Predicción de eventos de hiperglucemia e hipoglucemia en pacientes con diabetes tipo 1 mediante algoritmos de aprendizaje supervisado

Wilson Danilo Malaver Fonseca, Manuel José Narváez Guerra y Andrea Paola Reyes Carreño

Trabajo de Grado para Optar al Título de Ingeniero/a electrónico/a

Director

Rodolfo Villamizar Mejía

PhD. Tecnologías de la información

Codirector

Edward Alfonso Rodríguez Moreno

Magister en Ingeniería Electrónica

Universidad Industrial de Santander

Facultad de Fisicomecánicas

Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones

Ingeniería Electrónica

Bucaramanga

2023

Dedicatoria

Andrea Paola Reyes Carreño

Quiero dedicar este trabajo de grado a mi hermano Diego Alejandro Reyes, quien ha sido mi más grande motor. A mi madre, Gilma Andrea Carreño, quien ha sido mi roca y mi ejemplo a seguir, su amor incondicional y sabiduría han sido fundamentales en mi vida, a mi padre Diego Reyes por su dedicación y esfuerzo constante en brindarnos una vida digna y llena de oportunidades. Les agradezco de todo corazón por su amor, apoyo y sacrificio. Este logro también es de ustedes.

Quiero comenzar expresando mi agradecimiento a Dios por haber sido mi guía y apoyo en cada paso de este proceso, su sabiduría y protección me han permitido superar todos los desafíos que he enfrentado. Así mismo, agradezco especialmente a mis padres por su amor incondicional, apoyo y sacrificio durante este importante capítulo de mi vida. También, quiero agradecer a mi novio, cuyo entendimiento y motivación en los momentos más difíciles me han ayudado a alcanzar este logro. Agradezco a mis amigos Danilo Malaver y Manuel Narváez por su compañía y apoyo constante durante todo el proceso. Finalmente, mi profundo agradecimiento a mi codirector Edward Alfonso Rodríguez y director Rodolfo Villamizar por su dedicación, paciencia, conocimientos y sabiduría que han sido esenciales en el desarrollo de este proyecto.

Manuel José Narváez Guerra

Le quiero dedicar este trabajo a Sadis Guerra Contreras y Manuel Narváez Suárez, mis papás, que fueron mi motivación y los principales influyentes en el éxito de mi carrera, cuando no tenía ganas de estudiar o estaba desanimado solo tenía que pensar en ellos.

Agradezco principalmente a Dios y a los miembros de mi familia, a mis padres, que sin ellos estoy seguro de que no lo hubiera logrado, a pesar de no contar con los recursos económicos se las ingeniaron para darme una profesión. Deseo que la vida me dé la oportunidad de recompensarlos por todos sus sacrificios y que sigan estando orgullosos de mí. Agradezco a mis hermanas Sandry Melissa y Sadith Paola por ser incondicionales y por apoyarme.

La vida universitaria ha sido la etapa más desafiante de mi vida, pero también la más bonita y en la que he aprendido mucho. En ella he conocido amigos que serán de toda la vida y también futuros colegas en la vida profesional. Entre ellos les doy mis más sinceros agradecimientos a:

Mi mejor amigo, Héctor Felipe Gamboa, te convertiste en un hermano. Cuando llegué a Bucaramanga no sabía lo que era vivir en una ciudad, pero tú me guiaste y me ensañaste de que no solo se trata de estudiar, sino que hay que disfrutar del proceso.

A Deyber Manrique con quien nos imaginábamos proyectos e ideas de empresas, Alejandra Ramírez que siempre me hace reír, Gabriela Suárez con la que tengo largas conversaciones por llamadas. Gracias por hacer parte de este proceso.

A los miembros de este proyecto de grado. A Andrea Reyes, a Danilo Malaver por ser excelente amigo, compañero de trabajo y de estudio. Al codirector Edward Rodríguez y al director Rodolfo Villamizar por su tiempo, correcciones y recomendaciones.

Por último, a mi alma mater, la Universidad Industrial de Santander. Gracias por brindarme beneficios como comedores, los programas de tutorías, por la selección de ajedrez, por la biblioteca, por el préstamo del PC que me permitió el desarrollo de este proyecto, son muchas las ayudas que nos brindas y que no aprovechamos al máximo.

Wilson Danilo Malaver Fonseca

Quiero dedicar este trabajo de grado principalmente a las personas que me han ayudado, acompañado y apoyado durante este largo pero provechoso proceso de mi vida. En primer lugar, a mis padres Marlen Fonseca y Nelson Malaver, quienes me han brindado un apoyo incondicional en todo momento, al igual que mis hermanos Liliana y Julian Malaver, quienes siempre han estado dispuestos a apoyarme en todo lo que he necesitado, también a mi novia que siempre me ha apoyado en cada paso que doy. Gracias a ellos, hoy estoy aquí y he podido alcanzar muchas de las metas que me he propuesto.

Por otro lado, quiero expresar mi agradecimiento a la comunidad universitaria de la UIS, que ha contribuido significativamente a mi formación académica y personal. En particular, algunos profesores y compañeros han sido fundamentales durante mi proceso de aprendizaje. Entre ellos, quiero destacar a mis compañeros Andrea Reyes y Manuel Narváez, quienes han sido más que compañeros, amigos en todo el sentido de la palabra.

Finalmente, quiero agradecer especialmente a mi codirector Edward Rodríguez que nos brindó información y orientación importante al inicio y durante este proceso, también a mi director Rodolfo Villamizar por su dedicación y orientación en el desarrollo de este proyecto, así como por la transferencia de conocimientos y motivación que me han brindado. Sin su ayuda, este trabajo no habría sido posible.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

pág.

Glosario

Diabetes: se refiere a un grupo de enfermedades metabólicas (Diabetes tipo 1, 2 y gestacional) que afecta la forma en que el cuerpo utiliza la glucosa en la sangre, se origina por que el páncreas no sintetiza la cantidad de insulina que el cuerpo humano necesita (García, 2017).

Glucemia: es una variable importante que regula nuestro cuerpo humano, hace referencia a la concentración de glucosa presente en la sangre, sí los niveles se elevan por encima de un umbral determinado se denomina hipoglucemia, si descienden se le conoce como hiperglucemia (Ponce, 2009).

Insulina: es una hormona producida por el páncreas, su principal función es el mantenimiento de los valores adecuados de glucosa en sangre, permite que la glucosa entre al organismo y sea trasportada al interior de las células, en donde se transforma en energía para el funcionamiento adecuado de los diferentes órganos, músculos y tejidos del cuerpo humano (Higuera y Basina, 2021).

Resumen

Título: Predicción de eventos de hiperglucemia e hipoglucemia en pacientes con diabetes tipo 1 mediante algoritmos de aprendizaje supervisado.

Autor: Wilson Danilo Malaver Fonseca, Manuel José Narváez Guerra y Andrea Paola Reyes Carreño[**]

Palabras Clave: Glucosa, redes neuronales, predicción, diabetes, GAN, GRU, SVM.

Descripción: Este proyecto tiene como objetivo utilizar técnicas de aprendizaje automático para predecir eventos de hiperglicemia e hipoglicemia en pacientes diabéticos, a partir de datos de ingesta de carbohidratos, glucosa e insulina generados por el simulador UVA Padova. La utilización de estas técnicas permitirá predecir el comportamiento de la glucosa en un tiempo determinado.

Para llevar a cabo el proyecto, se realiza un preprocesamiento del conjunto de mediciones generados por el simulador con el fin de mejorar la calidad de los datos de entrada y a su vez la precisión y eficiencia de la red neuronal. Se entrenan tres algoritmos de aprendizaje automático: GAN, GRU y SVM. La Red Generativa Antagónica (GAN) es un algoritmo de aprendizaje profundo que se utiliza para generar nuevos datos similares a los datos de entrenamiento. La Unidad Recurrente Cerrada (GRU) es un tipo de red neuronal recurrente que se utiliza para

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones. Ingeniería Electrónica. Director: Rodolfo Villamizar Mejía. Doctor en Tecnologías de la información. Codirector: Edward Alfonso Rodríguez Moreno. Magister en ingeniería electrónica

procesar secuencias de datos. La Máquina de Vector Soporte (SVM) es un algoritmo de aprendizaje supervisado que se utiliza para clasificar datos en dos grupos. Los resultados obtenidos de este proyecto podrían tener un impacto significativo en la prevención de sucesos desfavorables en pacientes diabéticos, mejorando su calidad de vida y su tratamiento médico.

Abstract

Title: Prediction of hyperglycemic and hypoglycemic events in patients with type 1 diabetes using supervised learning algorithms.[*]

Author(s): Wilson Danilo Malaver Fonseca, Manuel José Narváez Guerra and Andrea Paola Reyes Carreño [**]

Key Words: Glucose, neural networks, prediction, diabetes, GAN, GRU, SVM.

Description:

This project aims to use machine learning techniques to predict hyperglycemic and hypoglycemic events in diabetic patients, based on data on carbohydrate intake, glucose, and insulin generated by the UVA Padova simulator. The use of these techniques will allow for the prediction of glucose behavior at a specific time.

To carry out the project, a preprocessing of the set of measurements generated by the simulator is performed in order to improve the quality of the input data and, in turn, the accuracy and efficiency of the neural network. Three machine learning algorithms are trained: GAN, GRU, and SVM. The Generative Adversarial Network (GAN) is a deep learning algorithm used to generate new data similar to training data. The Gated Recurrent Unit (GRU) is a type of recurrent neural network

---

[*] Degree Work

[**]Faculty of Physicomechanical Engineering. School of Electrical, Electronic and Telecommunications Engineering. Electronic Engineering. Director: Rodolfo Villamizar Mejía. PhD in Information Technology. Codirector: Edward Alfonso Rodríguez Moreno. Master in Electronic Enineering

used to process sequences of data. The Support Vector Machine (SVM) is a supervised learning algorithm used to classify data into two groups.

The results obtained from this project could have a significant impact on the prevention of unfavorable events in diabetic patients, improving their quality of life and medical treatment.

Introducción

En Colombia para el año 2020 se reportó que al menos 3 de cada 100 habitantes son diagnosticados con diabetes mellitus. Sin embargo, se estima que el número real es mayor debido a que algunas personas desconocen que padecen dicha enfermedad (de Colombia.MS, *s.f.*). Además, se sabe que no solo en Colombia sino a nivel mundial la diabetes es una de las principales causas de muerte, debido a la presencia de eventos de hiperglucemia e hipoglucemia, lo cual ocasiona casos de infartos, trombosis, deterioro de funciones del riñón, dificultad para que las heridas se curen naturalmente y amputación de algunas extremidades (Viloria *et al*., 2020).

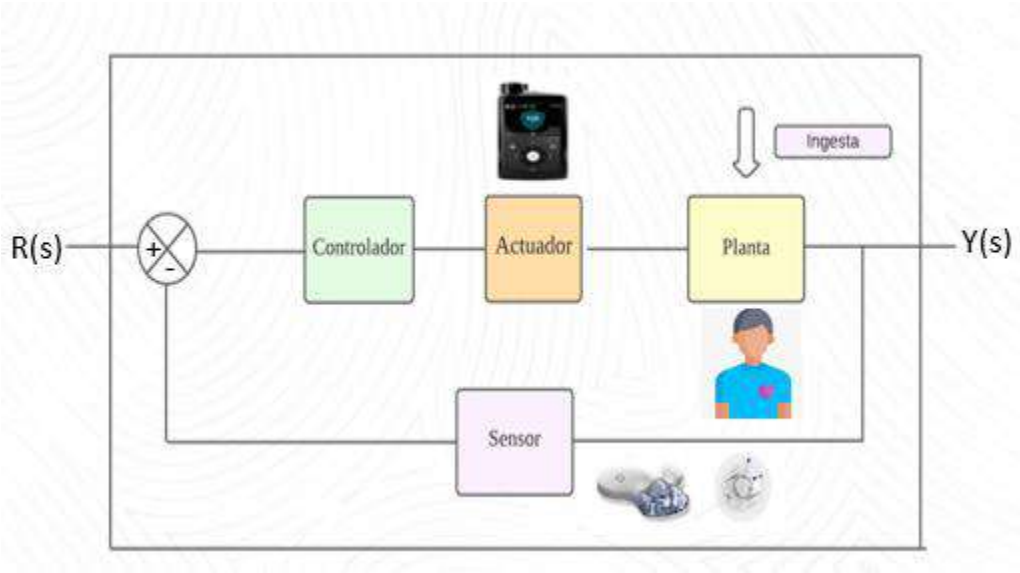
La diabetes de manera concreta se trata de una afección de salud que se presenta cuando los niveles de glucosa aumentan por encima de los niveles permitidos por el cuerpo. Como la glucosa es la principal fuente de energía en la sangre, se puede considerar como una variable dinámica dentro del cuerpo, ya que proviene directamente de los alimentos consumidos a diario. Por otro lado, la insulina es una hormona que se produce en el páncreas y es esencial para que la glucosa presente en el cuerpo pueda ingresar a las células y posteriormente transformarse en energía (CDC, 2022). Cuando el páncreas produce muy poca o nada de insulina necesaria para procesar la glucosa presente en el torrente sanguíneo, esta se acumula constantemente, y es ahí cuando se produce un alza de niveles de azúcar en la sangre. Dependiendo de la gravedad, se puede clasificar en tres tipos de diabetes, Tipo 1 (DT1), Tipo 2 (DT2) y Gestacional (DMG), siendo la Diabetes tipo 1 la más complicada donde el nivel de producción de insulina es mínimo o nulo. Por tanto, se debe suministrar de inmediato y constantemente insulina al cuerpo para el control de

diabetes y evitar que las complicaciones de salud aumenten hasta llegar a eventuales casos de muerte (OPS y OMS, *n.d.*).

Este proyecto está pensado para ser aplicado en un páncreas artificial. En la figura 1 se muestra el sistema de control de un páncreas artificial en el que su objetivo es mantener los niveles de glucosa en los valores aceptados por el cuerpo Bergman *et al*. (2019).

Figura 1

Sistema de control de un páncreas artificial



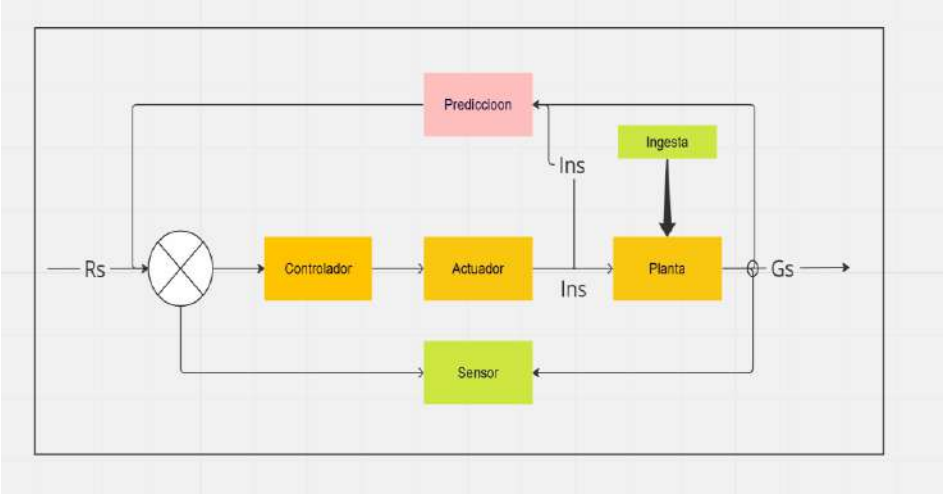
Nota. Se muestra los componentes del sistema de control de un páncreas artificial

Un problema que se detecta en el sistema de control de la figura 1 es que no es posible medir el momento ni la cantidad de carbohidratos consumidos por el paciente, por tanto, el controlador toma decisiones a partir del error que consiste en la diferencia entre un valor de

referencia constante y la concentración de glucosa medida. Por tanto, el sistema de control es reactivo a mantener dicho nivel constante, mientras que en un sujeto sano es una curva de glucosa. Esto puede llevar a que se apliquen altas dosis de insulina con la consecuencia de potenciales hipoglicemias o por el contrario que no se aplique a tiempo llevando a hiperglucemias. Para solucionar en parte este problema, en este proyecto se propone un bloque adicional (figura 2) capaz de predecir la futura curva de glucosa cuando hay ingesta de carbohidratos y así el páncreas artificial pueda seguir no una referencia constante, sino una curva equivalente de un sujeto sano cuando consume similar cantidad de carbohidratos.

Figura 2

Sistema de control de un páncreas artificial con un bloque de predicción



Nota. Se muestra los componentes del sistema de control de un páncreas artificial con un bloque adicional que predice los niveles de glucosa

Para la ejecución del trabajo de grado, se toman los datos generados a partir del simulador UVA Padova, el cual es capaz de proporcionar datos de pacientes in sílico, que contiene mediciones de variables como: glucosa, insulina e ingesta de carbohidratos (Visentin *et al*., 2018). Luego de obtener los datos se debe hacer un filtrado de estos con el fin de seleccionar los intervalos de tiempo relevantes y eliminar los datos que no aportan información. Para continuar con el proceso se debe crear un grupo de entrenamiento y un grupo de validación, la proporción de cada uno de estos grupos depende directamente del tamaño de la base de datos. Finalmente se entrenan 3 diferentes algoritmos de aprendizaje: GAN (Red Generativa Antagónica), GRU (Unidad Recurrente Cerrada) y SVM (Maquina de Vector Soporte), con el fin de evaluar y comparar la eficiencia de cada uno de ellos, buscando predecir de manera acertada el comportamiento de la glucosa en un tiempo determinado y así poder estimar la glucosa ingerida por el paciente y en función de ella intentar generar en el un comportamiento de glucosa en sangre similar al de un sujeto sano mediante el páncreas artificial.

1. Antecedentes

El monitoreo y control de glucosa en la sangre es vital para personas con diabetes. Se han realizado numerosos estudios a nivel local e internacional para encontrar y/o mejorar métodos que permitan un correcto monitoreo.

A nivel local, el grupo de investigación CEMOS en la Universidad Industrial de Santander ha hecho varios estudios respecto a este tema, como es el caso de Vejar *et al.* (2010) que propone un modelo que representa la dinámica de la concentración de la glucosa en la sangre para pacientes con diabetes mellitus tipo I.

A nivel nacional, Viloria *et al.* (2020) indica que los factores más importantes para la correcta detección de la diabetes son la edad, el índice de masa corporal y la concentración de glucosa en el torrente sanguíneo. Plantea que las SVM son un conjunto de algoritmos que son capaces de identificar y representar relaciones no lineales y se han caracterizado por tener un buen desempeño en problemas de regresión como clasificación. Además, menciona que las diferentes pruebas de sangre que se realizan normalmente en pacientes diabéticos no son del todo satisfactorias, por lo cual se implementa un algoritmo basado en SVM, donde se realizan las pruebas con datos de 500 pacientes de un hospital público de Colombia. El 80% del conjunto de mediciones se usó para el entrenamiento y el 20% para la validación, en base a los resultados obtenidos se obtuvo una precisión del 95,36% en pacientes colombianos.

Para el caso de predicción de glucosa en sangre, Narmadha *et al.* (2020) propone dos modelos para predecir enfermedades cardiacas en pacientes con diabetes, LSTM (Memoria a largo plazo) y GRU (Unidad Recurrente Cerrada), donde se resalta la gran eficiencia de la unidad recurrente cerrada en cuanto a obtener mejores resultados. Las variables empleadas para llevar el proceso de predicción son la identificación del paciente, código y valor que indica el tipo de diabetes, valor del índice glucémico, la herencia, medicación, presión arterial y finalmente el colesterol. Así mismo Zhang *et al.* (2021) plantea un modelo basado en GRU atencional donde se evidencia la eficacia de este modelo de predicción en comparación con LSTM Y SVM. A partir de la evaluación de desviación (RMSE) para el modelo propuesto se obtuvo un porcentaje del 26,1% en comparación con LSTM de 29,2% y SVM de 28,5% dentro de la predicción dinámica de 60 minutos.

Existen diferentes modelos que se basan en predicciones de glucosa en sangre, Zhu *et al.* (2020) presenta el diseño de un modelo de aprendizaje profundo para predecir glucosa en sangre usando una red generativa antagónica (GAN) para personas con diabetes tipo 1, a partir de la base de datos de ohioT1DM. Por otro lado, Wang *et al.* (2020) emplea un modelo con LSTM y VMD para predecir a partir de la recopilación de datos reales en diferentes pacientes, los resultados experimentales constataron la efectividad del método empleado y se obtiene un buen resultado de predicción con 60 minutos de anticipación.

Zhang *et al.* (2021) menciona que los diferentes modelos actuales utilizados para predecir glucosa en sangre tales como LSTM, GRU, ARIMA, LSSVM no pueden obtener un resultado óptimo en cuanto a predicción ya que se puede presentar ciertas características lineales y no lineales, por lo cual proponen un modelo hibrido ARIMA-LSSVM-GRU, el modelo ARIMA es

un modelo lineal utilizado para el análisis y predicción de series temporales, por otra parte LSSVM se basa en el reconocimiento de patrones, regresión de datos y series temporales, las GRU cuentan con la misma capacidad de extracción de datos. Para predecir glucosa en sangre, el modelo que se vuelva una situación muy delicada centra su investigación en el desarrollo de tres modelos basados en el aprendizaje automático como Maquina Vector Soporte, Regresión Logística y Redes Neuronales Artificiales.

2. Objetivos

2.1 Objetivo general

Predecir niveles de glucosa en la sangre para anticipar posibles eventos de hipoglucemia e hiperglucemia en pacientes con diabetes tipo 1 mediante la implementación de un algoritmo de aprendizaje supervisado.

2.2 Objetivos específicos

Sintetizar un conjunto apropiado de mediciones de ingesta de carbohidratos, glucosa e insulina mediante el simulador UVA/PADOVA.

Seleccionar y entrenar las arquitecturas de red neuronal tipo GAN, GRU y un algoritmo de SVM, teniendo en cuenta el desempeño de cada algoritmo.

Validar numéricamente el desempeño de la red entrenada usando parámetros de desempeño.

Validar en tiempo real el algoritmo con mejor desempeño mediante su programación en una unidad de procesamiento.

3. Marco conceptual

3.1 Diabetes

La diabetes es una enfermedad crónica que se caracteriza por un aumento persistente de los niveles de glucosa en la sangre. Esto ocurre debido a que el cuerpo no produce suficiente insulina o no la utiliza de manera efectiva (Keays, 2007). Según la Organización Panamericana de la Salud (OPS, *s.f.*), "la diabetes es una de las principales causas de ceguera, insuficiencia renal, ataques cardiacos, derrames cerebrales y amputación de miembros inferiores" (párr. 4).

Estudios realizados por la Organización Mundial de la Salud (OMS, 2022), refiere que entre el año 2000 hasta 2019 las tasas de mortalidad por esta enfermedad en personas de diferentes edades aumentaron un 3%, y en los países de medianos o pocos ingresos incremento un 13%. Por otra parte, la Décima edición 2021 del atlas de la diabetes de la de la Federación Internacional de Diabetes (FID, 2021) menciona que la diabetes causo 6,7 millones de muertes en el 2021 y proporciona las cifras más actuales en cuanto a esta enfermedad, donde se evidencia que 537 millones de adultos y 1,2 millones de niños y adolescentes tienen diabetes, y a su vez prevé que para 2030 y 2045 la diabetes aumente a 643 millones y 783 millones respectivamente.

