PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Escuela de Ingeniería Eléctrica
PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

Paulo Eduardo Granados Núñez

Pronóstico de series temporales utilizando Deep Learning con aplicación en fusión nuclear

Informe Proyecto de Título de Ingeniero Civil Electrónico

Escuela de Ingeniería Eléctrica
Facultad de Ingeniería

Valparaíso, 3 de enero de 2019

PONTIFICIA
UNIVERSIDAD
CATÓLICA DE
VALPARAÍSO

Pronóstico de series temporales utilizando Deep Learning con aplicación en fusión nuclear

Paulo Eduardo Granados Núñez

Informe Final para optar al título de Ingeniero Civil Electrónico,

aprobada por la comisión de la

Escuela de Ingeniería Eléctrica de la

Pontificia Universidad Católica de Valparaíso

conformada por

Sr. Gonzalo Farías Castro

Profesor Guía

Sr. Gabriel Hermosilla Vigneau

Segundo Revisor

Sr. Sebastián Fingerhuth Massmann

Secretario Académico

Valparaíso, 3 de enero de 2019

A mis padres.

Resumen

La fusión nuclear se ha presentado como la energía del futuro debido a su abundante materia prima y su muy reducido impacto en el medio ambiente. Esta energía se da naturalmente en el centro del Sol, haciendo realmente complicado replicar las condiciones necesarias para aprovechar dicha energía para el consumo de la humanidad. El principal problema de implementar la generación de energía por fusión nuclear es la disrupción, fenómeno asociado a la pérdida de estabilidad del plasma y que puede significar daños irreparables a los dispositivos de generación.

Existen dispositivos dedicados al estudio de la fusión nuclear y su posible implementación. Estos dispositivos generan una cantidad gigantesca de información por cada campaña de experimentación, la cual es prácticamente imposible de analizar por un usuario. Se plantea entonces un método de aprendizaje automático para colaborar en la realización de esta importante tarea.

El aprendizaje automático es una rama de la ingeniería que tiene como objetivo principal lograr que las máquinas aprendan. El Deep Learning es un tipo de aprendizaje automático capaz de crear modelos de redes neuronales profundos. Esto permite obtener características abstractas difíciles de observar por una persona.

Este proyecto contempla el uso de estas redes neuronales artificiales con el enfoque de Deep Learning. Las redes neuronales elegídas fueron las recurrentes, en particular las Long Short-Term Memory (LSTM) y Gated Recurrent Unit (GRU), las cuales son ideales para trabajar con secuencias de información.

Se plantean dos tareas importantes: el pronóstico de series temporales y la clasificación de descargas disruptivas y no disruptivas. Para ambos casos se implementarán modelos basados en las redes LSTM y GRU con el objetivo principal de lograr predecir datos futuros en una secuencia y decidir si estos datos son seguros o disruptivos. De esta forma se pueden tomar acciones de evitación y mitigación en caso de predecir un evento disruptivo.

Los resultados obtenidos para el pronóstico de series temporales fueron bastante acertados, logrando tasas de acierto de 99 % en el mejor de los casos. Para la clasificación de descargas entre disruptivas y no disruptivas, por otro lado, los resultados no fueron muy alentadores, puesto que se logró un 44 % de alarmas válidas en el mejor de los experimentos, dejando en evidencia la necesidad de mayor profundización en el problema.

Palabras claves: fusión nuclear, disrupción, series temporales, Deep Learning, LSTM, GRU, pronóstico, clasificación.

Abstract

Nuclear fusion has been presented as the energy of the future due to its abundant raw material and its very reduced impact on the environment. This energy occurs naturally in the center of the Sun, making it really complicated to replicate the necessary conditions to take advantage of this energy for the consumption of humanity. The main problem of implementing the generation of energy by nuclear fusion is the disruption, phenomenon associated with the loss of stability of the plasma and that can mean irreparable damage to the generation devices.
here are devices dedicated to the study of nuclear fusion and its possible implementation. These devices generate a gigantic amount of information for each experimentation campaign, which is practically impossible for a user to analyze. A machine learning method is then created to collaborate in the accomplishment of this important task.
Machine learning is a branch of engineering whose main objective is to get machines to learn. Deep Learning is a type of machine learning capable of creating models of deep neural networks. This allows to obtain abstract characteristics difficult to observe by a person.
This project contemplates the use of these artificial neural networks with the approach of Deep Learning. The neural networks chosen were the recurrent ones, in particular the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which are ideal for working with information sequences.
Two important tasks are posed: the forecast of time series and the classification of disruptive and non-disruptive discharges. For both cases, models based on the LSTM and GRU networks will be implemented with the main objective of predicting future data in a sequence and deciding if this data is safe or disruptive. In this way, avoidance and mitigation actions can be taken in case of predicting a disruptive event.
The results obtained for the forecast of time series were quite successful, achieving success rates of 99 % in the best of cases. For the classification of discharges between disruptive and non-disruptive, on the other hand, the results were not very encouraging, since 44 % of valid alarms were achieved in the best of the experiments, leaving in evidence the need for further deepening in the problem.

