft Office 365

En nuestro caso al pertenecer a la comunidad universitaria de Las Palmas de Gran Canaria, no tenemos que pagar por su licencia ya que se dispone de una, por lo cual los costes de amortización serán nulos.

2. Trabajo tarifado por tiempo empleado

El proyecto se ha llevado a cabo durante un período de 4 meses, abarcando todas las tareas de formación, especificación, desarrollo y documentación necesarias para su ejecución. Para estimar los honorarios que corresponden al ingeniero por el trabajo realizado, se siguen las recomendaciones del Colegio Oficial de Graduados e Ingenieros Técnicos de Telecomunicación (COITT) y se emplea la siguiente fórmula:

$$H = Ct * 74{,}88 * Hn + Ct * 96{,}72 * He$$

Donde:
- *H* son los honorarios totales para percibir.
- *Ct* es un factor de corrección en función de las horas empleadas.
- *Hn* son las horas trabajadas dentro del horario laboral.
- *He* son las horas extras trabajadas fuera del horario laboral.

La duración de este proyecto tal y como se planificó en el anteproyecto en el Diagrama de Gantt tiene una duración de 300 horas. Para calcular el factor de corrección hay que mirarlo en la tabla publicada por el COITT en función del número de horas empleadas.

Horas	Factor de Corrección
Hasta 36	1
De 36 a 72	0.9
De 72 hasta 108	0.8
De 108 hasta 144	0.7
De 144 hasta 180	0.65
De 180 hasta 360	0.6

De 360 hasta 510	0.55
De 510 hasta 720	0.5
De 720 hasta 1080	0.45
Más de 1080	0.4

Una vez conocido el factor de corrección que habría que aplicar (en mi caso 0.6 debido a que son 300 horas) y teniendo en cuenta que no se ha trabajado fuera del horario laboral, podemos aplicar la formula anterior, resultando en unos honorarios de:

$$H = 0.6 * 74{,}88 * 300 + 0.6 * 96{,}72 * 0 = 13478.40€$$

Los honorarios totales por tiempo dedicado libres de impuestos ascienden a la cuantía de **trece mil cuatrocientos setenta y ocho euros y cuarenta céntimos (13.478,40 €).**

3. Costes asociados a la redacción del documento

El importe de la redacción del proyecto se calcula de acuerdo con la siguiente expresión:

$$R = 0.07 * P * Cn$$

Donde:

- P es el presupuesto del proyecto.
- Cn es el coeficiente de ponderación en función del presupuesto.

Para el presupuesto del proyecto, se han calculado los gastos de los apartados anteriores, los cuales suman un total de 13.545,28 €. Dado que el importe es inferior a 30.050,00 €, el coeficiente de ponderación para este presupuesto es de 1. Por lo tanto, al aplicar la fórmula B.3 obtendremos:

$$R = 0.07 * 13.545.28 * 1 = 948{,}17€$$

Los costes asociados a la redacción del documento tienen un importe libre de impuesto de novecientos cuarenta y ocho euros con diecisiete céntimos (948,17 €).

4. Derechos del visado del COITT

Los gastos de visado del COITT se tarifan mediante la siguiente expresión:

$$V = 0.006 * P * Cv$$

Donde:

- *P* es el presupuesto del proyecto.
- *Cv* es el coeficiente de ponderación en función del presupuesto.

Para calcular el presupuesto total, se suman todos los gastos acumulados hasta la fecha, resultando en un total de 14.493,45 €. Dado que este monto es inferior a 30.050,00 €, el coeficiente de ponderación aplicable es de 1. Por consiguiente, al emplear la fórmula B.5, obtendremos el coste correspondiente al visado del trabajo:

$$V = 0.006 * 14.493{,}45 * 1{,}00 = 86{,}96€$$

el coste de los derechos de visado del proyecto asciende a **ochenta y seis euros y noventa y seis céntimos (86,96 €).**

5. Gatos de tramitación y envío

Según el COITT , los gastos de tramitación y envío están estipulados en **seis euros (6,00€).**

6. Aplicación de impuestos

Al importe de nuestro presupuesto hay que sumarle una cantidad del 7 % derivado de aplicar el IGIC. Por lo tanto, el presupuesto final quedaría de la siguiente manera:

Coste total del proyecto	
Descripción	**Coste (€)**
Recursos materiales	66,88
Trabajo tarifado por tiempo empleado	13.478,40
Costes asociados a la redacción del documento	948,17
Derechos del visado del COITT	86,96
Gastos de tramitación y envío	6,00
Subtotal	**14.586,41**
Aplicación de impuestos (IGIC 7%)	1.021,05
Total	**15.607,46**

El presupuesto total de este Trabajo de Fin de Grado asciende a la cuantía de **quince mil seiscientos siete euros y cuarenta y seis céntimos** (15.607,46€).

Las Palmas de Gran Canaria a 20 de Julio de 2024

Firma:

Sergio Pinto Pérez

PARTE 3: PLIEGO DE CONDICIONES

Pliego de Condiciones

En esta sección se proporcionan los detalles sobre las condiciones en las que se llevó a cabo este Trabajo de fin de grado, incluyendo tanto los requisitos de hardware como los requisitos de software que se necesitó para su ejecución.

1. Requisitos Hardware

El portátil que se utilizó para la realización de este trabajo presenta las siguientes características:

- Tipo de dispositivo: Ordenador portátil.
- Fabricante: HP
- Procesador: 11th Gen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz 2.42 GHz
- Memoria RAM: 8,00 GB
- Tarjeta gráfica: NVIDIA GeForce MX350
- Almacenamiento: 512 GB

2. Requisitos Software

Para la realización y la redacción de este trabajo, se utilizó los softwares que se detallan a continuación:

- Microsoft Windows 11
- Microsoft Office 365
- Matlab R2021b

3. Recursos humanos

Para la realización de este proyecto se ha necesitado una base de datos, que ha sido cedida por el personal médico de la UCI del Complejo Hospitalario Universitario Insular Materno Infantil de Las Palmas de Gran Canaria. Para el seguimiento del Trabajo Fin de Grado se ha necesitado la ayuda de mis dos tutores, Carlos Travieso Y Sergio Celada.