La FID (2021) en base a sus causas define tres tipos de diabetes.

Diabetes tipo I: Es causada por una reacción autoinmune del cuerpo humano, donde son atacadas las células que producen la insulina, lo que conlleva al cuerpo a producir muy poca, puede afectar a personas con diferentes edades, pero en especial a niños y jóvenes, las personas que la

padecen deben inyectarse insulina en su día a día para mantener los niveles adecuados de glucosa en la sangre. Un 10% del total de personas con diabetes presentan diabetes tipo I.

Diabetes tipo II: Se produce cuando el cuerpo no responde de manera adecuada a la insulina, generando que los niveles de glucosa en sangre se eleven y por consiguiente el páncreas libere más insulina de lo normal, es diagnosticada en adultos mayores, pero a menudo se evidencia en jóvenes y niños debido a la mala alimentación, poco ejercicio físico y problemas de obesidad. Un 90% del total de personas con diabetes presentan diabetes tipo II.

Diabetes Gestacional: Ocurre durante el embarazo, se caracteriza por que la madre no puede utilizar o en algunos casos producir toda la insulina que requiere durante esta etapa, suele diagnosticarse mediante pruebas prenatales, algunos mecanismos para contrarrestar este padecimiento pueden incluir planes de comidas especiales, actividad física o inyecciones de insulina.

Las personas que padecen diabetes tipo 1 necesitan administrarse insulina para mantener niveles óptimos de glucosa en sangre. El centro para el control y prevención de enfermedades (CDC, 2022); FID (2021) refiere los tipos de insulina:

De acción rápida: Se administra antes de cada comida, limita el azúcar en la sangre que se genera por la ingesta de comidas, la insulina de acción rápida incluye Asparat, Glulisine y Lispro.

De acción corta: Se toma 30 a 60 minutos antes de cada comida, la insulina de acción corta incluye Actrapid, Humulin R e Insuman Rapid.

De acción intermedia: Se toma justo con una insulina de acción corta, actúa en un intervalo de tiempo de 1 a 7 horas máximo, la insulina de acción intermedia incluye Humulin NPH, Protaphane e Insulatard.

De acción prolongada: Actúa en el cuerpo hasta 24 horas, en la mayoría de los casos suele suministrarse en las horas de la mañana o en la noche antes de acostarse, la insulina de acción prolongada incluye Detemir y Glargine.

De Acción Ultraprolongada: Suministra insulina por periodos largos de tiempo.

Premezclada: El perfil de acción es una combinación de la insulina de acción corta e intermedia.

3.1.1 Medición de Glucosa

Existen diversas formas que le permiten a un paciente medir y conocer si el nivel de glucosa presente en la sangre es alto o bajo. Inicialmente se cuenta con pruebas de sangre en laboratorio, donde un profesional de la salud toma una muestra de sangre en uno de los brazos. También existen dispositivos como glucómetros que es una técnica más sencilla, se basa en la obtención de una gota de sangre producto de un pinchazo en uno de sus dedos por medio de una lanceta, esta pequeña muestra se fija sobre una tira reactiva y posteriormente se introduce en el medidor (CDC, 2022). Otras alternativas son los sistemas flash de monitorización y los sistemas de monitorización continuos de glucosa (MCG). El primero consta de un sensor ubicado en la parte superior del brazo que mide en tiempo real la glucosa y permite observar el comportamiento de las últimas 8 horas y el segundo proporciona lecturas continuas cada 5 minutos durante el transcurso del día (Hernández, 2016).

3.2 Simulador UVA/PADOVA

Según Tegara (2021): UVA/PADOVA T1DMS es un simulador informático del sistema metabólico humano basado en un modelo dietético de cinética glucosa-insulina, como alternativa a los estudios en animales en ensayos preclínicos de nuevas estrategias de tratamiento para la diabetes tipo 1. Se puede manejar dentro del mismo entradas de simulación básica definida por el usuario como perfiles de ingestas (cantidad, horario y duración), cantidad de insulina (momento de la dosis de insulina basal/bolo), tiempo de simulación (duración y hora del día), entre otros. Los resultados de las pruebas metabólicas pueden simularse para pacientes de forma individual, así mismo muestra la simulación de una población de 10 niños, 10 adolescentes y 10 adultos.

3.3 Base de Datos

Según Elmasri y Navathe (2019), una base de datos se define como una colección de datos interrelacionados, diseñados para ser almacenados en un formato estructurado, que puede ser consultado, actualizado y manipulado eficientemente por usuarios y aplicaciones. Las bases de datos se utilizan en una variedad de campos, incluyendo la empresa, la educación, la salud y la ciencia, y su diseño y gestión es esencial para garantizar la integridad, seguridad y disponibilidad de los datos. Además, las bases de datos pueden ser utilizadas para generar informes, realizar análisis de datos y respaldar la toma de decisiones en una organización (p. 3-4).

3.4 Algoritmos de aprendizaje supervisado

Machine learning (ML) es un campo de la inteligencia artificial (IA) que permite a las computadoras por medio de sus algoritmos aprender y desarrollar tareas de predicción,

clasificación e identificación. Los algoritmos de aprendizaje supervisado constituyen una rama de ML, construye un modelo bajo supervisión con la ayuda de un conjunto de datos para proporcionar la entrada y salida que se desea (Saravanan y Sujatha, 2018). Dentro de sus funciones, Suthaharan (2016) refiere que el aprendizaje supervisado ayuda a que el algoritmo desarrolle bien su tarea a partir de la búsqueda de parámetros adecuados que se ajusten de mejor manera al modelo. Describe a su vez dos aspectos importantes, inicialmente definir las tres etapas para su diseño, etapa de entrenamiento, etapa de validación y etapa de prueba, también seleccionar oportunamente las métricas de evaluación de desempeño para validar, entrenar y probar dichos modelos (p. 183-206).

La clasificación y predicción en los algoritmos de aprendizaje supervisado son métodos importantes en el análisis de datos. Hodeghatta y Nayak (2017) menciona que “la clasificación predice la clase categórica (o valores discretos), mientras que la regresión y otros modelos predicen funciones de valores continuos” (p.131). Dentro de los clasificadores se encuentran los árboles de decisión, K vecinos más cercanos (K-NN), máquinas de soporte vectorial, clasificador Naive Bayes y redes neuronales (Hodeghatta y Nayak, 2017, p.132-150).

3.5 Redes neuronales

Según (Atria Innovación, 2019) “son un modelo inspirado en el funcionamiento del cerebro humano, está formado por un conjunto de nodos conocidos como neuronas que están conectadas y trasmiten señales entre sí” (párr.1). El objetivo principal de este modelo matemático es aprender modificándose automáticamente a sí mismo, de forma que puede llegar a realizar tareas complejas como predicción y clasificación.

3.5.1 Unidad Recurrente Cerrada (GRU)

La unidad recurrente cerrada (GRU) tiene como finalidad resolver el problema de gradiente de fuga (dificultad en el entrenamiento de redes neuronales con métodos de aprendizaje basados en gradiente) que viene con una red neuronal recurrente estándar (Kostadinov, 2017). GRU emplea elementos importantes dentro de su estructura (figura 1), puerta de actualización (Z_t), puerta de reinicio (r_t) y un estado oculto candidato. La puerta de actualización controla cuanto influye el estado oculto actual y siguiente en la salida, mientras que la puerta de reinicio identifica que información es irrelevante y la elimina de la red (Malingan, 2023).

Estas puertas se hallan a partir de la multiplicación de la unidad oculta anterior por sus pesos respectivos y añadiendo ese valor al producto de la entrada con sus propios parámetros de peso (Ecuación 1 y 2), la puerta de actualización y reinicio realizan el mismo procedimiento, pero difieren en los parámetros de peso por lo cual se va a proporcionar una salida sigmoidea diferente.

Una vez reestablecido un estado oculto anterior las salidas se unen a nuevas entradas (X_t) multiplicando por sus propios pesos y agregando sesgos antes de pasar a la función de activación tanh (Ecuación 3), posteriormente el candidato al estado oculto se multiplica por los resultados de una puerta de actualización y se suma a h_{t-1} previamente modificado para formar el nuevo estado oculto h_t como se ve reflejado en la ecuación 4 (Kostadinov, 2017).

$$Z_t = \sigma(W_z X_t + U_z h_{t-1}) \qquad \text{Ecuación 1}$$

$$r_t = \sigma(W_r X_t + U_r h_{t-1}) \qquad \text{Ecuación 2}$$

$$\tilde{h}_t = tanh(W X_t + r_t * U h_{t-1}) \qquad \text{Ecuación 3}$$

$$h_t = \left(Z_t h_{t-1} + (1 - Z_t)\tilde{h}_t\right) \qquad \text{Ecuación 4}$$

Ecuación 1 (Puerta de actualización - Z_t):

Z_t: *Es la puerta de actualización en el instante de tiempo t. Controla cuánto de la información del estado oculto anterior (h_(t-1)) se debe mezclar con el estado oculto candidato (h̃_t) para obtener el nuevo estado oculto (h_t).*

σ: *Función de activación sigmoide que comprime los valores en el rango (0, 1).*

W_z: *Matriz de pesos que se aplica a la entrada actual (X_t).*

U_z: *Matriz de pesos que se aplica al estado oculto anterior (h_(t-1)).*

Ecuación 2 (Puerta de reinicio - r_t):

r_t: *Es la puerta de reinicio en el instante de tiempo t. Controla qué información se debe olvidar del estado oculto anterior (h_(t-1)) antes de calcular el estado oculto candidato (h̃_t).*

σ: *Función de activación sigmoide que comprime los valores en el rango (0, 1).*

W_r: *Matriz de pesos que se aplica a la entrada actual (X_t).*

U_r: *Matriz de pesos que se aplica al estado oculto anterior (h_(t-1)).*

Ecuación 3 (Candidato al estado oculto - h̃_t):

h̃_t: *Es el estado oculto candidato en el instante de tiempo t. Es una versión actualizada del estado oculto anterior, que luego será combinado con la puerta de actualización para formar el nuevo estado oculto (h_t).*

tanh: *Función de activación tangente hiperbólica que comprime los valores en el rango (-1, 1).*

W: *Matriz de pesos que se aplica a la entrada actual (X_t).*

r_t: *Puerta de reinicio que controla la importancia del estado oculto anterior (h_(t-1)).*

Ecuación 4 (Nuevo estado oculto - h_t):

h_t*: Es el nuevo estado oculto en el instante de tiempo t. Es una combinación de la puerta de actualización (Z_t) multiplicada por el estado oculto anterior (h_(t-1)), y la puerta (1 - Z_t) multiplicada por el estado oculto candidato (h̃_t).*

Z_t: *Puerta de actualización que controla la importancia del estado oculto anterior (h_(t-1)).*

h_(t-1): *Estado oculto anterior.*

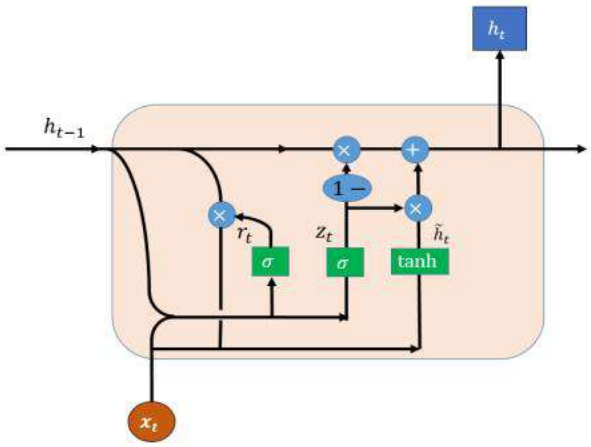
(1 - Z_t): *Complemento de la puerta de actualización.*

h̃_t: *Estado oculto candidato.*

En resumen, una unidad GRU utiliza las puertas de actualización y reinicio para controlar la información que fluye a través del estado oculto en función de la entrada actual y el estado oculto anterior. Estas puertas permiten mitigar el problema de gradiente de fuga y mejorar el entrenamiento de redes neuronales recurrentes.

Figura 3

Arquitectura de la GRU



Nota. Adaptado de "Unidad recurrente cerrada convolucional: red neuronal recurrente para la estimación del estado de carga de las baterías de iones de litio", por Huang et al., 2019 (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8759027). CC

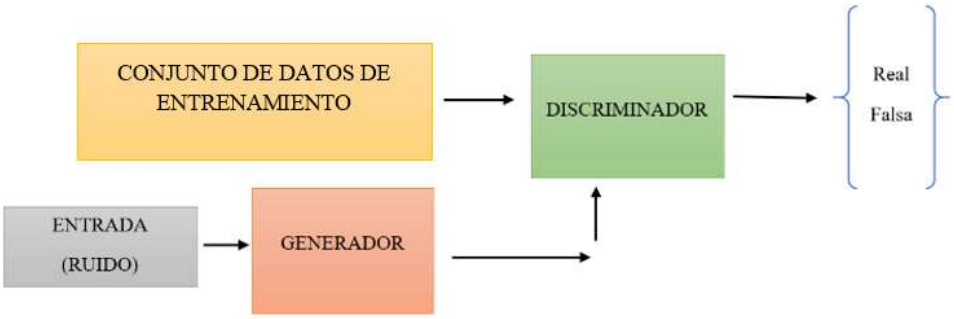
3.5.2 Red Generativa Antagónica (GAN)

La arquitectura GAN es un tipo de modelo de aprendizaje profundo que está compuesta por dos redes neuronales, un generador y un discriminador. El discriminador toma una muestra ya sea sintética o real en su entrada y produce una salida binaria que indica si la muestra es real o falsa, el generador como su nombre lo indica se encarga de generar muestras sintéticas las cuales se obtienen del mapeo de una distribución de ruido latente en un espacio de datos. Durante la etapa de entrenamiento las redes se entrenan conjuntamente en un proceso adversarial, el generador intenta engañar al discriminador con muestras sintéticas las cuales el discriminador no puede distinguir de las muestras reales, en esta medida durante el entrenamiento, el generador va

aprendiendo a producir muestras sintéticas más realistas y el discriminador se vuelve eficiente a la hora de identificar muestras que no son reales (Goodfellow et al., 2014).

Figura 4

Arquitectura de la GAN



Nota. Se ilustra la arquitectura de una GAN. Elaboración propia

El entrenamiento contiene dos funciones de perdida, una para el generador y otra para el discriminador (Zhu et al., s.f).

$$L_G = \lambda_1 L_{SL} + \lambda_2 m \sum_{i=1}^{m} log\left(1 - f_D\left(\hat{y}^{(i)}\right)\right) \quad \text{Ecuación 5}$$

$$L_D = \frac{1}{m}\sum_{i=1}^{m}\left[-log f_D\left(\hat{y}^{(i)}\right) - \left(log\left(1 - f_D\left(\hat{y}^{(i)}\right)\right)\right)\right] \quad \text{Ecuación 6}$$

$$L_{SL} = \sum_{i=1}^{m}\left(G_{t+w}{}^{(i)} + \hat{G}_{t+w}{}^{(i)}\right)^2 \quad \text{Ecuación 7}$$

Donde f_D representa el cálculo en el discriminador, L_{SL} la perdida de error cuadrático medio en el aprendizaje supervisado, λ_1 y λ_2 se emplean para ajustar la perdida supervisada y la contradictoria y finalmente m representa el tamaño del lote.

3.6 Máquina de Soporte Vectorial (SVM)

Las Máquinas de Soporte Vectorial (SVM, por sus siglas en inglés) son un tipo de algoritmo de aprendizaje supervisado utilizado para la clasificación y regresión. El objetivo de un SVM es encontrar un hiperplano en un espacio de alta dimensión que separe dos clases de datos de entrada, maximizando la distancia entre ellos (Suykens y Vandewalle, 1999).

En otras palabras, las SVM son capaces de aprender a clasificar o predecir valores de una variable de interés, a partir de un conjunto de datos previamente etiquetados, encontrando la mejor frontera de separación entre las clases. Además, las SVM son capaces de manejar tanto datos linealmente separables como datos no linealmente separables, gracias al uso de una función de kernel que transforma los datos de entrada en un espacio de características de mayor dimensión, donde sí pueden ser separados linealmente (Cortes y Vapnik, 1995).

Las SVM son ampliamente utilizadas en la clasificación de imágenes, reconocimiento de voz, detección de spam, detección de fraude, análisis de sentimientos, entre otros. Debido a su capacidad para trabajar con datos de alta dimensionalidad, las SVM son una herramienta útil para muchas aplicaciones en el campo de la ciencia de datos y el aprendizaje automático.

3.7 Parámetros de evaluación

Este apartado se describe las métricas de evaluación utilizadas en el aprendizaje automático Mughees *et al.* (2023)

3.7.1 Error Cuadrático Medio (RMSE)

Determina el error entre el valor predicho y el real, la raíz cuadrada se emplea para medir la magnitud del error con las mismas unidades que el valor real. Cuando RMSE es cero, se puede decir que el modelo se ajusta a los datos y mejora significativamente las predicciones. Su expresión matemática está dada por la siguiente ecuación:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y_i}\right)^2} \qquad \text{Ecuación 8}$$

$$Y_i : Es\ el\ valor\ real;\ \widehat{Y_i}: Es\ el\ valor\ predicho;\ n: número\ de\ muestras$$

3.7.2 Error Absoluto Medio (MAE)

Se halla promediando la diferencia absoluta entre el valor predicho y real sobre todo el conjunto de medición, todos los errores se ponderan en la misma escala. Provee una medida uniforme de como el modelo está funcionando. Su expresión matemática está dada por la siguiente ecuación:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \widehat{Y_i}\right| \qquad \text{Ecuación 9}$$

$$Y_i : Es\ el\ valor\ real;\ \widehat{Y_i}: Es\ el\ valor\ predicho;\ n: número\ de\ muestras$$

3.8 Parámetros de los modelos empleados

Dentro de los algoritmos de las redes neuronales se pueden encontrar diferentes parámetros, los cuales se deben ir ajustando en base a la necesidad que se tenga, algunos de ellos son:

3.8.1 Batch Size

Este es el número de muestras que se le introduce a la red para que entrene. Si el número es un valor pequeño significa que la red posee poca cantidad de datos y ayuda a que el entrenamiento se realice de manera más rápida, pero en consecuencia se puede perder características que pueden ser importantes. Por otro lado, cuando el número de muestras resulta ser más grande, la red entrena de manera más lenta, pero resulta más eficiente ya que tiene en cuenta casos relevantes para que la red logre aprender (Gonzales, 2020).

3.8.2 Unidades

Las unidades o número de neuronas dentro de una arquitectura de red neuronal representan el número de nodos de la capa, como no se puede deducir que tan relevante resulta el valor de este parámetro para realizar la predicción, se requiere de más tiempo de cómputo para realizar las diferentes pruebas que ayuden a determinar la cantidad que unidades con la cual se pueda obtener una predicción adecuada (Gonzales, 2020).

3.8.3 Dropout

Es un método que se utiliza para evitar un overfitting (Sobreajuste o sobre entrenamiento) en las redes neuronales, para ello desactiva un numero de neuronas de forma aleatoria en cada capa oculta, obligando a las neuronas a trabajar de mejor manera por si solas (Srivastava *et al*., 2014).

3.8.4 Función de activación

Es una función matemática que establece si una entrada en particular debe ser utilizada o por el contrario no. Son importantes en el diseño de redes neuronales por que introducen la no linealidad (Kumar, 2022). Las funciones de activación más comunes que podemos encontrar en las redes neuronales son la función tanh, sigmoide y ReLU.

La función Tanh, es una función no lineal en la cual un valor de entrada es reducido en el rango de –1 y 1, es diferenciable y continua. Presenta una simetría respecto al cero, genera valores cercanos a –1 o 1 cuando la entrada es de gran magnitud (negativa o positiva). Por otro lado, la función sigmoide o también denominada función de aplastamiento, comprime los valores de entrada entre 0 y 1, es diferenciable lo cual es óptima para el entrenamiento de redes neuronales, suele emplearse en tareas de clasificación. Finalmente, la Unidad Lineal Rectificada (ReLU) es un tipo de función que devuelve la entrada si es positiva y cero si es negativa, esta función suele emplearse en las capas intermedias de las redes ayudando a prevenir el problema del desvanecimiento del gradiente, no es diferenciable en cero lo que resulta ineficiente para algoritmos de optimización (Kumar, 2022).

3.8.5 Optimizadores

En el aprendizaje profundo son funciones matemáticas que se emplean para intentar reducir la función perdida o lo que se conoce como error. Para ello realiza un ajuste en los parámetros de la red como los pesos y tasa de aprendizaje con el fin de lograr un buen funcionamiento y resultados más precisos en el modelo (Chauhan, 2020).

- **Descenso de Gradiente Estocástico (SGD):** Se caracteriza por utilizar solo una muestra de todo el conjunto de datos para realizar cada iteración, como resultado de esto se obtiene un optimizador más rápido, eficiente en memoria y con la capacidad de converger a un mínimo global escapando de los mínimos locales. Dentro de sus desventajas SGD tarda en converger y presenta actualizaciones ruidosas (ML, 2019).
- **Descenso de Gradiente Adaptable (AdaGrad):** La principal función es tratar de mantener una tasa de aprendizaje que se acomode para cada lote en cada paso del tiempo, utiliza actualizaciones pequeñas para ciertos atributos asociados a características que suceden con mayor frecuencia, este optimizador cambia la tasa de aprendizaje en la medida que se acomode en cada iteración, puede realizar el entrenamiento de una red con pocos datos. No obstante, presenta algunas desventajas ya que a nivel de cómputo es costoso y tiene una convergencia lenta (Doshi, 2019).
- **Propagación Cuadrática Media (RMSprop):** Es una versión especial de AdaGrad, donde la tasa de aprendizaje se ajusta de manera automática y se selecciona de manera diferente para cada parámetro (Chauhan, 2020).
- **Estimación del Momento Adaptativo (Adam):** Es uno de los optimizadores más utilizados en redes neuronales, es rápido y eficiente en su entrenamiento cuando emplea conjuntos de datos muy grandes. Es un optimizador que funciona de mejor manera que SGD cuando se trata de modelos complejos (Khan, 2022). Combina ciertas características de RMSprop y AdaGrad, lo cual hace que tenga un mejor rendimiento. Al igual que otros optimizadores calcula la tasa de aprendizaje adaptativa para cada parámetro (Musstafa, 2021).

4. Algoritmos propuestos

El desarrollo del modelo de predicción se hace en 4 pasos generales. El primero consiste en generar el conjunto de mediciones usando el simulador UVA/PADOVA y hacer un preprocesamiento con los datos generados para adecuar la base datos con los pacientes que se van a usar para el entrenamiento de los modelos. Segundo, entrenar dos redes neuronales que predigan el comportamiento de la glucosa de los pacientes de acuerdo con la ingesta y comparar los resultados de cada algoritmo. Tercero, predecir del valor de la ingesta de carbohidratos con la red que mejor desempeño obtuvo usando un algoritmo de máquina de soporte vectorial. Cuarto, mostrar y comparar los resultados obtenidos.