Key words: nuclear fusion, disruption, time series, Deep Learning, LSTM, GRU, forecast, classification.

Índice general

Introducción

La demanda energética ha crecido considerablemente y lo seguirá haciendo, dejando la necesidad de crear nuevas fuentes de generación que vayan de la mano con el cuidado del medio ambiente. En este contexto nace la idea de generación de energía por medio de fusión nuclear, la cual es considerada como la fuente de energía del futuro. La energía por fisión nuclear, actualmente vigente, utiliza uranio como materia prima y genera residuos radioactivos que demoran años en degradarse. Por su parte, la fusión nuclear promete una generación de energía más limpia y con requerimientos sencillos de materia prima, ya que utiliza hidrógeno, abundante en la tierra, y genera residuos de helio, no contaminante ni radioactivo. Si bien es una fuente de energía no renovable, se considera que es sustentable y abundante en producción.

El principal problema radica en la dificultad de mantener un entorno donde la fusión nuclear pueda realizarse. Esta energía se da naturalmente en el sol, a temperaturas y presiones extremadamente elevadas, complicadas de replicar en un laboratorio. Sin embargo, existen dispositivos nucleares que se dedican a realizar esta tarea. Su objetivo principal es llevar la materia al cuarto estado, llamado plasma. Una vez alcanzado este estado, por medio de confinamiento magnético, se puede aprovechar la energía producto de la fusión nuclear. El gran problema de mantener este estado de la materia es la disrupción.

La disrupción nuclear, fenómeno asociado a la falta de estabilidad en el plasma, es la mayor problemática que enfrenta la generación de energía por fusión nuclear. Una disrupción se genera cuando se pierde el confinamiento del plasma en el dispositivo. Este plasma choca contra las paredes del dispositivo generando daños a la estructura, junto con pérdidas de dinero y tiempo. Cada vez que se realiza un ensayo se corre el riesgo de generar una descarga disruptiva, razón por la cual se almacena la mayor cantidad de datos en cada experimento para poder comprender de mejor manera el comportamiento del plasma.

Esta cantidad de datos es inmensa e imposible de analizar manualmente por una sola persona de forma eficiente. Se propone, entonces, un sistema inteligente capaz de pronosticar y clasificar eventos disruptivos en un dispositivo de fusión nuclear. Es por esto que se buscan implementar métodos para poder pronosticar y clasificar eventos disruptivos en un dispositivo de fusión nuclear por medio de algoritmos de aprendizaje automático y Deep Learning.

Para el aprendizaje automático se contempla el uso de redes neuronales artificiales, las cua-

les imitan el comportamiento biológico de neuronas para poder aprender, llevando dichos comportamientos a términos matemáticos y computacionales. Las redes neuronales que se utilizarán serán las recurrentes, en particular la Long Short-Term Memory (LSTM) y Gated Recurent Unit (GRU), por su especial rendimiento en el análisis de series temporales.

Objetivos generales

- Diseñar e implementar algoritmos de aprendizaje automático en dispositivos de fusión nuclear con Deep Learning.

Objetivos específicos

- Estudiar el enfoque Deep Learning en aprendizaje automático.
- Estudiar los principales problemas asociados a la fusión nuclear.
- Implementar un sistema de aprendizaje automático con Deep Learning para un dispositivo de fusión nuclear real.

1 Energía Nuclear

1.1 Demanda energética

Si nos ponemos a pensar sobre la demanda energética que tiene el mundo hoy en día, resulta evidente notar que ésta va en aumento, sobre todo la energía eléctrica, tal como se aprecia en la figura 1.1. Dar abasto al sector industrial es cada vez un tema más complicado, sin mencionar el constante crecimiento de la población. Ejemplo de esto son los proyectos de plantas hidroeléctricas en el sur de nuestro país que tienen como finalidad abastecer a la minería del norte. Por razones ambientales y sociales algunos de estos proyectos importantes no fueron llevados a cabo, dejando un déficit energético a la zona industrial.