4.1 Generación del conjunto de mediciones

4.1.1 Generación de los datos

Para la generación del conjunto de mediciones se usa el simulador UVA/PADOVA obteniendo datos de insulina, glucosa e ingesta de carbohidratos para 10 pacientes adultos. Así como el organismo de las personas reaccionan diferente a la ingesta, cada paciente generado por el simulador tiene un comportamiento de glucosa diferente, por tanto, es necesario hacer un modelo para cada paciente.

En cuanto a la generación de la base de datos, se debe tener en cuenta las variables de entrada y salida del simulador. En este caso, las variables de entrada que se modificaron son: Ingesta de carbohidratos e insulina de acción rápida. Por otro lado, la salida del sistema es la curva del comportamiento de la glucosa en sangre como respuesta a estas dos entradas. Para la entrada

de ingesta de carbohidratos se simula mediante un impulso de 15 minutos (15 muestras), 3 veces al día (simulando 3 ingestas diarias) por tres semanas. Dichas ingestas están relacionadas con el consumo promedio de carbohidratos por comida para un adulto en un rango de 30 y 100 g por ingesta, distribuidas de manera escalonada para tener una base de casos lo más completa posible. Cada ingesta de carbohidratos está directamente relacionada a una dosis de insulina ideal, encargada de estabilizar la glucosa en sangre dentro de los niveles recomendados. Finalmente se ejecuta el modelo y se obtienen los datos de entrada y salida del simulador UVA/PADOVA, donde se extraen 4 secuencias de variables: Tiempo, Ingesta de carbohidratos, Insulina inyectada y glucosa en sangre. Estas muestras se generan cada 1 minuto, lo que corresponde a secuencias de 20160 muestras para las 3 semanas.

4.1.2 Preprocesamiento

Una vez generados los datos en Simulink, se exportan en formato .xls obteniendo un numero mucho mayor de datos por lo que se hace necesario hacer un filtrado de los datos, el cual consiste en eliminar los datos que no corresponden a la línea de tiempo con espaciado de 1 minuto. Esta tarea se hace filtrando la columna de tiempo, en la que se eliminan todas las filas en las que el tiempo no es un numero entero o esta repetido, además de esto se deben eliminar los primeros 480 datos de tal manera que la primera ingesta inicie en el minuto 0 y se obtiene la cantidad exacta de 20160 datos por variable. Luego se agrega dos columnas, una de ellas contiene la información del número de ingesta a la que corresponden los datos, que varía cada 8 horas lo que corresponde a 480 datos, la otra columna es un índice de tiempo que se reinicia con cada ingesta, esto se

convierte en 42 ingestas de 480 datos cada una. Finalmente se exportan los datos en un archivo .CSV de tal manera que pueda ser leído por Python en Google Colab.

Figura 5

Dataframe en Python de la base de datos del paciente 5

	Carbo	Tiempo	Tiempo_F	Insulina	Glucosa	Ingesta
0	Ingesta1	0	0	4.545417	113.314862	2000.0
1	Ingesta1	1	1	4.545417	113.315147	2000.0
2	Ingesta1	2	2	4.545417	113.316748	2000.0
3	Ingesta1	3	3	4.545417	113.321662	2000.0
4	Ingesta1	4	4	4.545417	113.332735	2000.0
...	...	...	...	...	...	...
20155	Ingesta42	475	20155	1.545417	109.695502	0.0
20156	Ingesta42	476	20156	1.545417	109.681308	0.0
20157	Ingesta42	477	20157	1.545417	109.667465	0.0
20158	Ingesta42	478	20158	1.545417	109.653969	0.0
20159	Ingesta42	479	20159	1.545417	109.640816	0.0

20160 rows × 6 columns

Nota. La figura muestra como están distribuidos los datos del paciente 5. Los datos están organizados por columnas, en cada columna hay 20160 datos de 7 características diferentes como glucosa, insulina, ingesta, etc.

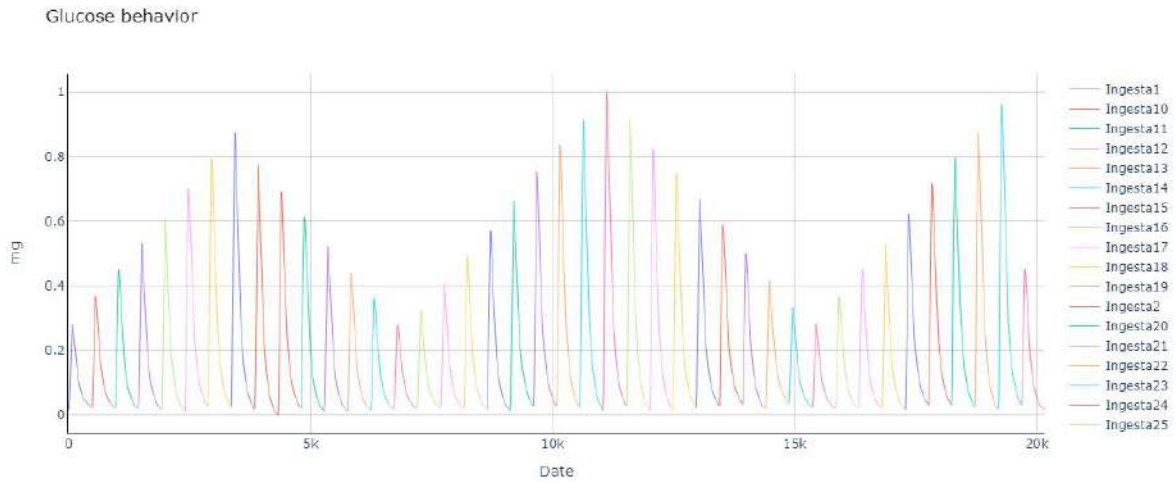
Una vez leída correctamente la base de datos, se reorganiza la información creando un DataFrame para cada una de las variables: glucosa, insulina e ingesta de carbohidratos. Cada uno de estos DataFrames contiene 42 filas que corresponden al total de las ingestas y 480 columnas que

corresponden a la medida en cada una de las muestras tomadas. Después de esto se hace un proceso de normalización, con el fin de que las dos variables glucosa e insulina estén dentro del mismo rango, en este caso entre 0 y 1, la finalidad es lograr que las dos variables tengan el mismo peso cuando pasen por el modelo de red neuronal.

En la figura 6 se presenta la gráfica del comportamiento de la glucosa en un paciente específico (paciente 5), donde se evidencia la respuesta ante 42 ingestas, además cada una de ellas se ve representada por 480 muestras, A lo largo del eje vertical se puede diferenciar cada ingesta por el cambio de color en la gráfica.

Figura 6

Comportamiento de la glucosa en las 3 semanas del paciente 5

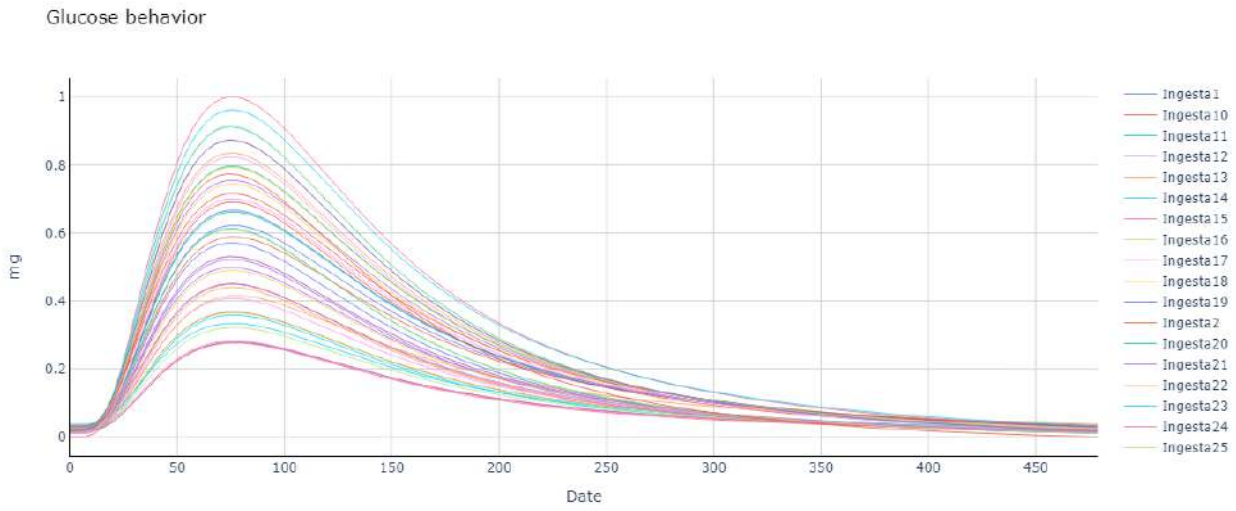


Nota. En la imagen se muestra el comportamiento de la glucosa del paciente 5 en un tiempo de 3 semanas.

En la figura 7, se grafica la misma información que en la figura 6, con la diferencia que en este caso se separan las ingestas para poder observar el comportamiento y los niveles alcanzados por cada una de ellas de manera mas detallada.

Figura 7

Comportamiento de la glucosa separada por ingestas para el paciente 5



Nota. Se muestra el comportamiento de la glucosa debido a cada una de las 42 ingestas.

4.2 Modelo GRU

Inicialmente se importa la base de datos y se lee como *Dataframe.* El primer paso es la normalización de las variables ya que es indispensable para equilibrar la relevancia de cada variable de entrada en el entrenamiento de la red neuronal.

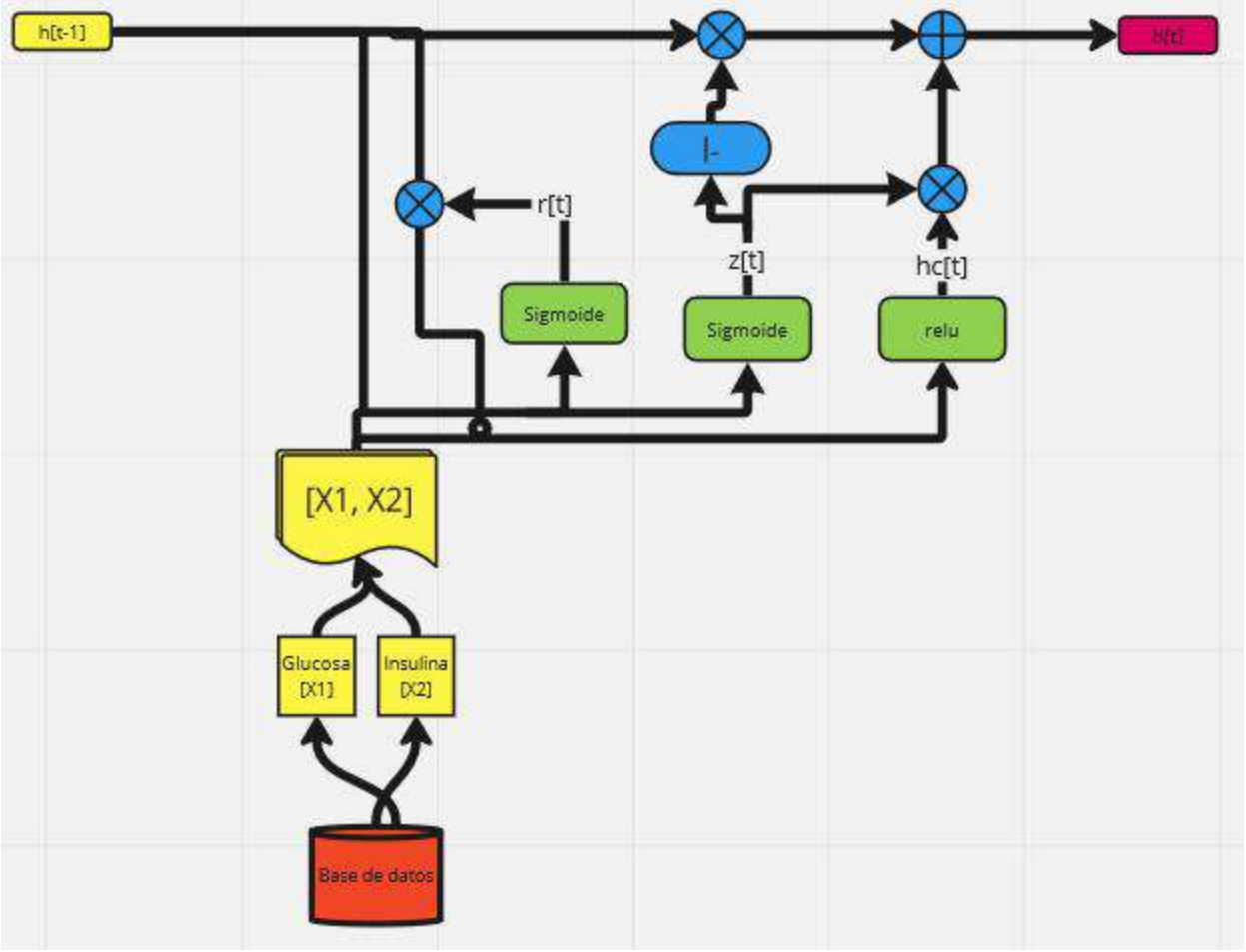
Respecto al modelo se definió un modelo base que consta de una capa de entrada y una de salida donde las unidades de cada una de ellas corresponden directamente a la dimensión del vector de entrada y salida, además se agrega una capa intermedia con una cantidad de unidades, variable definida inicialmente en 256. Se agregan algunos parámetros como Dropout, activación, activación recurrente. Al compilar el modelo se agrega un optimizador y las métricas de error MSE, RMSE

y MAE. Por último, se define un Callback con el fin de evitar un sobre ajuste en la red, usando como monitor el MAE.

En la siguiente figura se muestra un diagrama de bloque general de la red.

Figura 8

Diagrama de bloques del modelo



Nota. Se muestra el comportamiento de una unidad de toda la red GRU

Donde:

X1, X2: Vector de glucosa e insulina

Sigmoide: Activación

relu: Activación recurrente

h(t): Vector de salida

hc(t): Candidato a vector de actualización

r(t): Vector de reinicio de puerta

z(t): Vector de actualización de puerta

4.2.1 Separación de los datos para entrenamiento, test y validación

El conjunto de datos restante se divide en un 70% para entrenamiento y un 30% para validación.

4.2.2 Selección de los parámetros

Para seleccionar los parámetros se hacen varias pruebas donde se entrena el modelo con el mismo paciente (misma base de datos), se definen algunos parámetros iniciales y luego se entrena el modelo modificando cada uno de los parámetros: unidades, Batch_size, Dropout y activaciones uno a uno en rangos específicos. Se busca obtener la mejor configuración minimizando el error en la predicción, para esto se obtiene el error para cada entrenamiento y se compara gráficamente pasa así determinar la mejor alternativa. A continuación, se detalla el proceso de búsqueda para cada parámetro.

4.2.2.1 Unidades o Neuronas. Se entrena el modelo modificando únicamente las unidades de la capa densa, con valores desde 8 hasta 256 (en potencias de 2), donde se registran los datos del error para cada entrenamiento, se analiza numéricamente y se grafican las métricas de error como MAE, RMSE y MSE con el fin de observar el comportamiento de cada uno de ellos, se grafica de manera independiente el RMSE solo en su rango para que sea más perceptible el resultado. A continuación, se presenta la tabla 1 donde se consigna el error para cada entrenamiento, las gráficas mencionadas y una gráfica donde se compara el perfil de la predicción con la real.

Tabla 1

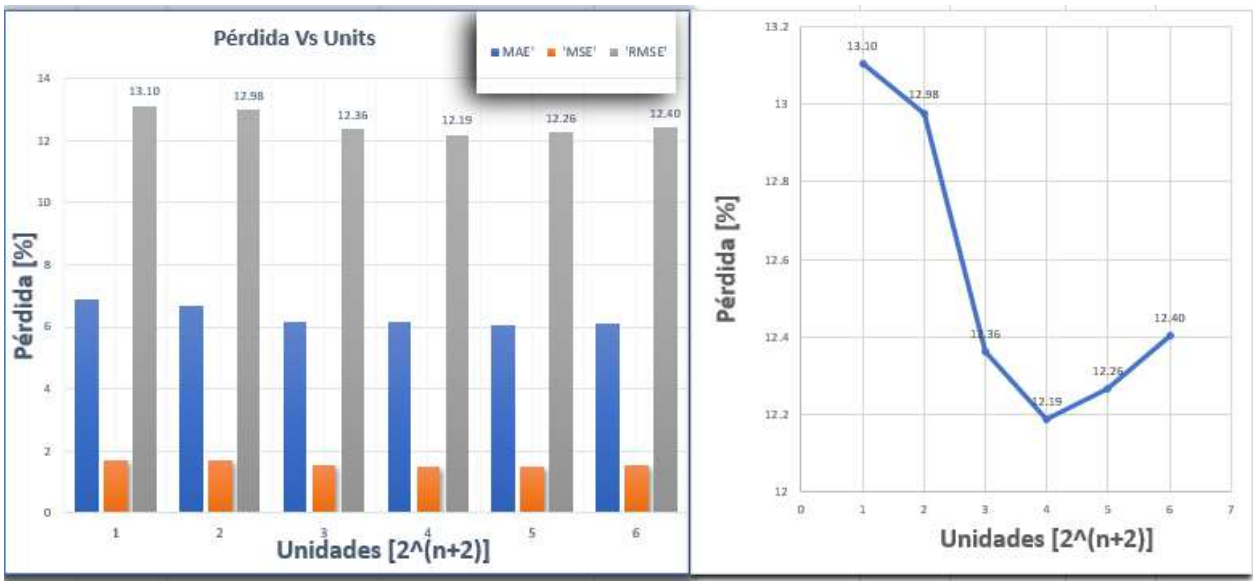
Error respecto a la variación de unidades

Error respecto a la variación de las Unidades. [%]				
Parámetros fijos				
Dropout	Épocas	Parámetros	Folders	Batch_size
0.2	5	5325	5	16
Parámetros variables				
		MAE	MSE	RMSE
Unidades	8	6.8632	1.7193	13.1030
	16	6.6650	1.6864	12.9772
	32	6.1579	1.5293	12.3607
	64	6.1618	1.4859	12.1886
	128	6.0692	1.5051	12.2650
	256	6.1053	1.5391	12.4035

Nota. Se muestra el error obtenido al variar las unidades o neuronas del modelo. Se muestra los distintos errores obtenidos al variar desde 8 hasta 256.

Figura 9

Relación de la pérdida respecto al número de unidades

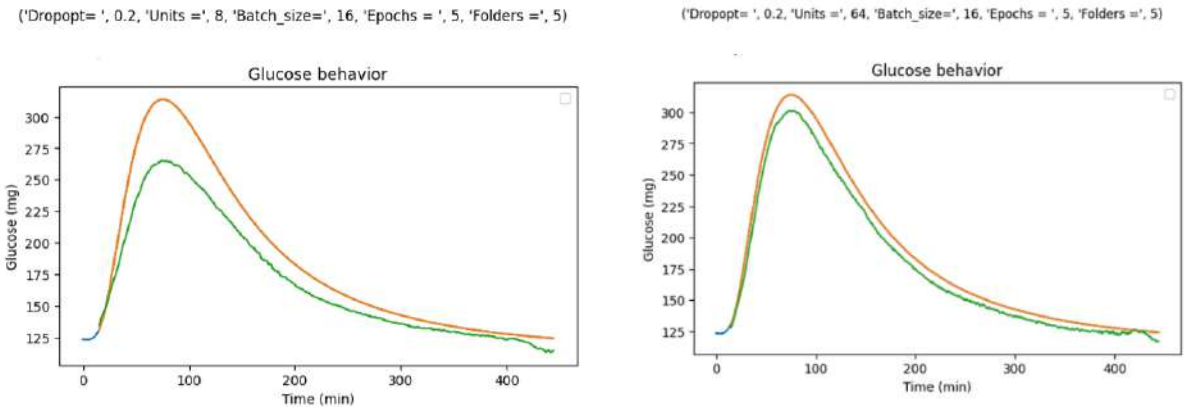


Nota. Se muestra la representación gráfica del error por número de unidades. En la figura de la izquierda se muestra la representación por barras de las 3 medidas de error y en la derecha solo la gráfica del RMSE para facilitar su visualización.

En la figura 10, se presenta gráficamente la predicción para un paciente en específico, donde se puede comprobar de manera visual la diferencia entre las predicciones con 8 y 64 unidades, esto se puede comprobar tanto en la tabla 1, como en la gráfica de la figura 9. De esta manera se selecciona la mejor de las configuraciones para el modelo, este proceso se repite modificando y comparando los resultados de los de más parámetros.

Figura 10

Predicciones de glucosa para el paciente 5



Nota. Se muestran las predicciones de glucosa obtenida con el modelo para el paciente 5.

4.2.2.2 Batch Size. Luego de establecer las unidades de la capa densa en 64, se continua con la variación del *Batch size* con el mismo rango entre 8 y 256, después de entrenar la red en diferentes ocasiones con el mismo paciente se condensan y grafican los datos a continuación.

Tabla 2

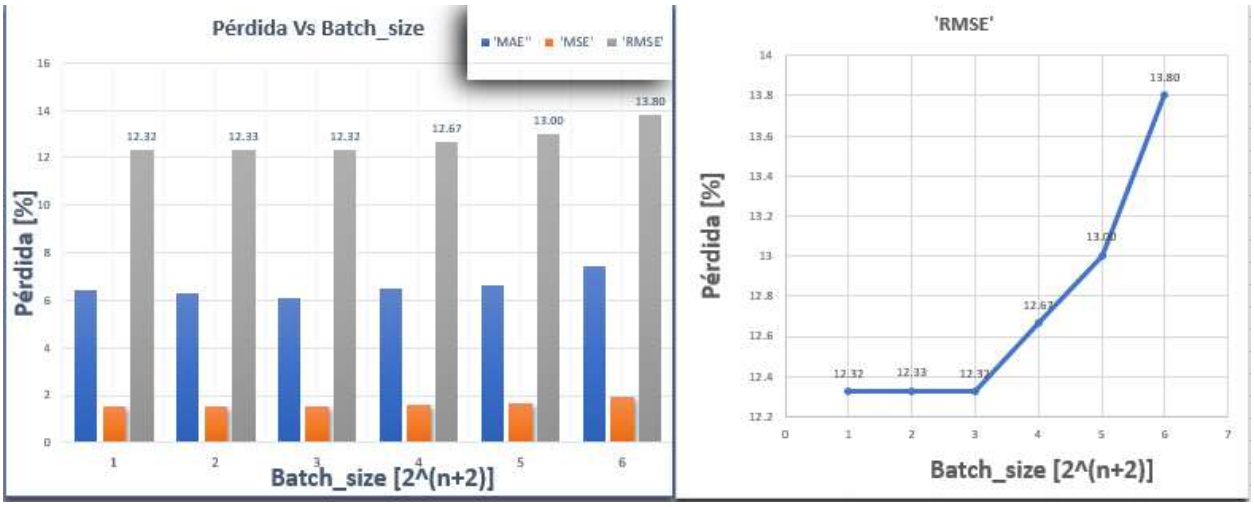
Error respecto a la variación del batch size

Error respecto a la variación de las Unidades. [%]				
Parámetros fijos				
Dropout	**Épocas**	**Parámetros**	**Folders**	**Unidades**
0.2	**5**	**5325**	**5**	**64**
Parámetros variables				
		MAE	**MSE**	**RMSE**
Batch_size	8	6.4255	1.5196	12.3235
	16	6.2797	1.5205	12.3292
	32	6.1178	1.5210	12.3239
	64	6.4717	1.6071	12.6656
	128	6.6467	1.6965	13.0045
	256	7.4604	1.9162	13.8019

Nota. Se muestra el error obtenido al variar el tamaño del *Batch size* del modelo. Se evidencian los distintos errores obtenidos al variar desde 8 hasta 256.

Figura 11

Relación de la perdida respecto al Batch size



Nota. Se muestra la representación gráfica del error por tamaño del *batch size*. En la figura de la izquierda se muestra la representación por barras de las 3 medidas de error y en la derecha solo la gráfica del RMSE en una línea.

Como se observa en la gráfica anterior se puede determinar que las tres primeras iteraciones arrojan predicciones con un error similar, sin embargo, en este caso se tuvo en cuenta el mayor tamaño (32) del Batch Size con el propósito de disminuir los recursos del sistema requeridos por la red y el tiempo de entrenamiento de esta.

4.2.2.3 Dropout. En cuanto al *Dropout* se realizaron entrenamientos con variaciones de este que van entre 0 y 0,5 con pasos de 0,1. Los resultados se condensan en la tabla 3 y a partir de

esta se genera la gráfica que nos permite hacer la comparación de los valores de error por cada entrenamiento. Finalmente se grafica específicamente el RMSE con el fin de tener una escala más ajustada a los valores de este y así tener una mejor perspectiva del rendimiento de la red.

Tabla 3

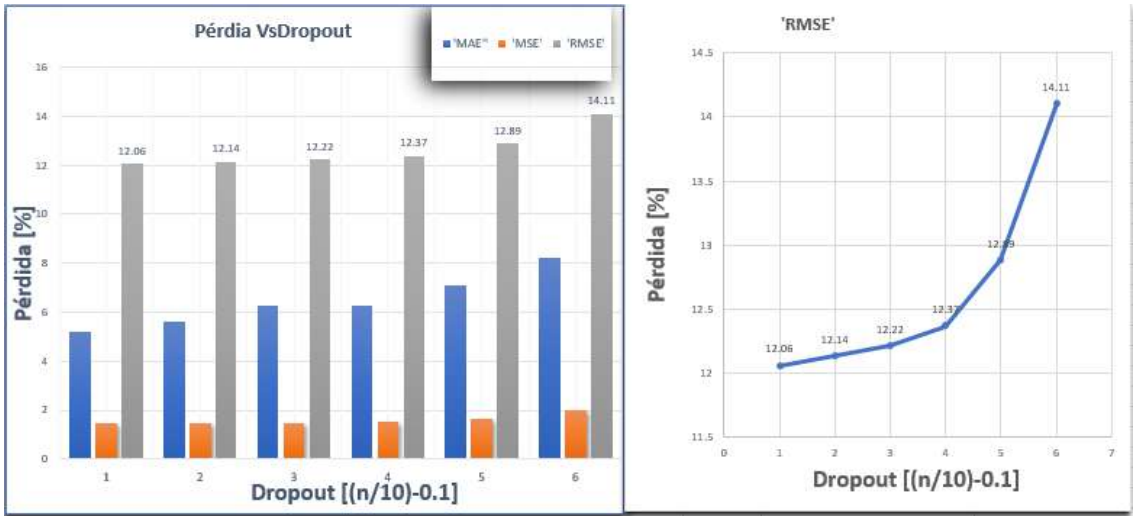
Error respecto a la variación del Dropout

Error respecto a la variación de las Unidades. [%]				
Parámetros fijos				
Batch_Size	Épocas	Parámetros	Folders	Unidades
32	5	5325	5	64
Parámetros variables				
		MAE	MSE	RMSE
Dropout	0	5.2261	1.4559	12.0629
	0.1	5.5871	1.4730	12.1359
	0.2	6.2451	1.4932	12.2185
	0.3	6.2645	1.5308	12.3695
	0.4	7.0680	1.6610	12.8866
	0.5	8.2284	1.9897	14.1056

Nota. Se muestra el error obtenido al variar el tamaño del *Dropout* del modelo desde 0 hasta 0.5.

Figura 12

Relación de la perdida respecto al Dropout



Nota. Como se observa en la figura anterior, el mejor resultado se obtiene con dropout igual a cero, por lo que se decide prescindir de este parámetro. Cabe aclarar que el ajuste o no de este influye directamente en el tiempo de entrenamiento de la red.

4.2.2.4 Activaciones. Finalmente se realizan entrenamientos de la red con los parámetros anteriormente definidos, con el fin de descartar o encontrar la mejor configuración del modelo, se usaron las diferentes activaciones como lo son: Relu, elu, Linear, Sigmoid y Tanh, de igual manera se condensa el error para cada configuración, se grafican y analizan los resultados.

Tabla 4

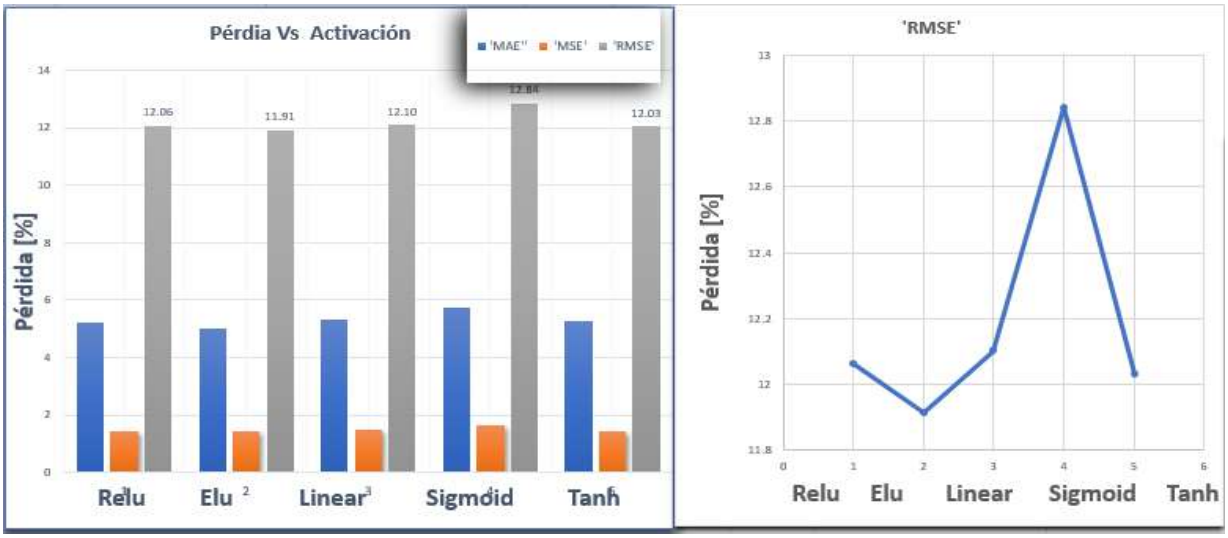
Error respecto a la variación de las activaciones

Error respecto a la variación de las Unidades. [%]					
Parámetros fijos					
Batch_Size	**Épocas**	**Parámetros**	**Folders**	**Unidades**	**Dropout**
32	**5**	**5325**	**5**	**64**	**0**
Parámetros variables					
		MAE	**MSE**	**RMSE**	**Error en área**
Activación	Relu	5.2261	1.4559	12.0629	0.0038
	Elu	5.0319	1.4195	11.9133	0.0058
	Linear	5.3436	1.4670	12.1018	0.0084
	Sigmoid	5.7438	1.6535	12.8414	0.0330
	Tanh	5.2743	1.4494	12.0319	0.0116

Nota. Se muestra el error obtenido al cambiar la activación del modelo. Se muestra los distintos errores con las activaciones Relu, Elu, Linear, Sigmoid Sy Tanh.

Figura 13

Relación de la perdida respecto a la activación

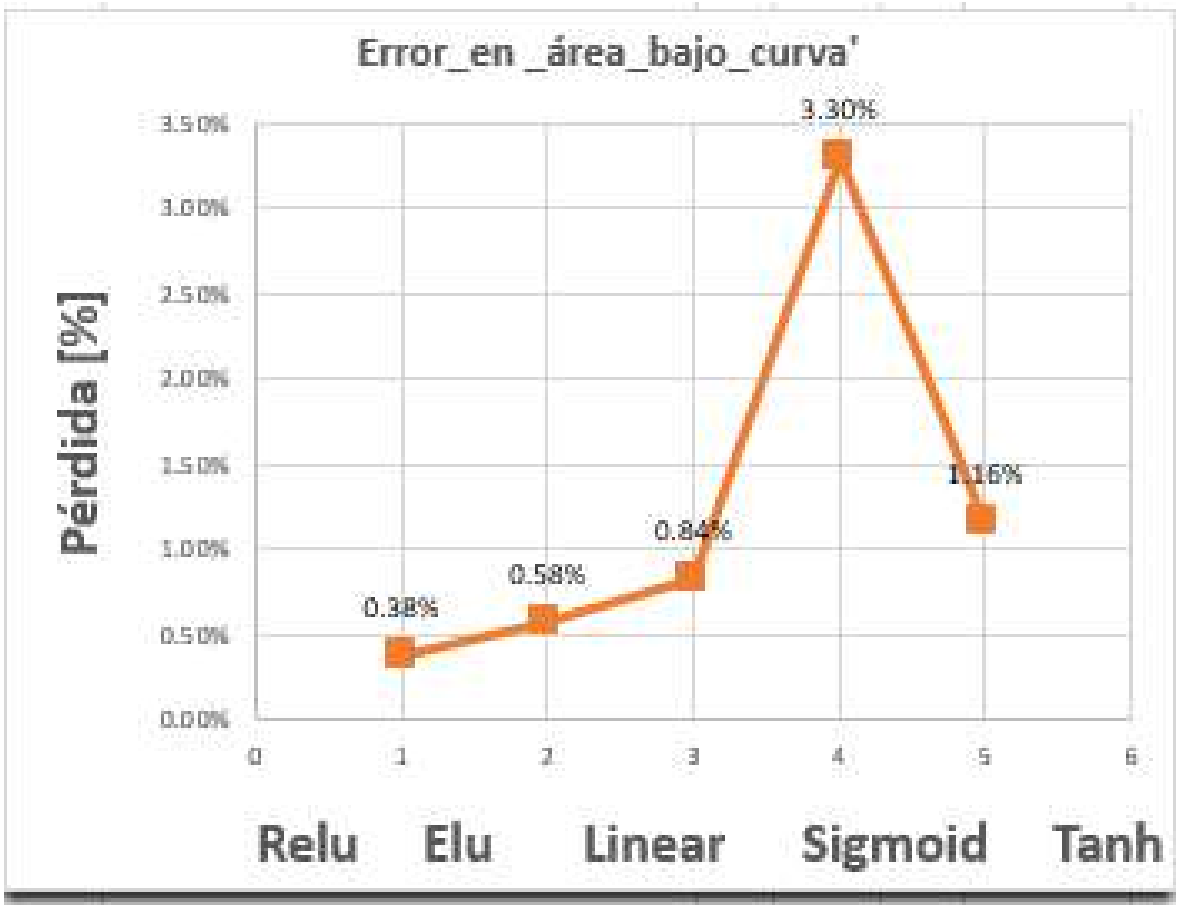


Nota. Se muestra la representación gráfica del error por activación. En la figura de la izquierda se muestra la representación por barras de las 3 medidas de error y en la derecha solo la gráfica del RMSE en una línea.

En este caso la gráfica del error no parece ser suficiente para determinar cuál de las activaciones es la adecuada para el algoritmo, ya que si se revisa directamente el error la Elu sería la primera opción. Sin embargo, en la tabla se añadió una columna donde se presenta el error comparando el área bajo la curva de la predicción respecto a la real, esta decisión se tomó porque al comparar gráficamente el perfil de la predicción no corresponde exactamente con el error, por esto se compara la integral de las predicciones, las cuales están directamente relacionadas con el tamaño de la ingesta de carbohidratos.

Figura 14

Relación del error bajo la curva de la predicción respecto a la activación



Nota. Se muestra la representación gráfica del error bajo la curva de la predicción por activación.

Se determina que la activación *Relu* presenta mejores resultados respecto a las demás. Se definen así los parámetros para el entrenamiento de la red neuronal GRU: 64 *Unidades*, tamaño del *Batch size* de 32, sin *Dropout* y activación *Relu*.

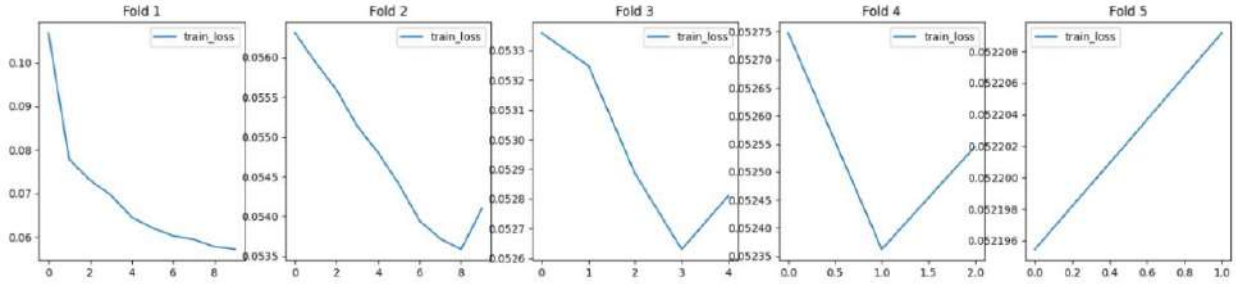
4.2.2.5 Validación Cruzada. Con la ayuda de la función *Kfold* que se encuentra en la librería ***sklearn*** se separa el conjunto de entrenamiento en 5 folders, en cada uno de ellos se entrena el modelo teniendo en cuenta que para cada uno deben ocurrir 10 épocas de entrenamiento. Estas 10 épocas no son estrictas ya que, si el error MAE no disminuye lo suficiente, es posible que el *callback* finalice el entrenamiento en el folder actual y no se completen las 10 épocas. Esto es más notorio en los folders finales donde se presentan menos de 5 épocas debido a que está muy cerca del error mínimo que puede lograr con esta configuración.

Por otro lado, se define el número de épocas en 10 por folder, de lo contrario, el modelo se sobreajusta con los datos de los dos primeros folders y para los otros 3 tan solo efectúa entre 2 y 3 épocas porque el *callback* se activa. En el otro extremo si se definen tan solo entre 3 a 5 épocas, generalmente se entrena durante todas las épocas en cada folder, pero no se logra encontrar el mejor resultado y el error logrado no es el mínimo. Así que se busca un equilibro en el entrenamiento con solo una parte de los datos y el error mínimo.

A continuación, se grafica el error MAE para cada folder de manera independiente y cada uno con una escala propia para visualizar el error al detalle por época, además de eso se puede notar el número de épocas que suceden en el entrenamiento del modelo.

Figura 15

Relaciones de la perdida respecto al número de épocas en 5 folders

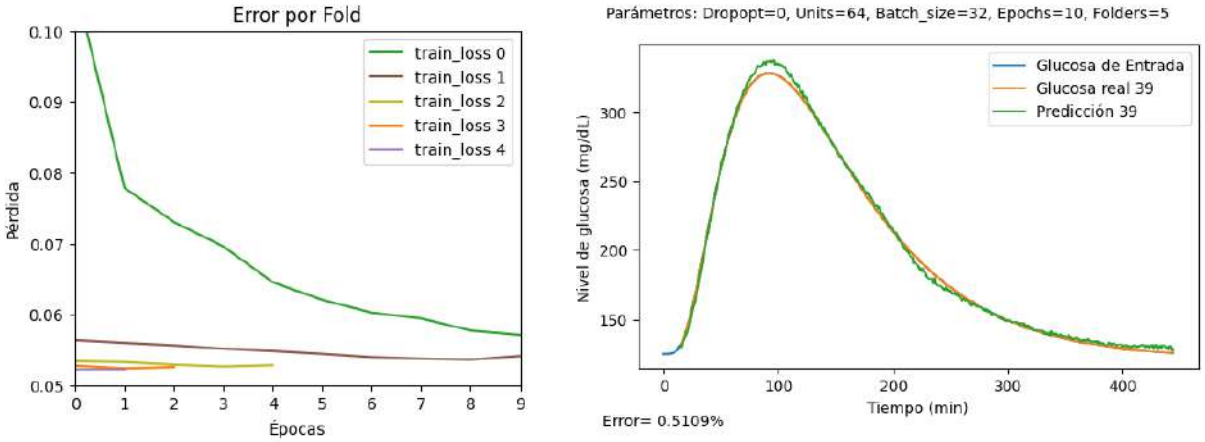


Nota. Se muestran los resultados obtenidos en 5 folders (o validaciones). En cada gráfica se representa el error obtenido respecto al número de épocas.

En la figura 15 se visualiza la misma información con la diferencia que en este caso se usa una sola escala en la gráfica, con el fin de poder observar cómo disminuye el error por cada folder. Es posible notar como la red se ajusta bastante con la primera época, lo cual es normal, en la segunda época suceden las mismas 10 épocas, pero el ajuste es mínimo. En el tercer, cuarto y quinto folder tan solo suceden entre 5 y 2 épocas, pero se alcanza un error mínimo. El número de épocas por folder puede variar por cada entrenamiento y también dependiendo de la base de datos o paciente usado, cosa que se puede observar en la figura 16. Finalmente se presenta la gráfica de la predicción lograda con la configuración de 5 folders y 10 épocas por cada uno de ellos. En la parte inferior de la predicción se agrega el porcentaje de error bajo la curva para la ingesta 39 del paciente 1.

Figura 16

Pérdida (MAE) en 5 folders y predicción obtenida del paciente 1

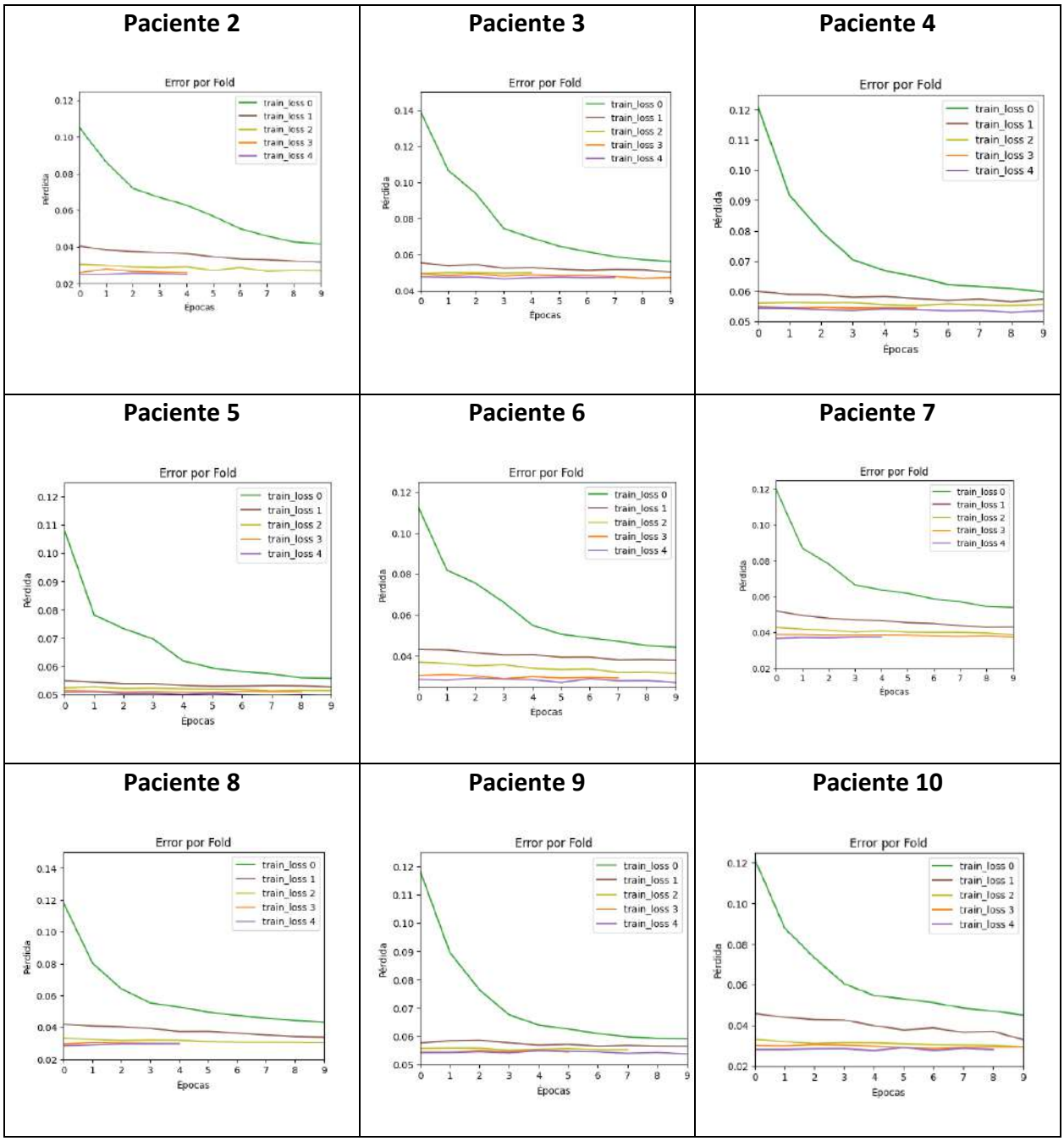


Nota. En la figura de la izquierda se muestra la perdida obtenida en los 5 folders y la figura de la izquierda se muestra la predicción obtenida para el paciente 1.

Luego de detallar como y con que parámetros se define el modelo, se continua con el entrenamiento de los 10 pacientes, donde se condensan las gráficas de error para cada paciente. En la figura 17, se puede notar de manera visual como disminuye el error y cuatas épocas se dan por folder, se presenta también la gráfica de la predicción para cada paciente, desde el 2 hasta el 9, ya que las gráficas del primer paciente se presentaron al inicio de esta sección.