Si bien existen muchas formas de generar energía eléctrica, las más rentables generan un problema ambiental importante. Las plantas termoeléctricas generan emisiones de carbono sumamente dañinas para la atmósfera, entre otras desventajas. Las fuentes de energía renovable han evolucionado bastante, y cada vez son más accesibles, representando una alternativa para suplir a los métodos convencionales. Sin embargo, a pesar que producen energía limpia, son costosas y no dan abasto para satisfacer las necesidades energéticas.

Por estas razones es necesario investigar nuevas formas de generación de energía. Se introduce, entonces, la energía nuclear como fuente de generación.

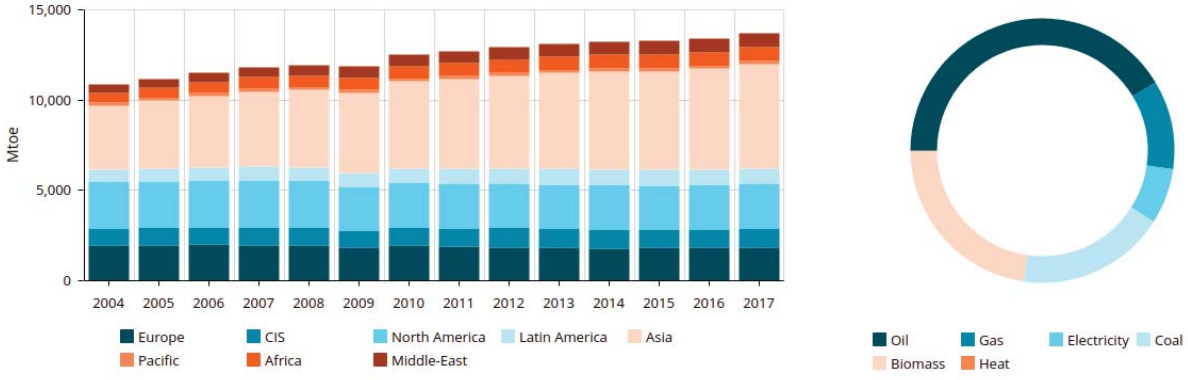


Figura 1.1: Demanda energética [1]

1.2 Fisión nuclear

La fisión nuclear es la reacción en la que el núcleo de un átomo pesado (normalmente uranio), al capturar un neutrón incidente, se divide en dos o más núcleos de átomos más ligeros, llamados productos de fisión, emitiendo en el proceso neutrones, rayos gamma y grandes cantidades de energía.

El núcleo que captura el neutrón incidente se vuelve inestable y, como consecuencia, se produce su división en fragmentos más ligeros dando lugar a una situación de mayor estabilidad. Además de estos productos, en la reacción de fisión se producen varios neutrones que al incidir sobre otros núcleos fisionables desencadenan más reacciones de fisión que a su vez generan más neutrones. Este efecto multiplicador se conoce como reacción en cadena y se ve ilustrado en la figura 1.2.

Para que se produzca una reacción de fisión en cadena es necesario que se cumplan ciertas condiciones de geometría del material fisionable y se supere un umbral determinado de cantidad del mismo, conocido como masa crítica. La fisión puede llegar a producirse de forma espontánea, pero es necesaria la existencia de un neutrón que incida con la energía adecuada. [2]

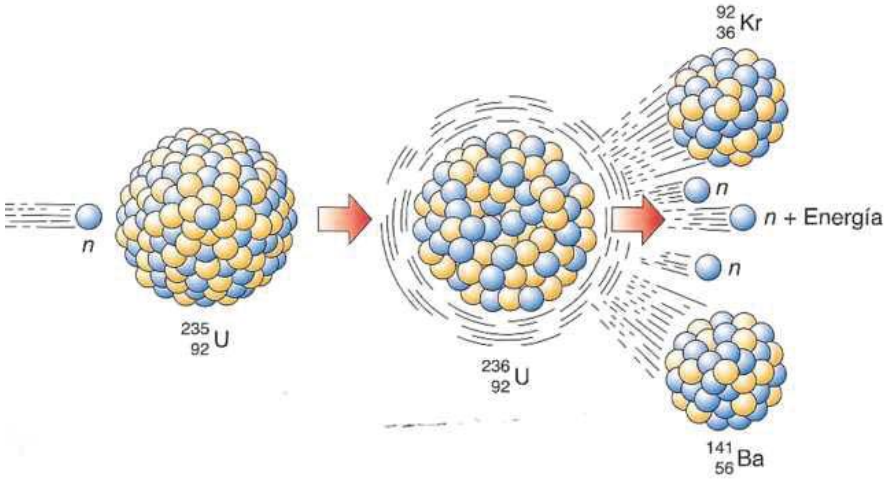


Figura 1.2: Esquema simplificado de la fisión nuclear [3]