Figura 17

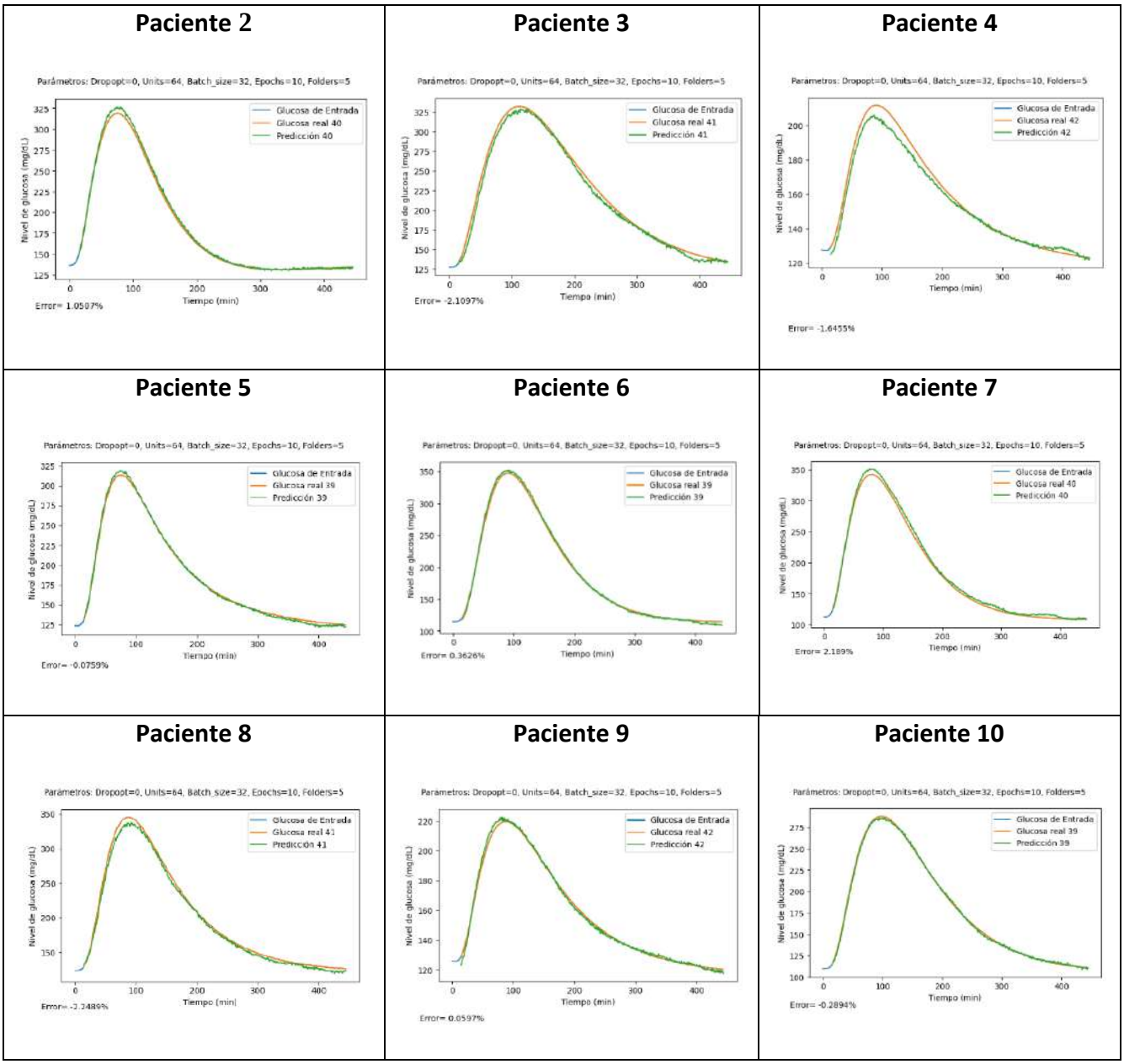
Pérdida por épocas para los 9 pacientes adultos



Nota. Cada gráfica corresponde a la perdida por época de cada paciente adulto con el modelo que se consideró óptimo.

Figura 18

Predicción de glucosa para los 9 pacientes adultos



Nota. Cada gráfica corresponde a la predicción de un paciente y a una ingesta específica. La línea azul corresponde a la glucosa de entrada al modelo, la línea naranja corresponde a la glucosa real y la línea verde corresponde a la predicción obtenida.

4.2.2.6 Área bajo la curva. El área bajo la curva de los datos predichos y los reales para el paciente 5 en específico son los siguientes:

El área bajo la predicción es 64949.547

El área bajo la curva real es: 65049.2868

El error calculado es: -0.1533 %

Se calcula como la integral de la curva desde la muestra 15 hasta la muestra 430 en la curva real, lo cual corresponde exactamente al número de muestras generados por el modelo.

El error se calcula comparando estos valores de tal manera que el signo representa si la predicción está superando o no la curva real. Además de esto al momento de mostrar los datos se presentan 4 cifras decimales.

4.3 Modelo GAN

4.3.1 Arquitectura del sistema

El código proporcionado define la arquitectura de una Red Generativa Adversarial (GAN) utilizada para generar datos sintéticos que imitan los datos reales. A continuación, se describen los bloques de código que componen la GAN y sus diferentes elementos:

1. **Crear Generador**: el bloque de código define la arquitectura del generador utilizado en la GAN. El generador toma como entrada dos secuencias de tamaño (16,2) y las concatena. Luego, pasa por seis bloques convolucionales y dos capas LSTM antes de pasar por una capa de salida. La salida tiene una forma (512,1). El modelo se compila con la función de pérdida *MAE*.

2. **Crear Discriminador**: El bloque de código define la arquitectura del discriminador utilizado en la GAN. El discriminador toma como entrada dos secuencias de tamaño (512,1) y las concatena. Luego, el discriminador pasa por seis bloques convolucionales y cinco capas densas

antes de producir una única salida (0 o 1). El modelo se compila con la función de pérdida *MAE* y el optimizador *RMSprop*.

3. **Crear GAN**: este bloque de código define la arquitectura de la GAN, esta toma tres entradas: las dos secuencias de entrada del generador y una secuencia de entrada adicional del discriminador. El generador y el discriminador se combinan para crear un modelo GAN. Primero, el generador toma las dos secuencias de entrada del generador y produce una secuencia generada. Esta secuencia generada se pasa al discriminador junto con la secuencia adicional de entrada del discriminador. El modelo GAN tiene dos salidas: la secuencia generada y la salida del discriminador. La GAN se compila con la función de pérdida *MAE* y el optimizador *RMSprop*.

Las muestras reales son los datos conocidos del comportamiento de la glucosa, es decir provienen de la base de datos. Estos datos representan la verdadera concentración de glucosa en el paciente en un intervalo de tiempo dado.

Por otro lado, las muestras falsas son aquellas que el modelo predice como valores de glucosa, pero que en realidad no han sido medidos o verificados por ningún dispositivo de medición. Estas muestras son predicciones del modelo que se basan en las características de entrada del modelo, como los datos históricos de glucosa y la ingesta de alimentos.

4.3.2 Conjunto de datos

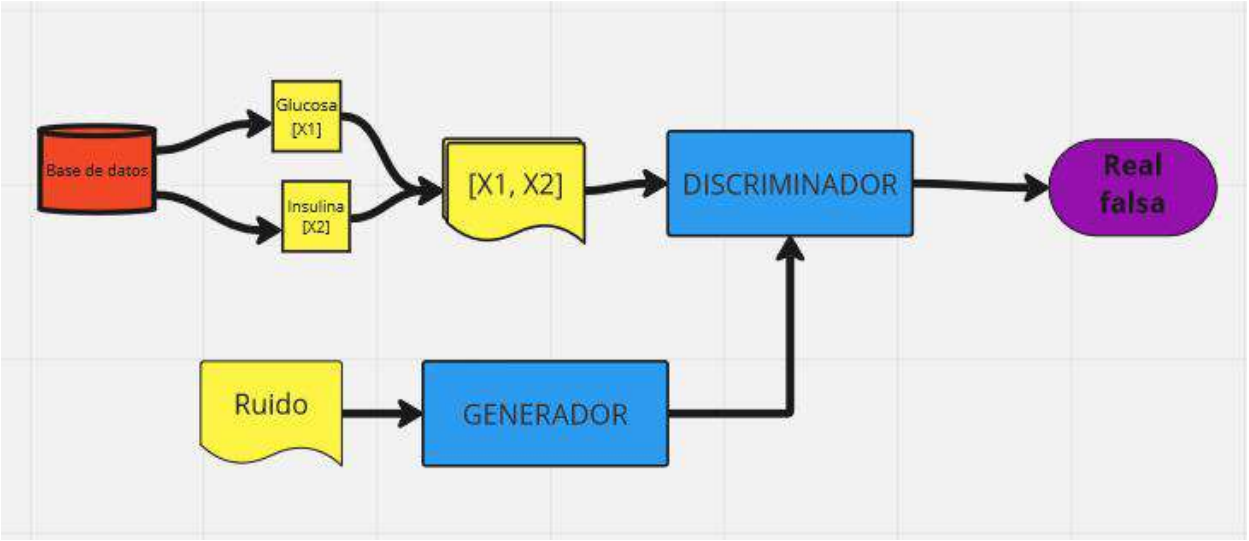
En cuanto al conjunto de datos, no es necesario extenderse demasiado ya que corresponde exactamente a las mismas bases de datos utilizadas anteriormente para el modelo GRU. El preprocesamiento es el mismo, con la única diferencia de que este modelo está diseñado para procesar 512 características por cada ingesta de carbohidratos. Por lo tanto, es necesario extender estas características en 32 muestras más. Para ello, se opta por concatenar 32 veces el valor final

en cada ingesta, considerando que la glucosa estará estable al tratarse de 8 horas después de la última ingesta. Luego, se realiza una separación de datos en dos conjuntos: entrenamiento y prueba, utilizando un 70% y 30%, respectivamente.

A continuación, se muestra el diagrama de bloques del modelo

Figura 19

Diagrama de bloques del modelo GAN



Nota. Este es un esquema general de la red GAN

4.3.3 Métricas

En general, las métricas que se utilizan en este código para evaluar el desempeño de la red GAN son el MAE (Mean Absolute Error) y el MSE (Mean Squared Error).

Para el discriminador, la pérdida promedio se mide con el MAE de las clasificaciones correctas y las incorrectas. En otras palabras, la pérdida del discriminador indica cuántas curvas se

clasificaron correctamente como reales y cuántas se clasificaron incorrectamente como falsas y viceversa. Para el generador, la pérdida se mide por el MAE entre la salida del discriminador para las imágenes falsas y un tensor de etiquetas de todos 1's. El objetivo es que el generador produzca curvas que se parezcan lo más posible a las curvas reales. La pérdida de la GAN en sí misma se mide como el MAE entre la salida del discriminador para las curvas falsas y un tensor de etiquetas junto con la pérdida del generador. Además de estas medidas de pérdida, también se incluyen gráficas para el MSE del discriminador tanto para las clasificaciones correctas como incorrectas, lo que permite evaluar la precisión de las clasificaciones. El MSE mide el promedio de los cuadrados de las diferencias entre los valores predichos y los valores reales, por lo que un valor bajo indica que el modelo es preciso en sus predicciones.

4.3.4 Resultados

Estas graficas se presentan a modo de comprender el proceso de entrenamiento de la red y ajuste de los parámetros específicamente en el Adulto 1, para los de más se muestra simplemente las métricas mínimas logradas después del entrenamiento. Estos condensan a continuación a modo de tablas y gráficos para facilitar su comprensión.

En la figura 20 se muestran 4 graficas de error, las cuales nos permiten analizar el comportamiento de la red GAN y cada uno de sus componentes, como lo son el discriminador y el generador, comparando el ajuste del error entre las muestras falsa y verdaderas.

Tabla 5

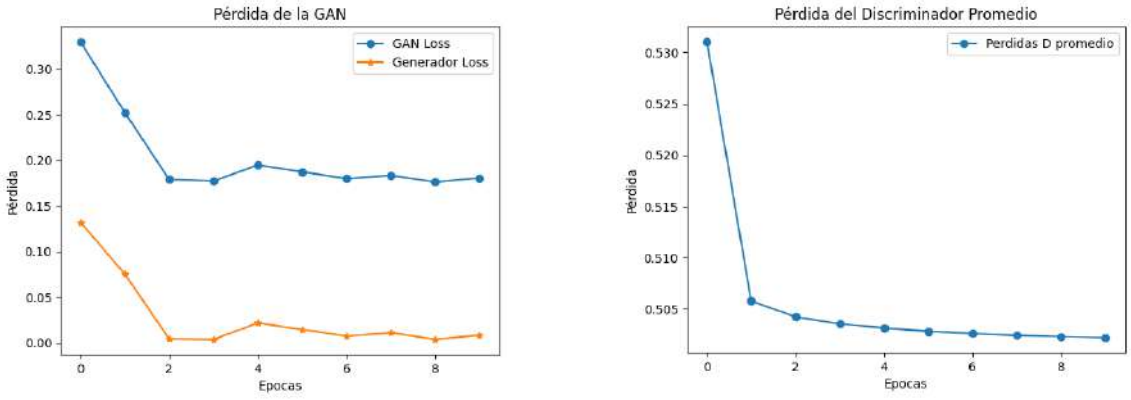
Ajuste de las métricas durante las 10 épocas

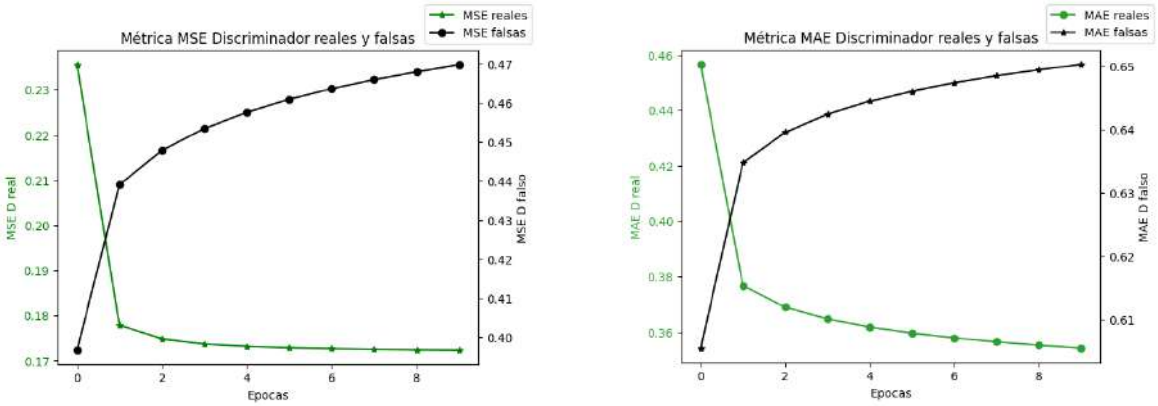
Métricas Paciente 1.					
	Discriminador		GAN		
Época	MAE	MSE	MAE	MAE generador	MAE discriminador
1	0.5062	0.3080	0.2674	0.0896	0.1778
2	0.5044	0.3110	0.1795	0.0047	0.1748
3	0.5036	0.3133	0.1841	0.0104	0.1737
4	0.5032	0.3152	0.1796	0.0064	0.1732
5	0.5029	0.3167	0.1747	0.0018	0.1729
6	0.5026	0.3180	0.1889	0.0162	0.1727
7	0.5025	0.3192	0.1801	0.0076	0.1725
8	0.5023	0.3201	0.1776	0.0051	0.1724
9	0.5022	0.3210	0.1766	0.0042	0.1724
10	0.5021	0.3218	0.1753	0.0030	0.1723

Nota. Se condensan los datos arrojados durante el entrenamiento de la red GAN, esto ocurre durante 10 épocas y se puede ver como mejora el error en cada una de ellas.

Figura 20

Gráficas de perdida por época en el generador, discriminador y GAN



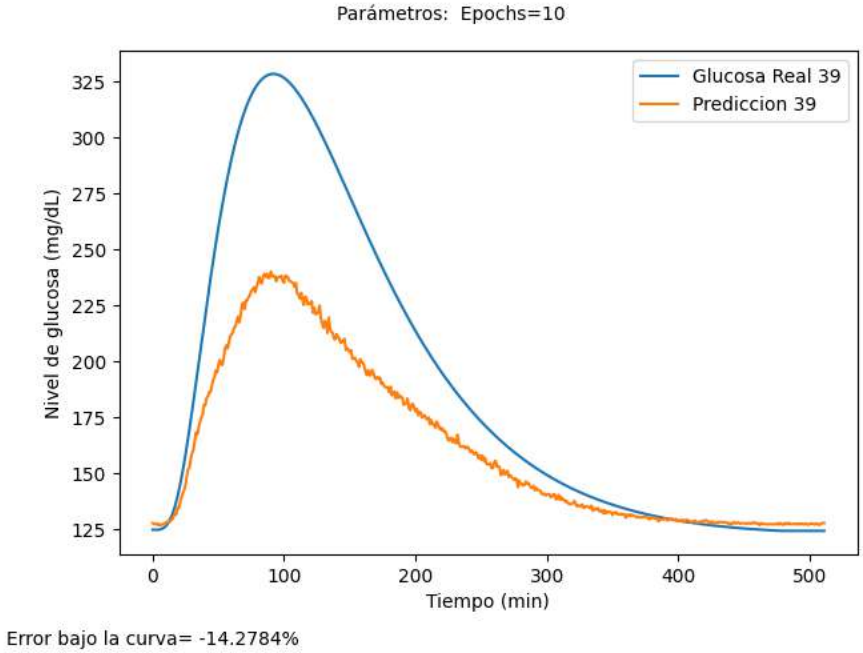


Nota. En la parte superior se presenta la perdida de la GAN y el generador, al lado derecho la gráfica de las perdidas promedio del discriminador. En la parte inferior se compara el comportamiento de las muestras falsas y reales mediante las métricas de MAS y MSE.

Se presenta entonces la gráfica comparativa para el Adulto 1, donde en color azul se presenta la curva real de la ingesta 39 y en color naranja se presenta la predicción lograda por la red después del ajuste de los parámetros. En la parte inferior izquierda se presenta el error bajo la curva con el fin de obtener otro punto de vista de la diferencia entre estas, el signo representa simplemente si el nivel de la predicción es superior o inferior a la real.

Figura 21

Comportamiento de la glucosa real y del modelo GAN para la ingesta 39

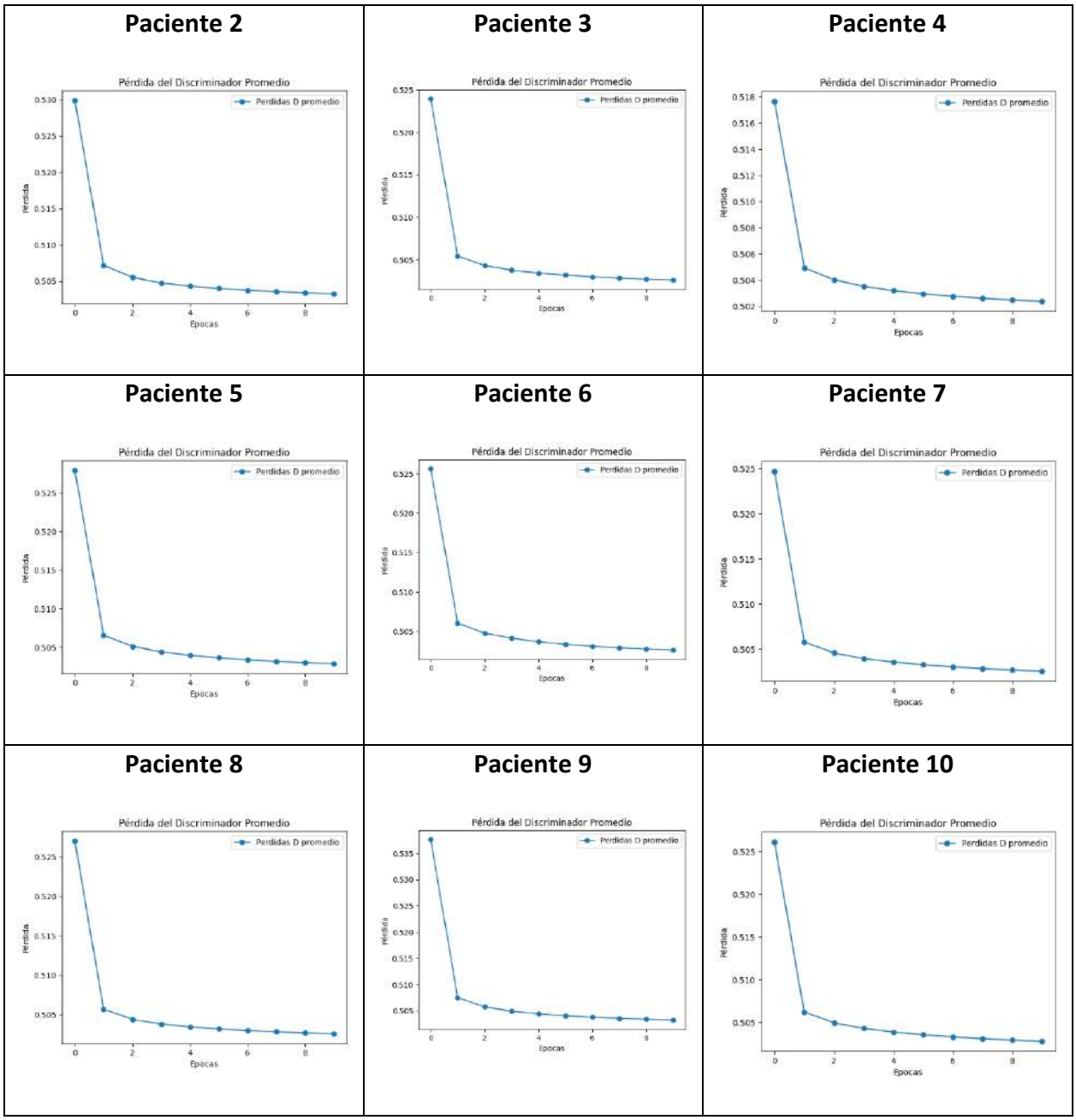


Nota: Se muestra en color azul el comportamiento real de la glucosa y en color naranja la predicción hecha por el modelo.

En la figura 22 se puede apreciar para cada paciente el comportamiento del error MAE en el Discriminador, a medida que suceden las épocas del entrenamiento, estas corresponden a los diez pacientes y en cada uno de ellos se presenta para una ingesta en específico que además corresponde con la misma a mostrada en el caso del modelo GRU.