1.3 Fusión nuclear

La fusión nuclear es una reacción nuclear en la cual se fusionan núcleos ligeros, normalmente isótopos del hidrógeno (Deuterio y Tritio), generando liberación de energía. Un diagrama sencillo es mostrado en la figura 1.3. Se da naturalmente en el Sol, por lo que reproducir este proceso de manera artificial significa manejar temperaturas y presiones similares a las presentes en el centro de nuestra estrella. Esta mezcla de temperatura y presión llevan a la materia a su cuarto estado: el plasma. Es necesario entonces crear y confinar plasma. Existen

distintos dispositivos de fusión nuclear que intentan confinar este plasma por medio de campos magnéticos, logrando separarlo de la pared contenedora. El proyecto ITER se presenta como el más importante para la generación de energía por fusión nuclear.

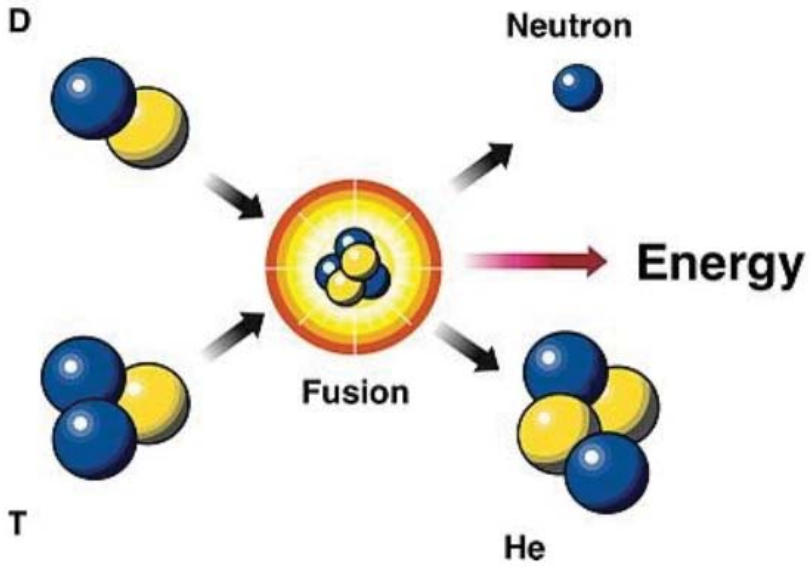


Figura 1.3: Esquema simplificado de la fusión nuclear [3]

1.4 Dispositivos de fusión nuclear

1.4.1 Tokamak JET

Un Tokamak es un reactor termonuclear por confinamiento magnético de creación rusa. Tiene forma de cámara toroidal y es un tubo hueco, rodeado exteriormente por bobinas que se encargan del confinamiento del plasma. Fue ideado por los físicos soviéticos Ígor Tam y Andréi Sájarov en 1950 [4]. La componente toroidal del campo magnético se genera mediante bobinas externas, haciendo que el embobinado primario induzca el campo sobre el plasma y utilizándolo este mismo como un embobinado secundario. El mayor reactor de este tipo es el Tokamak JET (Joint European Torus) y ha conseguido condiciones de fusión nuclear con factor Q >0.7 [4], es decir que la relación entre la energía generada por fusión y la energía requerida para sostener la reacción es 0.7 (para que el sistema sea viable claramente se debe alcanzar un factor de Q >1).

1.4.2 Stellarator

Los Stellarator fueron los primeros dispositivos de confinamiento magnético diseñados. Al igual que los reactores Tokamak, funcionan confinando el plasma mediante campos magnéticos a una temperatura elevada. Se diferencian principalmente en que el Stellarator puede

operar de forma continua. Actualmente se están haciendo estudios sobre la simplicidad técnica que puede brindar para una planta de fusión nuclear.

1.4.3 Proyecto ITER

ITER será la próxima gran instalación de tipo tokamak. Es uno de los proyectos más ambiciosos en el mundo hoy en día en relación a la generación de energía. ITER será el primer dispositivo de fusión nuclear en mantener la fusión por largos periodos de tiempo, con un factor Q >10, además de ser el primer dispositivo en testear tecnologías, materiales y régimen físicos necesarios para la producción comercial de energía eléctrica basada en fusión. Sus principales objetivos [5] son producir 500 MW de potencia de fusión, demostrar el funcionamiento integrado de las tecnologías para una planta de energía de fusión, alcanzar un nivel de plasma, deuterio-tritio, que se mantenga mediante calentamiento interno, probar la capacidad de producir tritio y demostrar las características de seguridad de un dispositivo de fusión.