Figura 22

Gráficas de error del discriminador promedio para los pacientes del 2 al 10

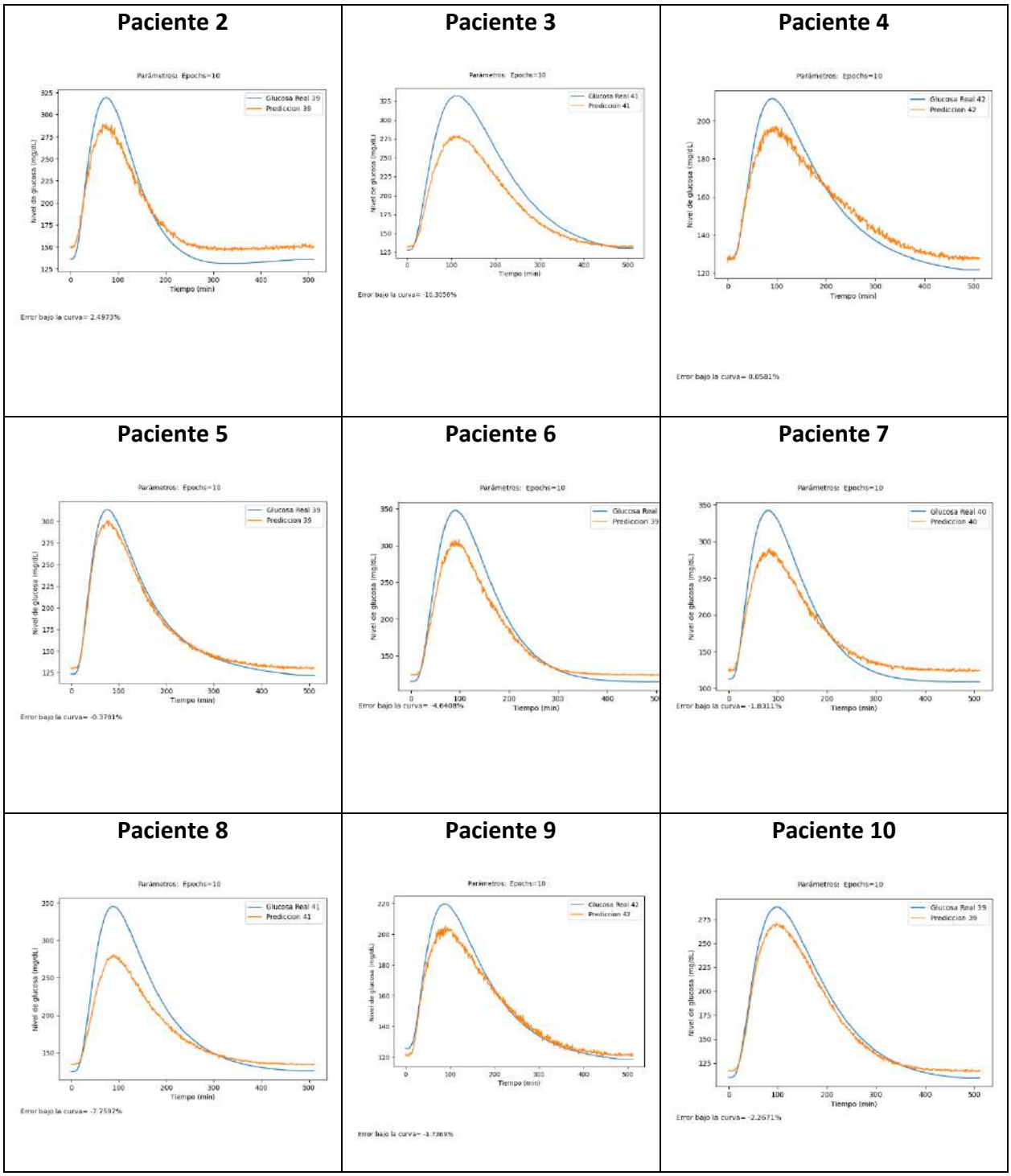


Nota. Se presenta la perdida en el discriminador promedio para cada paciente, se elige este parámetro ya que es el que presenta error más alto en comparación con los demás.

De la misma manera se presenta las gráficas comparativas entre el comportamiento real y la predicción de la curva para cada paciente. También se anexa en la parte inferior de la gráfica un error que corresponde con el error entre el área bajo la curva real y el área bajo la curva de la predicción, sin embargo, se comprobó que no es un buen indicador en este caso.

Figura 23

Predicciones hechas por el modelo GAN para los pacientes 2 al 10



Nota. Se muestran las gráficas comparativas para los 9 pacientes faltantes, con de poder observar la diferencia entre las predicciones.

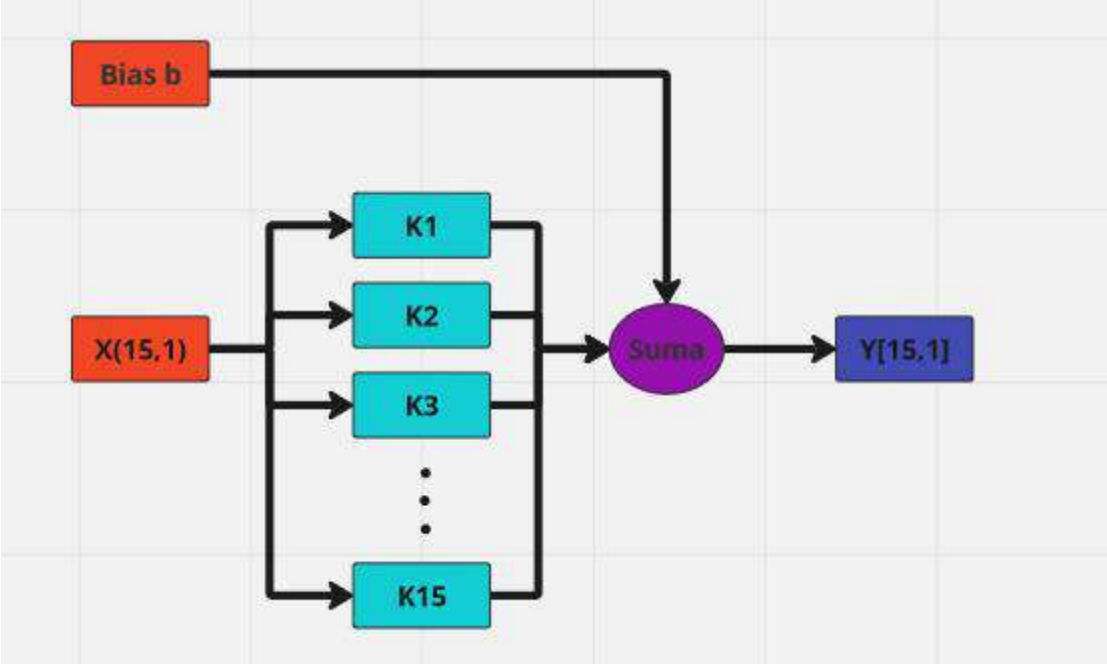
4.5 Modelo SVM

La SVM se usa para determinar a qué cantidad de ingesta de carbohidratos corresponde un determinado comportamiento de glucosa.

Inicialmente se define la arquitectura con la que se va a trabajar. Para este proyecto resultó conveniente trabajar con una SVM de regresión, es decir una “Maquina de Vectores Soporte para la Regresión” (SVR) porque esta nos permite predecir un valor numérico a partir de la entrada en lugar de clasificar. A continuación, se muestra el esquema de la SVM.

Figura 24

Diagrama de bloques de la SVM



Nota. Se muestra el diagrama de bloques para SVM

Se importa la base de datos y se hace un preprocesamiento para dejar todos los datos que contienen un valor de ingesta y eliminar los datos en donde la ingesta de carbohidratos es cero y queda la base de datos como se muestra en la figura 25.

Figura 25

Distribución del conjunto de medicines para el paciente 2

	Carbo	Tiempo	Tiempo_F	Insulina	Glucosa	Ingesta
0	Ingesta1	0	0	4.533647	139.590085	2000.000000
1	Ingesta1	1	1	4.533647	139.599586	2000.000000
2	Ingesta1	2	2	4.533647	139.613904	2000.000000
3	Ingesta1	3	3	4.533647	139.639607	2000.000000
4	Ingesta1	4	4	4.533647	139.685027	2000.000000
5	Ingesta1	5	5	4.533647	139.759275	2000.000000
6	Ingesta1	6	6	4.533647	139.871506	2000.000000
7	Ingesta1	7	7	4.533647	140.030457	2000.000000
8	Ingesta1	8	8	4.533647	140.244173	2000.000000
9	Ingesta1	9	9	4.533647	140.519879	2000.000000
10	Ingesta1	10	10	4.533647	140.863938	2000.000000
11	Ingesta1	11	11	4.533647	141.281855	2000.000000
12	Ingesta1	12	12	4.533647	141.778316	2000.000000
13	Ingesta1	13	13	4.533647	142.357249	2000.000000
14	Ingesta1	14	14	4.533647	143.021878	2000.000000

Nota. Se muestran las primeras 15 ingestas y su correspondiente valor de glucosa, insulina e ingesta de carbohidratos. Todas las filas en las cuales la ingesta es cero se ha eliminado.

4.5.1 Definición del modelo

El modelo se construyó utilizando una Máquina de Soporte Vectorial para regresión (SVR por sus siglas en inglés) con un kernel radial (rbf), que es un tipo de función de kernel utilizada en SVM. El grado del polinomio se estableció en 3 y el parámetro de escala gamma se fijó en 'escala'. El coeficiente de regularización C se estableció en 10000 y el parámetro de holgura epsilon se fijó en 0.1.

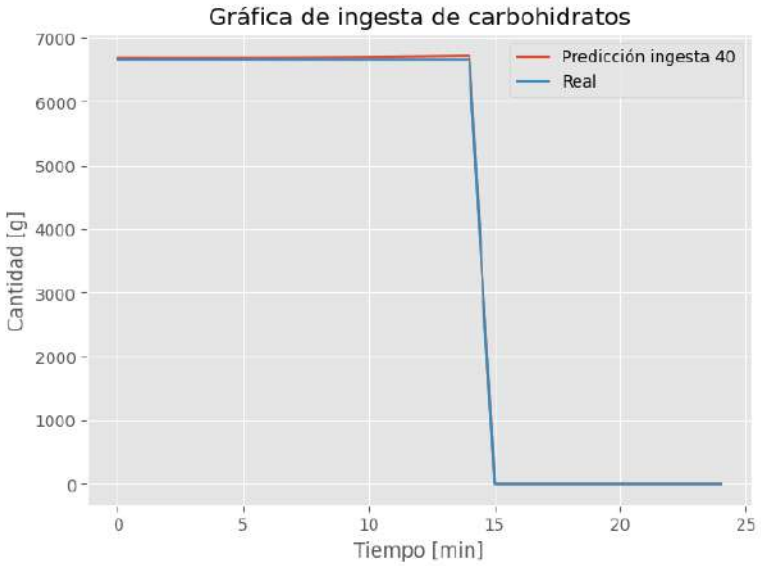
Además, se establecieron otros parámetros como el coeficiente constante (coef0), la tolerancia (tol), el tamaño de batch (batch_size), entre otros. Estos parámetros se ajustaron de acuerdo con las características específicas del conjunto de datos y se seleccionaron para optimizar el desempeño del modelo.

Luego, se entrenó el modelo usando el método fit() de la clase SVR, utilizando los datos de entrenamiento (X_train, y_train) y se ajustaron los parámetros del modelo para minimizar la función de pérdida MSE. Finalmente, el modelo se utilizó para hacer predicciones de ingesta de carbohidratos, estos datos se graficas a continuación para visualizar su eficiencia, además se condensan en la sección de resultados.

4.5.1 Resultados

Figura 26

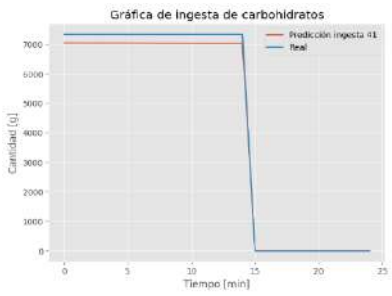
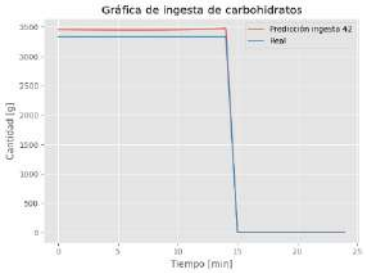
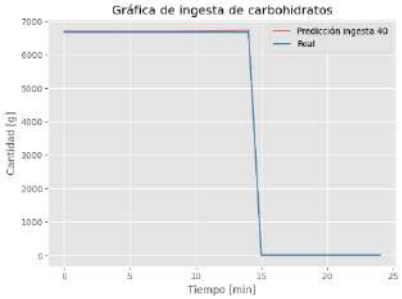
Predicciones de carbohidrato para el paciente 1

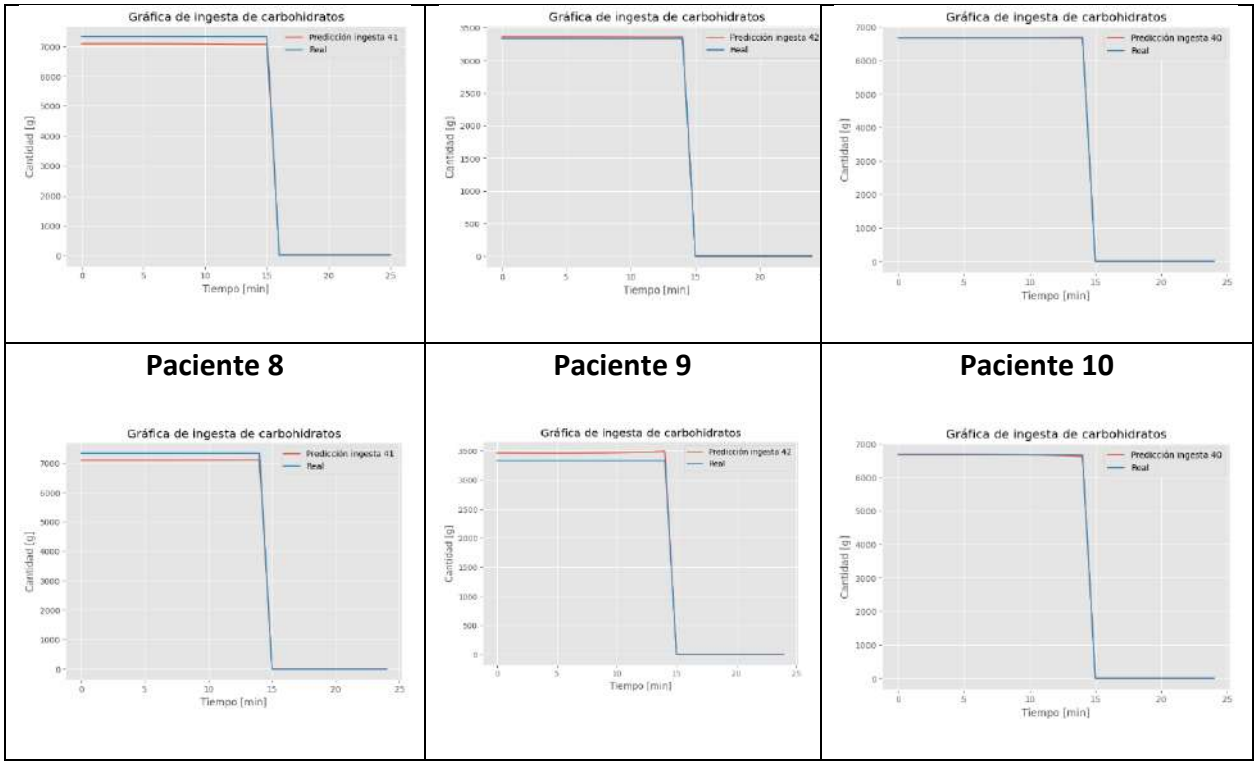


Nota. Gráfica con los valores predichos y reales de los carbohidratos correspondientes a una curva de glucosa

Figura 27

Predicciones de carbohidrato hecha por la SVR para los pacientes del 2 al 10

Paciente 2	**Paciente 3**	**Paciente 4**
		
Paciente 5	**Paciente 6**	**Paciente 7**



Nota. Se muestran las gráficas comparativas para los 9 pacientes faltantes, con de poder observar la diferencia entre las predicciones.

5. Análisis de resultados

5.1 Análisis GRU

Finalmente se extraen los datos más importantes del entrenamiento y predicción en cada modelo, estos datos se condensan de manera numérica en la tabla 6 y se grafican en la figura 23, con el fin de analizar la media de cada tipo de error entre los 5 folders para tener una certeza de la veracidad de la predicción.

Tabla 6

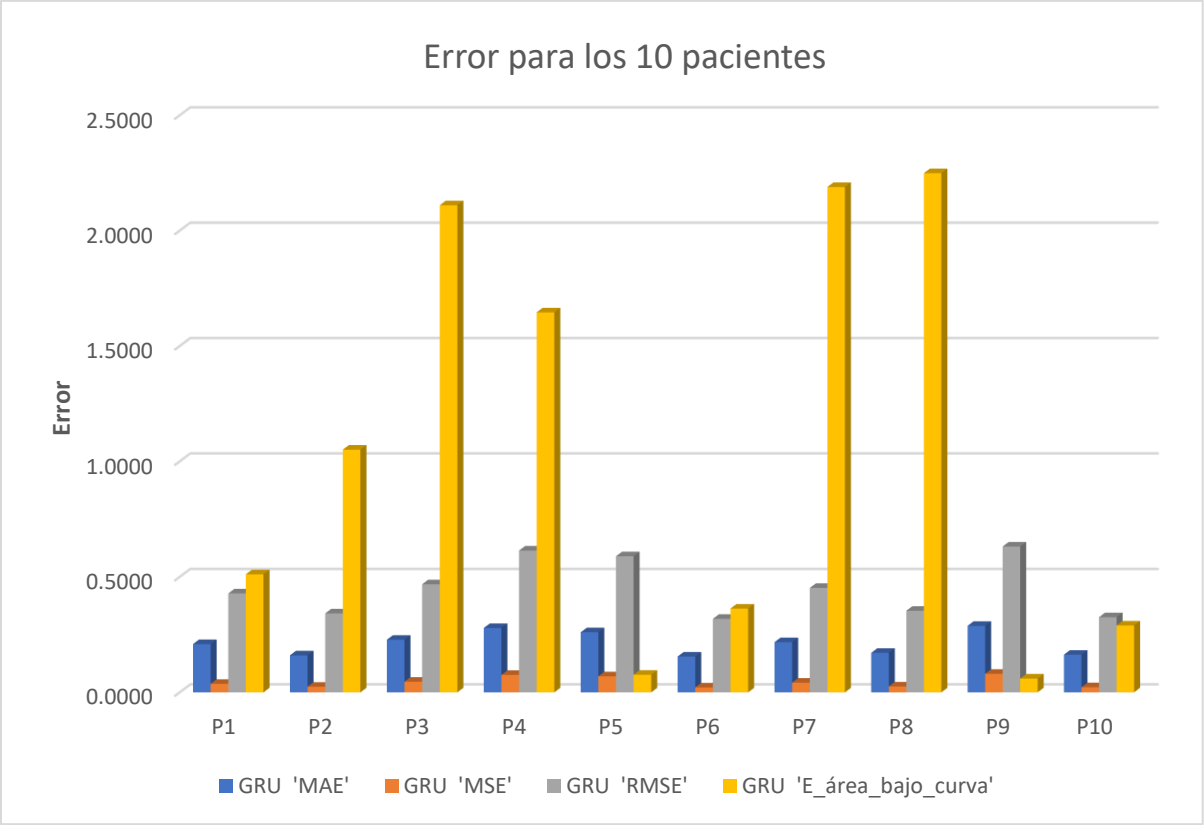
Error MAE, MSE, RMSE y error bajo la curva para los 10 pacientes

	GRU			
Paciente	**MAE**	**MSE**	**RMSE**	**Error bajo la curva**
P1	0.2091	0.0369	0.4282	0.5109
P2	0.1598	0.0242	0.3415	1.0507
P3	0.2278	0.0458	0.4678	2.1097
P4	0.2783	0.0754	0.6138	1.6455
P5	0.2600	0.0694	0.5890	0.0759
P6	0.1547	0.0211	0.3182	0.3626
P7	0.2172	0.0417	0.4520	2.1890
P8	0.1712	0.0251	0.3530	2.2489
P9	0.2879	0.0800	0.6314	0.0597
P10	0.1629	0.0222	0.3252	0.2894

Nota. En este caso se elige una ingesta de manera aleatoria para hacer la predicción.

Figura 28

Gráfica del error para los 10 pacientes mostrados en la tabla 6



Nota. Se grafica el error en la predicción de cada uno de los pacientes, donde más que compararlo, se pretende mostrar el error presentado para cada uno de ellos.

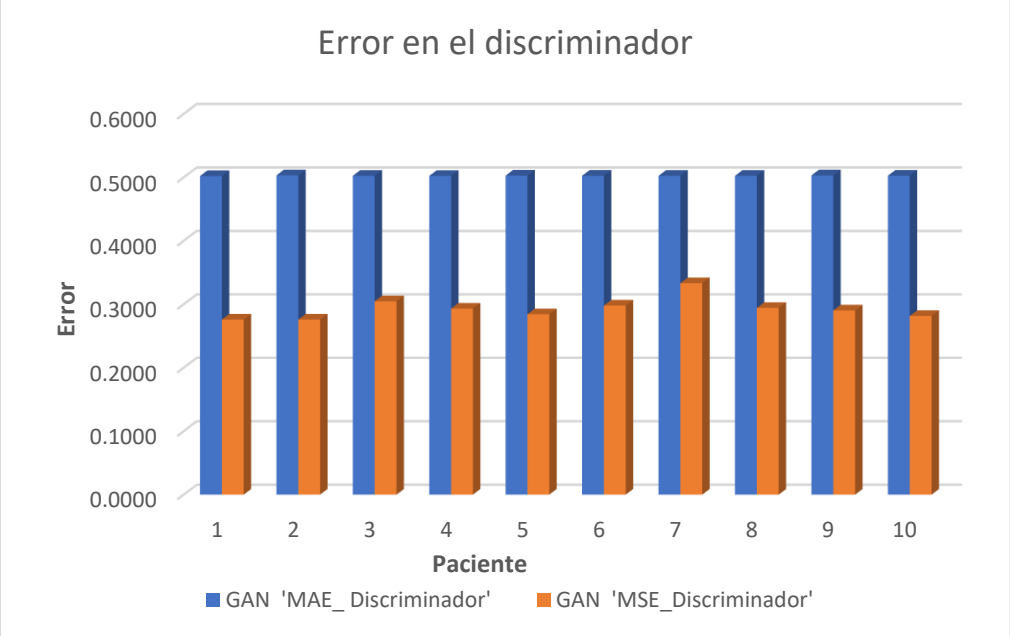
El error bajo la curva nos da una percepción más realista de que tan alejada esta la predicción de la curva real, como se puede examinar los pacientes 3, 4, 7 y 8 son los que presentan mayor error, esta información corresponde a lo observado en la figura 28.

5.2 Análisis GAN

A continuación, se presenta la gráfica del error MAE y MSE del discriminador, Esto se hace para cada paciente con el fin de mostrar el desempeño ante cada uno de ellos.

Figura 29

Resultados de las métricas de desempeño del discriminador

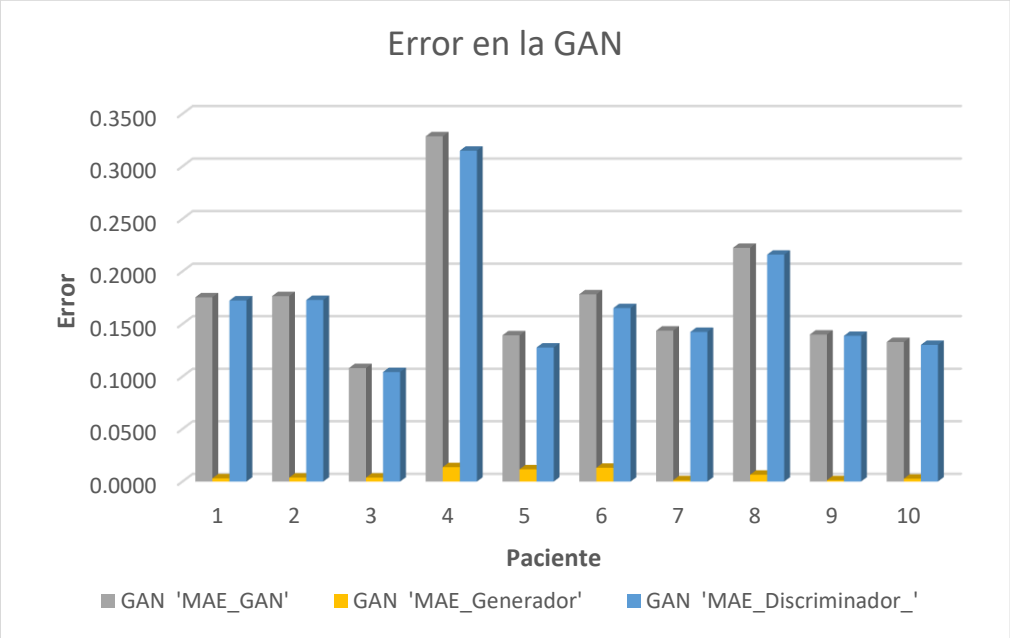


Nota. En la figura 29 se grafican los resultados logrados después el entrenamiento de los 10 pacientes.

Luego se grafican las métricas generales en modelo GAN, donde compara únicamente el MAE para la GAN, el Generador y el discriminador, estos valores hacen referencia a los mínimos de error encontrados después de 10 epocas de entrenamiento.

Figura 30

Gráficas del MAE para la GAN, el generador y el discriminador



Nota. Se muestras la comparación entre las métricas de desempeño (MAE) para los componentes de la GAN con el fin de comparar y analizar cual o cuales están generando error muy alto que puedan hacer que en general el modelo no presente buenos resultados.

En la Figura 30 se puede observar que el error en el modelo generado es muy bajo, lo que sugiere que el bloque del generador funciona bien en rangos muy bajos. Por otro lado, el error tanto en la GAN como en el discriminador es mucho más alto, lo que indica que no están funcionando correctamente. Debido a este error, se afecta el funcionamiento general de la red.

En cuanto a la Figura 29, se puede notar que tanto el MAE como el MSE son demasiado altos. Si nos fijamos en el MAE únicamente, vemos que está en el rango de 0,5, lo que equivale a un 50% de error. Esto indica que el entrenamiento inicial del discriminador es muy básico y no puede distinguir entre las curvas falsas y reales, lo que afecta directamente el comportamiento de la red.

Esto puede deberse a la falta de ajuste de algunos parámetros o al sobre entrenamiento de la red, ya que en las gráficas se observa que la salida o curva generada normalmente no varía mucho sin importar que el valor de las ingestas cambie.

5.3 Comparación del mejor modelo entre GRU y GAN

Tabla 7

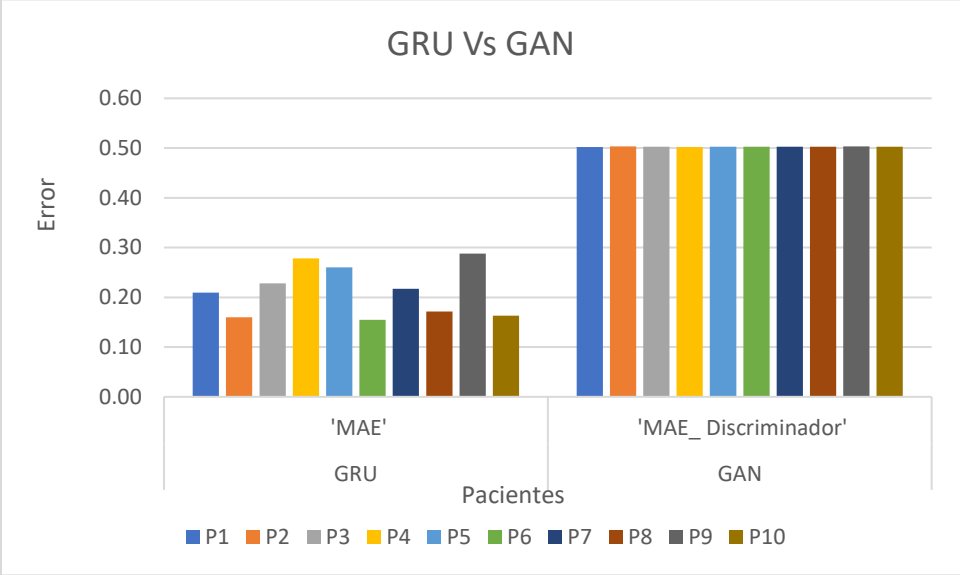
Métricas de desempeño de los dos algoritmos implementados

	GRU			GAN				
Paciente	MAE	RMSE	E_área	MAE_ D	MSE_D	MAE_GAN	MAE_Gen	E_área %
P1	0.2091	0.4282	0.5109	0.5021	0.2759	0.1753	0.0030	14.2784
P2	0.1598	0.3415	1.0507	0.5031	0.2759	0.1765	0.0037	2.4972
P3	0.2278	0.4678	2.1097	0.5025	0.3046	0.1079	0.0037	10.3055
P4	0.2783	0.6138	1.6455	0.5023	0.2931	0.3286	0.0137	0.0581
P5	0.2600	0.5890	0.0759	0.5028	0.2843	0.1391	0.0116	0.3780
P6	0.1547	0.3182	0.3626	0.5025	0.2979	0.1782	0.0131	4.6407
P7	0.2172	0.4520	2.1890	0.5025	0.3332	0.1436	0.0013	1.8310
P8	0.1712	0.3530	2.2489	0.5025	0.2942	0.2224	0.0065	7.7597
P9	0.2879	0.6314	0.0597	0.5031	0.2903	0.1399	0.0012	1.7368
P10	0.1629	0.3252	0.2894	0.5027	0.2815	0.1328	0.0027	2.2671

Nota. En la tabla anterior se presentan las métricas de desempeño MAE, MSE para cada uno de los modelos, así como el error bajo la curva obtenido.

Figura 31

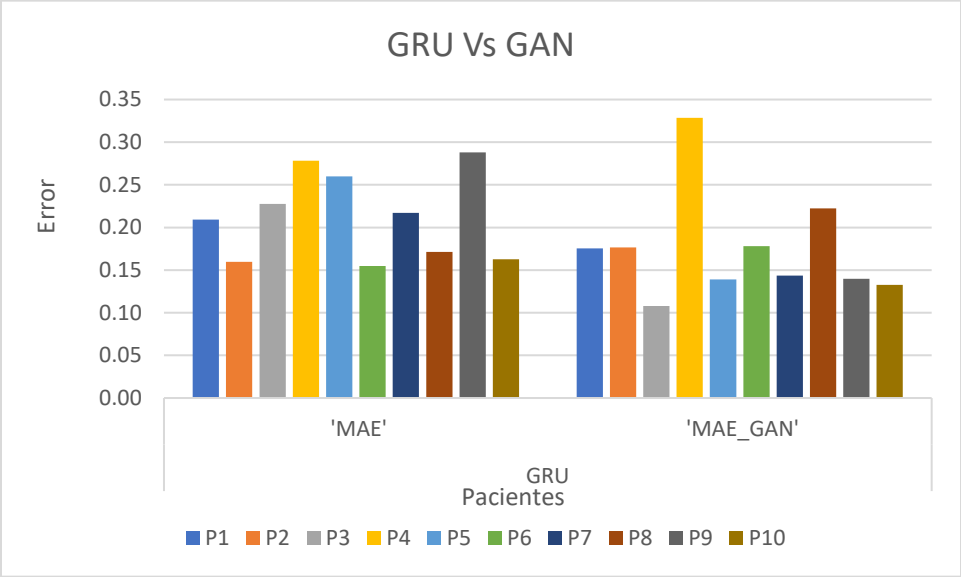
Comparativa de MAE entre GRU y el discriminador de la GAN



Nota. Se grafica el error en el modelo GRU respecto al del Discriminador en la GAN, con el fin de hacer comparaciones entre cada uno de los componentes de la GAN

Figura 32

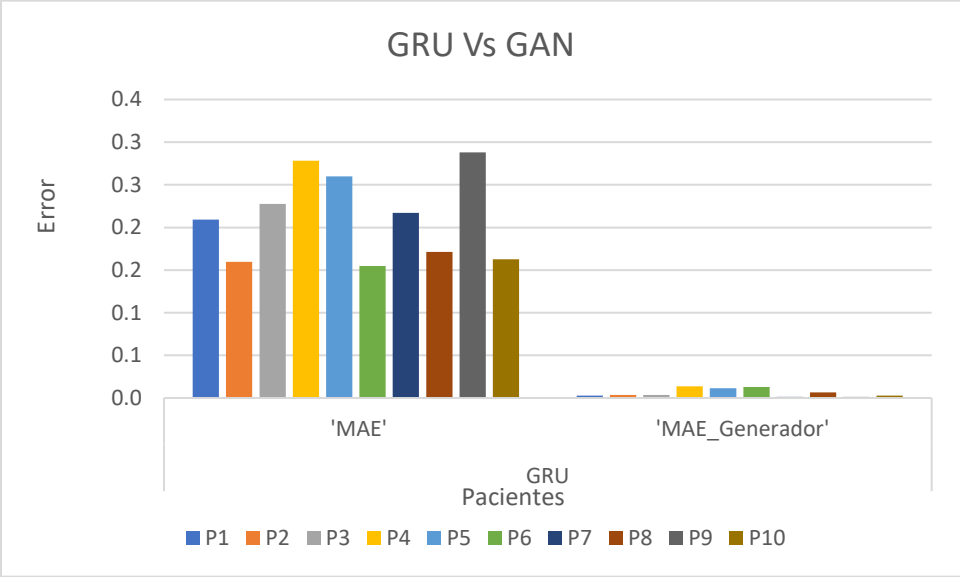
Comparativa del error MAE para la GAN, el generador y el discriminador



Nota. Se compara el error de la GAN general con el de la GRU.

Figura 33

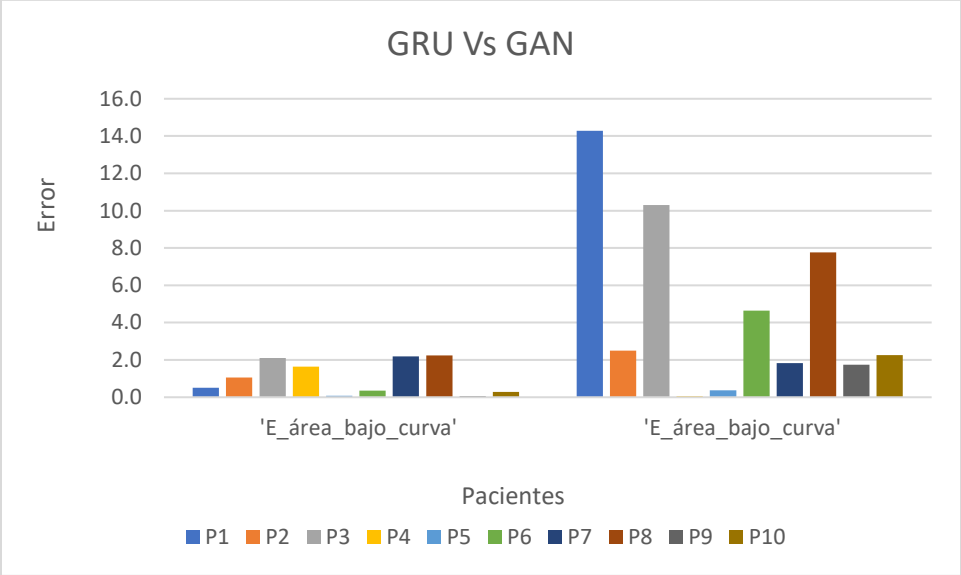
Comparativa entre la GRU y el generador de la GAN



Nota. Se compara el mismo error MAE de la GRU de las gráficas anteriores, pero en este caso con el MAE del generador.

Figura 34

Comparativa del error bajo la curva en las predicciones hechas por los modelos GRU y GAN



Nota. En este caso se calculó y graficó el error bajo la curva para cada paciente y cada modelo (GRU y GAN en ese orden), donde se puede establecer de manera visual la estabilidad de las predicciones a medida que se cambia el paciente.

5.4 Selección del modelo con mejor desempeño

Durante el desarrollo del proyecto, se evaluaron dos diferentes algoritmos para modelar las series de tiempo de la glucosa. La elección final fue el algoritmo GRU (Gated Recurrent Unit) debido a su mejor desempeño en comparación con la GAN (Generative Adversarial Network).

En primer lugar, la GRU presentó errores bajo la curva inferiores al 3%, lo que es significativamente mejor que los resultados obtenidos por la GAN, que pueden llegar al 14% o incluso más. Además, estos valores son altamente confiables, lo que significa que la GRU es capaz de predecir con precisión los niveles de glucosa en la sangre. Otra ventaja de la GRU es que se ajusta muy bien a las diferentes escalas y comportamientos de la glucosa. La GRU es capaz de adaptarse a los patrones de glucosa que varían entre pacientes, lo que es crucial en el contexto

clínico. Además, la GRU es más simple y fácil de optimizar en comparación con la GAN, lo que la hace una opción más práctica.

En resumen, la elección de la GRU se basó en su mejor desempeño en términos de precisión y adaptabilidad, así como en su simplicidad y facilidad para el uso y la implementación, permitiendo implementarse en un procesador o controlador más sencillo. Por todo esto el desarrollo del proyecto continua con el modelo GRU.

5.5 Análisis SVM

Tabla 8

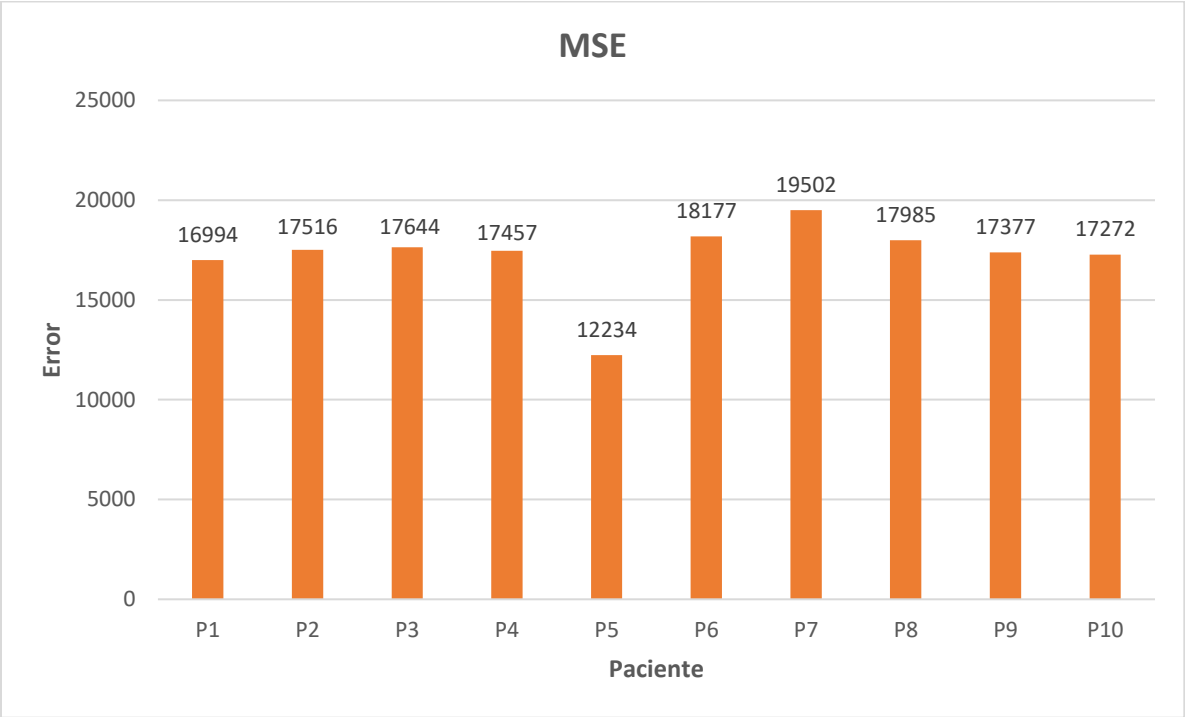
Pérdidas MSE y MAE para cada paciente con el modelo SVM

Paciente	MAE	MSE
P1	34.8798	1320.475
P2	4.33459	25.68773
P3	25.64724	1273.993
P4	28.83036	954.994
P5	40.95778	1689.848
P6	52.07569	2712.407
P7	12.27083	176.8038
P8	45.23567	2047.029
P9	15.64883	632.6497
P10	17.5449	412.2784

Nota. Se muestran los errores MSE y MAE para cada paciente y se resalta con verde y naranja los pacientes con menor y mayor pérdida MSE respectivamente

Figura 35

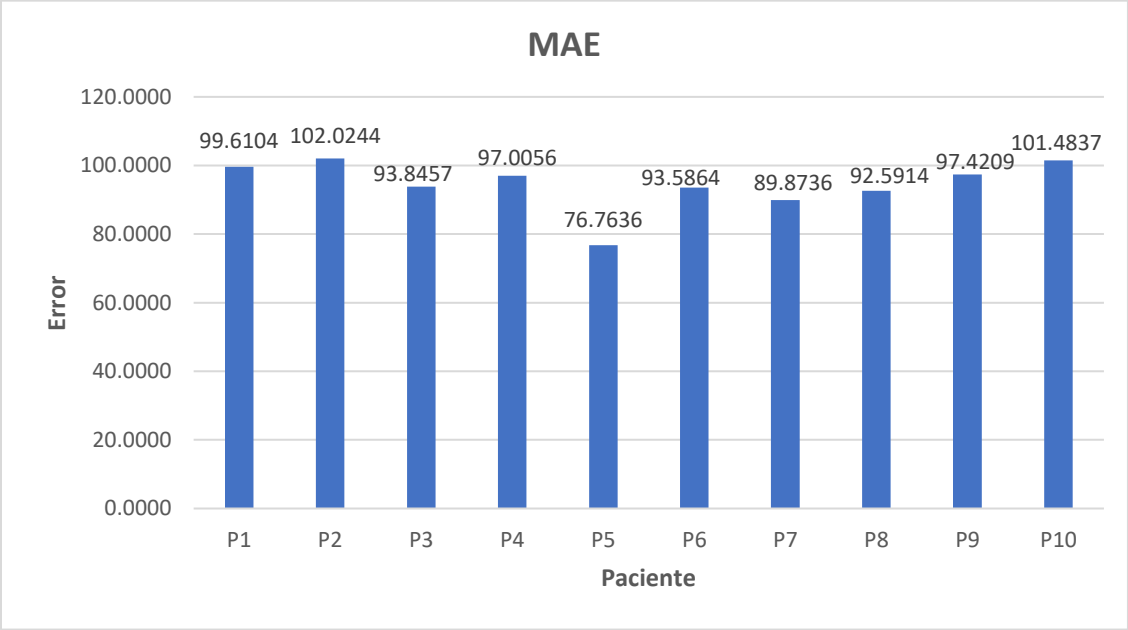
Representación del error MSE en barras



Nota. Se muestran el error MSE para cada paciente

Figura 36

Representación en barras de la pérdida MAE de los 10 pacientes



Nota. Se muestran el error MAE para cada paciente

6. Implementación del algoritmo con mejor desempeño

6.1 Extracción del modelo

En el desarrollo del último objetivo, el cual comprende la validación en tiempo real del algoritmo con mejor desempeño mediante su programación en un sistema embebido, se definió inicialmente el modelo GRU como el modelo a implementar. Manteniendo exactamente la misma

configuración y ajuste de parámetros, entonces se parte del punto donde el modelo ya se ha entrenado y probado su desempeño.

El proceso que se lleva a cabo en primera instancia es guardar el modelo en un archivo con formato ***model_name.h5,*** esto genera un archivo que se puede descargar y compartir a otro dispositivo. Este tipo de archivo contiene la información completa del modelo entrenado y con todos sus respectivos parámetros.

6.2 Implementación en microprocesador.

Inicialmente se hace una comparación entre las características relevantes entre los modelos de dispositivos de desarrollo más comunes.

Tabla 9

Comparativa básica de algunos microcontroladores o microprocesadores más comunes.

Característica	Raspberry Pi 3	Raspberry Pi Zero W	Arduino	ESP32
Procesamiento	Broadcom BCM2837B0, Quad-core 1.2GHz	Broadcom BCM2835, Single-core 1GHz	Diversos modelos con microcontroladores	Xtensa Dual-core 32-bit LX6, hasta 240MHz
Velocidad del procesador	1.2 GHz	1 GHz	Varía según el modelo, hasta 84MHz	Hasta 240 MHz
Memoria RAM	1 GB DDR2 SDRAM	512 MB LPDDR2 SDRAM	Depende del modelo y puede ser limitada	520 KB SRAM + 4 MB PSRAM
Sistema operativo	Permite instalar un sistema operativo completo (ej. Raspbian)	Permite instalar un sistema operativo completo (ej. Raspbian)	Sin sistema operativo, programa directamente en lenguaje C/C++	Puede ejecutar MicroPython o programas en lenguaje C/C++
Facilidad de instalación	Fácil instalación del sistema	Fácil instalación del sistema	Programación directa sin necesidad de	Requiere cargar un firmware

Característica	Raspberry Pi 3	Raspberry Pi Zero W	Arduino	ESP32
	operativo en una tarjeta SD	operativo en una tarjeta SD	instalar software adicional	específico para la programación
Lenguajes de programación	Amplio soporte de lenguajes de programación, incluyendo Python	Amplio soporte de lenguajes de programación, incluyendo Python	Mayor enfoque en lenguaje C/C++	Soporte para MicroPython y lenguaje C/C++
Conectividad	Ethernet, Wi-Fi 802.11n, Bluetooth 4.2, USB 2.0, GPIO	Wi-Fi 802.11n, Bluetooth 4.1, USB micro-B, GPIO	Depende del modelo y puede ser limitada	Wi-Fi 802.11n, Bluetooth 4.2, USB 2.0, GPIO
Periféricos compatibles	Amplia variedad de periféricos compatibles	Menos variedad de periféricos compatibles	Amplia variedad de periféricos compatibles	Menos variedad de periféricos compatibles
Consumo de energía	Aprox. 2.5 W en funcionamiento normal	Aprox. 0.5 W en funcionamiento normal	Bajo consumo de energía	Consumo de energía moderado
Costo	Varía según el modelo	Económico	Económico	Económico

Nota. Se resaltan las características optimas o requeridas para la implementación del modelo.

En la anterior tabla se presenta la comparación entre cuatro de las principales opciones de microcontroladores disponibles en el mercado. Entre las opciones se encuentran: dos modelos de raspberry pi, un Arduino genérico y una ESP 32. Para la elección del microcontrolador se tuvieron en cuenta las siguientes características:

- Capacidad de ejecutar un script en Python
- Una conexión inalámbrica a Wi-Fi para hacer la transmisión de los datos
- Alta capacidad de procesamiento

Teniendo en cuenta estas características se seleccionó la Raspberry pi 3b para la implementación. Para este proyecto se requiere procesar una gran cantidad de datos y como se muestra en la tabla 9 esta opción tiene la mejor capacidad de procesamiento.

En el sistema desarrollado, se muestra un diagrama que representa el flujo de información a través de dos dispositivos. El primero de ellos es responsable de enviar la información, simulando el paciente censado. El segundo dispositivo, una Raspberry Pi 3, se encarga del procesamiento y generación de predicciones. Para esto, es necesario cargar las librerías y el modelo de la red neuronal en la Raspberry Pi, además de asegurarse de que esté lista para recibir los datos.

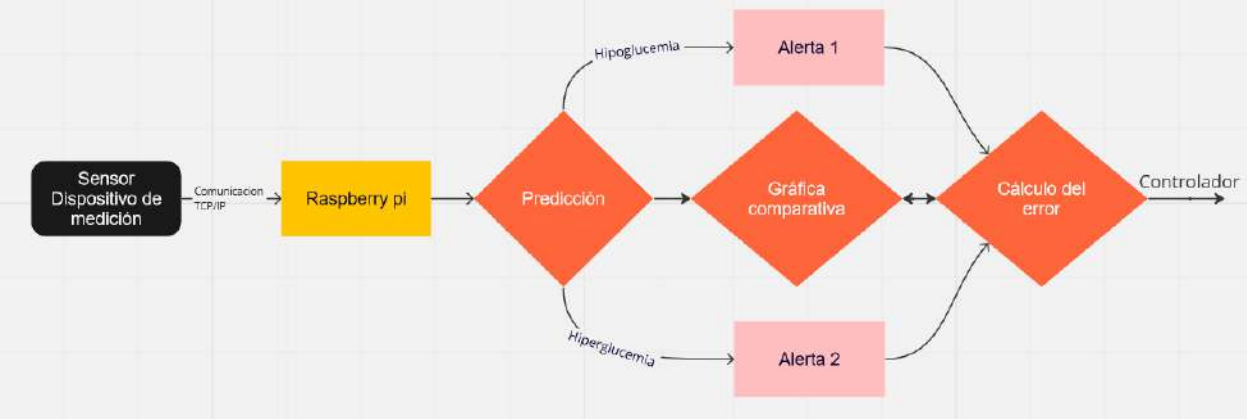
Por otro lado, el segundo dispositivo ejecuta un script de Python capaz de enviar los datos de insulina y glucosa mediante una conexión inalámbrica. Esto ocurre cada minuto durante un período de 15 minutos. El script puede ser ejecutado en una computadora o en una Raspberry Pi que cuente con Wi-Fi y sea capaz de ejecutar Python. Este dispositivo se comunica con la Raspberry Pi mediante el protocolo TCP/IP.

Una vez completada la transmisión de los datos, el script pasa estos datos al modelo y se inicia un temporizador para monitorear el tiempo de procesamiento. El tiempo se mide hasta que el modelo genera la predicción, a partir de la cual se pueden extraer diferentes estados del paciente, como hiperglucemia o hipoglucemia en caso de que se presenten. Después de esto, se genera una gráfica comparativa que permite visualizar el comportamiento de la glucosa en un tiempo

determinado junto con el comportamiento real. Esto nos brinda la posibilidad de comparar numéricamente las gráficas y mostrar el error del área bajo la curva.

Figura 37

Flujo de datos en el sistema



Nota. Se muestra el flujo de información dentro de los sistemas usados.

En esta sección se continua con la selección del microprocesador, en este caso se eligió la Raspberry pi3B+, teniendo en cuenta que se trata de un dispositivo con una gran capacidad de procesamiento y ofrece una gran ventaja en el sentido que permite hacer la implementación en Python, además instalando un sistema operativo Raspbian de 64bits, nos permite instalar directamente tensorflow, keras y las librerías con exactamente la misma versión que se usó en el entrenamiento, después de instalar el sistema operativo, Python y las librerías adecuadas se continua con el proceso así:

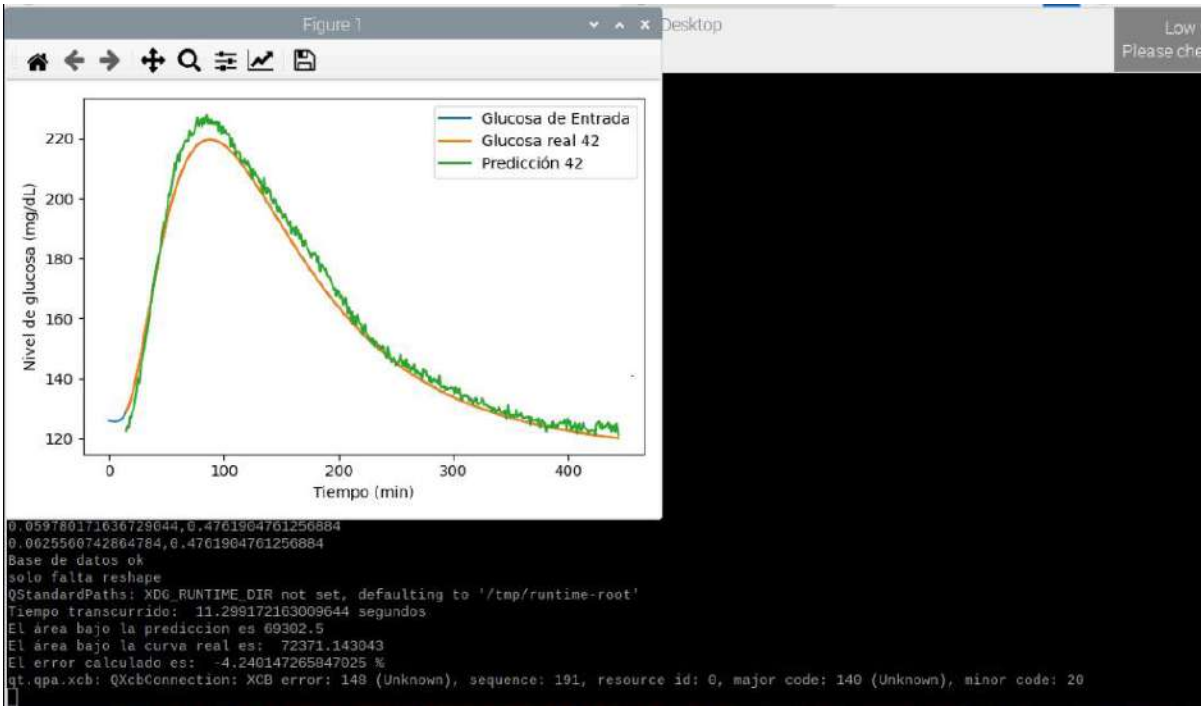
Se crea un script en Python el cual contiene el código necesario para importar las librerías como tensorflow, numpy, pandas, keras, matplotlib, entre otras. Posteriormente se lee el modelo model_name.h5, a continuación, se imprime el resumen del modelo, donde de primera mano se puede comprobar que corresponde exactamente con el generado después del entrenamiento de la GRU base.

Finalmente, solo resta hacer una predicción con el modelo importado, para lo cual se le debe suministrar de la misma manera las 15 muestras de insulina y glucosa para que realice la predicción.

Para el envío de información se usa una comunicación TCP/IP entre la Raspberry y un dispositivo emisor que en este caso puede ser otra Raspberry o un computador. (Cabe resaltar que para este tipo de comunicación los dos dispositivos deben estar conectados a la misma red Wi-Fi). Primero se debe ejecutar el programa en la Raspberry de tal forma que cargue el modelo y espere por una conexión para iniciar a recibir la información, cuando esto suceda se ejecuta el programa en el otro dispositivo(este permite seleccionar el número de ingesta a enviar) y en cuestión de segundos inicia la transmisión de los datos, donde se envía dos datos cada minuto, glucosa e insulina, cuando el último de los 15 datos ha sido trasmitido, se genera automáticamente la gráfica de la glucosa real y la glucosa generada como predicción, el tiempo que tarda el sistema en generar esta respuesta es medido y en la siguiente figura se puede apreciar tanto la gráfica como el tiempo que tardo el algoritmo en hacer la predicción.

Figura 38

Comunicación y resultados



Nota. Se puede apreciar los resultados generados, el tiempo tardado en generar la predicción y el formato de los valores transmitidos.

En la figura 38 se puede ver como en el terminal se muestran algunos avisos o mensajes exponiendo el error calculado bajo la curva en cada una de las predicciones, con el fin de comprobar la precisión o que tanto se acerca la predicción a la curva real. Este error se calcula comparando el área bajo cada una de las curvas, en este caso específico para el paciente 9 en su ingesta 42 se presenta un error del 4.24 % que, aunque aumenta el error en comparación con la ejecución directamente desde el mismo dispositivo, sigue siendo un resultado notablemente bueno y confiable. Es decir que el proceso de la comunicación de los datos genera un aumento en el error

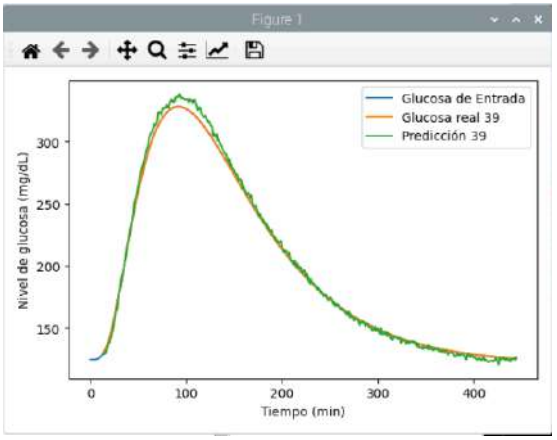
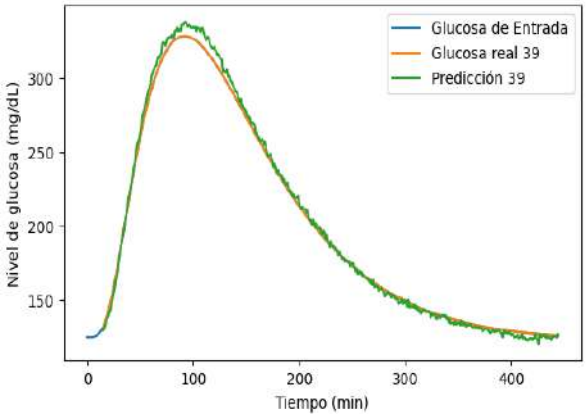
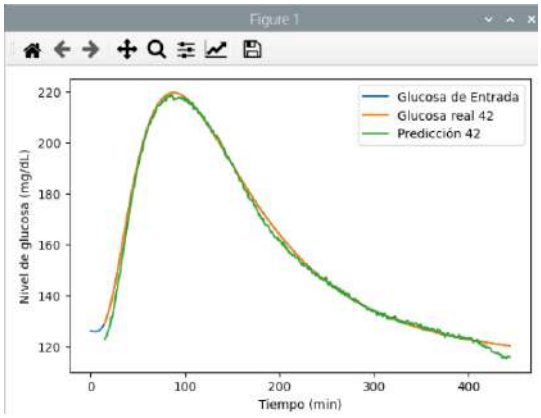
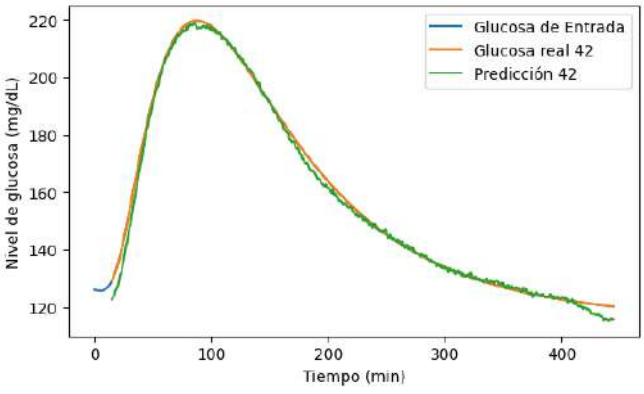
debido a el corte en la cantidad de decimales y al proceso de re-escalamiento para retornar los valores a la escala real.

Cuando finaliza la ejecución del programa podemos encontrar algunos mensajes en la terminal, entre estos el error bajo la curva que se calcula con la comparación entre la predicción y la curva real, la otra variable hace referencia al tiempo que tarda el modelo en generar la predicción después de qué recibe el último dato. En las diferentes pruebas realizadas se encontró que este tiempo no supera los 20 segundos, lo cual se considera muy bueno ya que se está hablando de alrededor de ocho horas, visto de esta manera este retardo podría considerarse como despreciable.

6.3 Pruebas de funcionamiento

Figura 39

Comparativa de la predicción obtenida con la Raspberry pi 3 y la obtenida con el modelo GRU en Google Colab

Raspberry pi 3	**Modelo GRU**
Paciente 1	**Paciente 1**
	
Paciente 9	**Paciente 9**
	

Nota. En la columna derecha se muestran las predicciones hechas en la Raspberry y en la columna izquierda se muestran las predicciones hechas por el modelo en Google Colab.

7. Conclusiones

Se crearon dos algoritmos capaces de predecir los niveles de glucosa en la sangre y se realizó la implementación de uno de ellos logrando anticipar posibles eventos de hiperglucemia e hipoglucemia en pacientes con diabetes tipo 1, después de comparar los dos modelos de aprendizaje usando las métricas de desempeño propuestas se encontró que el Modelo GRU presenta notoriamente mejores resultados respecto a la GAN**.**

A partir del simulador UVA/PADOVA se extrajeron los datos apropiados para construir el conjunto de mediciones con variables de ingesta de carbohidratos, glucosa e insulina, pero se debe tener en cuenta que el tiempo de simulación debe contener al menos 40 ingestas, de tal manera que la base de datos brinde la información suficiente para entrenar los modelos de aprendizaje supervisado.

Se seleccionaron y entrenaron las arquitecturas de redes neuronales GAN, GRU y el algoritmo SVM, donde después de comparar las métricas de error mínimo logrado por cada algoritmo se concluye que la res GRU tiene mejor desempeño, ya que presenta un error máximo de 2.5% respecto al 15% de la GAN aproximadamente

Finalmente se hizo la implementación de la red GRU en un microprocesador Raspberry pi 3 y se hizo la transmisión de los datos en tiempo real, logrando resultados confiables ya que el error del área bajo la curva no supera el 7%, además de esto el tiempo de procesamiento no supera los 20 segundos y se puede considerar como un buen resultado ya que este es el tiempo que se necesitaría para hacer una predicción de al menos de 7 horas.

8. Recomendaciones

Aunque se ha demostrado que la red GRU presenta un buen desempeño en la predicción de los niveles de glucosa en la sangre, no se descarta la posibilidad de obtener resultados similares o incluso mejores con la red GAN. Por tanto, se recomienda continuar con el estudio y la optimización del modelo, explorando diferentes arquitecturas y bloques en el generador y discriminador para mejorar la precisión de las predicciones.

En cuanto a la implementación, se sugiere buscar un dispositivo más compacto y de menor consumo energético para establecer una conexión directa con los instrumentos de medición de glucosa, lo que permitiría desarrollar un dispositivo portátil y funcional para cualquier paciente. Esto facilitaría el monitoreo continuo y preciso de los niveles de glucosa, mejorando el control y la calidad de vida de los pacientes diabéticos.

Referencias Bibliográficas

¿Qué es la diabetes? (2022, July 13). Cdc.gov. https://www.cdc.gov/diabetes/spanish/basics/diabetes.html

Anaya Vejar, Tibaduiza Burgos, D. A., & Villamizar Mejia, R. (2010). Propuesta del modelo reducido orientado hacia el control del comportamiento dinámico de la glucosa en pacientes con diabetes mellitus tipo I [recurso electrónico]. UIS

Atria Innovation. (2019, October 22). Qué son las redes neuronales y sus funciones. ATRIA Innovation. https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

Barquilla García, A. (2017). Actualización breve en diabetes para médicos de atención primaria. Isciii.Es.https://scielo.isciii.es/pdf/sanipe/v19n2/es_04_revision.pdf

Bergman, R. N., Cobelli, C., & Heise, T. (2019). Non-adjunctive use of continuous glucose monitoring for diabetes treatment decisions. The Lancet Diabetes & Endocrinology, 7(11), 836-843.

CDC Español. (2023, March 27). Centros para el Control y la Prevención de Enfermedades CDC. Centers for Disease Control and Prevention. https://www.cdc.gov/spanish/

Chauhan, N. S. (2020). Optimization algorithms in neural networks. KDnuggets. https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297. https://doi.org/10.1007/BF00994018

De Colombia, M. de S. y. P. S. (s.f.). Tres de cada 100 colombianos tienen diabetes. Gov.co. Retrieved March 28, 2023, from https://www.minsalud.gov.co/Paginas/Tres-de-cada-100-colombianos-tienen-diabetes.aspx

Doshi, S. (2019, January 13). Various optimization algorithms for training neural networks. Towards Data Science. https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Elmasri, R., & Navathe, S. B. (2019). Fundamentos de sistemas de bases de datos (7.ª ed.). Pearson Educación.

Gonzalez, J. M. C. (2020). Tuneando los hiperparámetros de una red neuronal LSTM para obtener un aprendizaje más eficiente. Linkedin.com. https://es.linkedin.com/pulse/tuneando-los-hiperpar%C3%A1metros-de-una-red-neuronal-lstm-casas-gonzalez

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems. https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

Hernández. R, (2016). Sistemas de monitorización continua de glucosa. Fundaciondiabetes.org. Retrieved March 28, 2023, from

https://www.fundaciondiabetes.org/general/articulo/173/sistemas-de-monitorizacion-continua-de-glucosa

Higuera, V. (2021, June 5). Insulina: Qué es, cómo funciona y más. Healthline. https://www.healthline.com/health/es/insulina

Hodeghatta, UR, Nayak, U. (2017). Aprendizaje automático supervisado: clasificación. En: Business Analytics usando R: un enfoque práctico. Apress, Berkeley, CA. https://doi-org.bibliotecavirtual.uis.edu.co/10.1007/978-1-4842-2514-1_6

Huang, Z., Yang, F., Xu, F., Song, X., & Tsui, K.-L. (2019). Convolutional gated recurrent unit–recurrent neural network for state-of-charge estimation of lithium-ion batteries. IEEE Access: Practical Innovations, Open Solutions, 7, 93139–93149. https://doi.org/10.1109/access.2019.2928037

International diabetes federation - home. (n.d.). Idf.org. Retrieved March 28, 2023, from https://idf.org/

Joshi, T. N., & Pramila, M. Chawan (2018). Diabetes prediction using machine learning techniques. https://doi.org/10.9790/9622-0801020913

Keays, R. (2007). Diabetes. Current Anaesthesia and Critical Care, 18(2), 69–75. https://doi.org/10.1016/j.cacc.2007.03.007

Keays, R. (2007). Diabetes. Current Anaesthesia and Critical Care, 18(2), 69–75. https://doi.org/10.1016/j.cacc.2007.03.007

Khan. (2022). Machinelearningmastery. https://machinelearningmastery.com/using-optimizers-from-pytorch/

Kostadinov, S. (2017, December 16). Understanding GRU networks. Towards Data Science. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Kostadinov, S. (2017, December 16). Understanding GRU networks. Towards Data Science. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Kumar, A. (2022, May 28). Activation functions in neural networks: Concepts. Data Analytics. https://vitalflux.com/different-types-activation-functions-neural-networks/

Malingan, N. (2023, February 22). Gated Recurrent Unit (GRU). Scaler Topics. https://www.scaler.com/topics/deep-learning/gru-network/

Man, C. D., Micheletto, F., Lv, D., Breton, M., Kovatchev, B., & Cobelli, C. (2014). The UVA/PADOVA type 1 Diabetes Simulator: New features. Journal of Diabetes Science and Technology, 8(1), 26–34. https://doi.org/10.1177/1932296813514502

ML. (2019, February 15). GeeksforGeeks. https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/

Mughees, A., Sharma, A., Ansari, S. S., & Ibrahim, S. M. (2023). Prediction of the compressive strength of nano-titanium based concrete composites using machine learning. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2023.03.540

Musstafa. (2021, March 27). Optimizers in deep learning - MLearning.Ai - medium. MLearning.Ai. https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0

Narmadha, S., Gokulan, S., Pavithra, M., Rajmohan, R., & Ananthkumar, T. (2020). Determination of various deep learning parameters to predict heart disease for diabetes patients. 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), 1–6.

OPS/OMS. (n.d.). Paho.org. Retrieved March 28, 2023, from https://www.paho.org/es

Organización mundial de la salud (OMS). (2005). En Anuario de las Naciones Unidas 2005 (págs. 1572–1573). NACIONES UNIDAS.

Ponce, I. G. (2009, February 18). Diabetes. CuidatePlus. https://cuidateplus.marca.com/enfermedades/medicina-interna/diabetes.html

Saravanan, R., & Sujatha, P. (2018). A state of art techniques on machine learning algorithms: A perspective of supervised learning approaches in data classification. 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), 945–949.

Suthaharan, S. (2016). Algoritmos de aprendizaje supervisado. En: Modelos y algoritmos de aprendizaje automático para la clasificación de Big Data (pp. 183–206). Serie integrada en sistemas de información, vol 36. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7641-3_8

Suykens, J. A. K., & Vandewalle, J. P. (1999). Least squares support vector machine classifiers. Neural processing letters, 9(3), 293-300. https://doi.org/10.1023/A:1018628609742

Tegara. (2021, June 9). Simulation software for research and development dealing with type 1 diabetes “UVA / Padova T1DMS.” TEGAKARI. https://www.tegakari.net/en/2021/06/t1dms/

Velásquez, S., Velásquez, R., Leyton, M., Borjas, J., & Custodio, Á. (2013). Modelado del control de la regulación de Glucosa. Universidad, ciencia y tecnología, 17(66), 11–18. http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S1316-48212013000100002

Viloria, A., Herazo-Beltran, Y., Cabrera, D., & Pineda, O. B. (2020). Diabetes diagnostic prediction using vector support machines. Procedia Computer Science, 170, 376–381. https://doi.org/10.1016/j.procs.2020.03.065

Viloria, A., Herazo-Beltran, Y., Cabrera, D., & Pineda, O. B. (2020). Diabetes diagnostic prediction using vector support machines. Procedia Computer Science, 170, 376–381. https://doi.org/10.1016/j.procs.2020.03.065

Visentin, R., Campos-Náñez, E., Schiavon, M., Lv, D., Vettoretti, M., Breton, M., Kovatchev, B. P., Dalla Man, C., & Cobelli, C. (2018). The UVA/Padova Type 1 diabetes simulator goes from single meal to single day. Journal of Diabetes Science and Technology, 12(2), 273–281. https://doi.org/10.1177/1932296818757747

Wang, W., Tong, M., & Yu, M. (2020). Blood glucose prediction with VMD and LSTM optimized by improved particle swarm optimization. IEEE Access: Practical Innovations, Open Solutions, 8, 217908–217916. https://doi.org/10.1109/access.2020.3041355

What is diabetes? (2013). Idf.org. Retrieved March 28, 2023, from https://www.idf.org/aboutdiabetes/what-is-diabetes.html

Zhang, Y., Ning, Y., & Huan, Z. (2021). An intelligent Attentional-GRU-based model for dynamic blood glucose prediction. 2021 2nd International Conference on Artificial Intelligence and Education (ICAIE), 10–14.

Zhang, Y., Ning, Y., Li, B., Liu, Y., & Dang, Y. (2021). Short-term prediction for dynamic blood glucose trends based on ARIMA-LSSVM-GRU model. Journal of Physics. Conference Series, 2030(1), 012057. https://doi.org/10.1088/1742-6596/2030/1/012057

Zhu, T., Yao, X., Li, K., Herrero, P., & Georgiou, P. (2020, January). Blood glucose prediction for type 1 diabetes using generative adversarial networks. In CEUR Workshop Proceedings (Vol. 2675, pp. 90-94).

Zhu, T., Yao, X., Li, K., Herrero, P., & Georgiou, P. (s.f.). Blood glucose prediction for type 1 diabetes using generative adversarial networks. https://ceur-ws.org/Vol-2675/paper15.pdf

Apéndices

Apéndice A. Código fuente y resultados, se puede descargar y revisar el repositorio en GitHub con el siguiente link.

https://github.com/ManuelN12/Prediccion-de-eventos-de-hiper-e-hipoglucemia-en-pacientes-con-diabetes-mediante-redes-neuronales.git