2 Problemática y Motivación

La fusión nuclear se presenta como la fuente de generación ideal gracias a su mínimo impacto ambiental y a su abundante materia prima. Sin embargo esta fuente de energía aún no es viable pues el principal problema tiene relación con la inestabilidad del plasma, o también llamada disrupción nuclear.

2.1 Disrupción nuclear

La disrupción nuclear se puede entender como una pérdida de estabilidad en el confinamiento magnético. Es un fenómeno poco deseable que involucra la inestabilidad magnetohidrodinámica (dinámica de fluidos conductores de electricidad en presencia de campos eléctricos y magnéticos) del plasma, lo que se traduce en una rápida pérdida de calor, conllevando una descarga que puede tener consecuencias destructivas en la infraestructura. Por lo tanto, es importante poder contar con un sistema que prediga la aparición de estas disrupciones, para así poder evitar o reducir los daños asociados que pueden resultar sumamente costosos en tiempo y dinero. Algunos de estos daños se pueden ver en la figura 2.1.

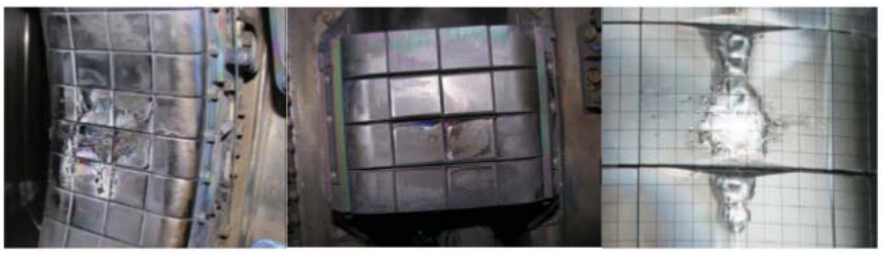


Figura 2.1: Daños en la pared del dispositivo.

2.2 Mitigación y evitación

Es necesario, entonces, encontrar estrategias que logren la evitación y la mitigación de los fenómenos disruptivos descritos anteriormente. La idea principal es que estas estrategias actúen antes de la primera descarga del plasma.

La evitación se realiza al inicio de la disrupción. Las opciones son: reparar y continuar con el experimento, retroceso y apagado del dispositivo (proceso normalmente utilizado) y “aterriza-

je suave" (parada rápida) del experimento. Es necesario tener en cuenta las diferencias entre los tiempos de recuperación y reinicio para poder realizar cualquier tipo de evitación. Además existe la posibilidad de inyección lenta de gas o de pellets.

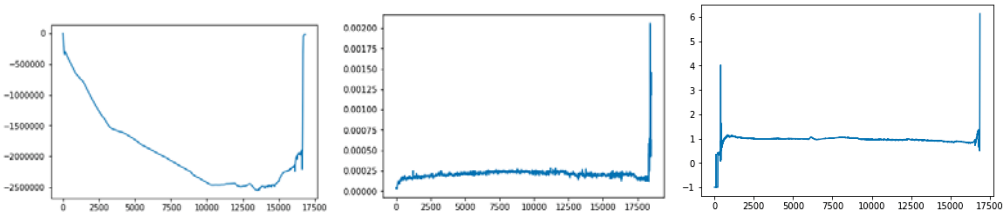
La mitigación se aplica cuando fracasa la tarea de evitación, conllevando complejas consecuencias. El método candidato para la mitigación es la inyección masiva de gas, que puede incluso re-evitar la disrupción. Una de las desventajas notables de los métodos mencionados son el agotamiento del sistema tras el impacto del gas y la acumulación de polvo.

2.3 Datos de información

Para el desarrollo de este trabajo fue necesario manejar información real proveniente del dispositivo Tokamak JET. Estos datos fueron extraídos de campañas de experimentación, las cuales están conformadas por descargas realizadas en el dispositivo. Cada descarga contiene señales que monitorean el comportamiento del plasma. Estas descargas están agrupadas en dos principales categorías: descargas disruptivas y no disruptivas. Dentro de las descargas disruptivas están las intencionales y las no intencionales. Se considera que una descarga es disruptiva cuando se produce un evento disruptivo durante la realización de ese experimento, etiquetando a todas las señales como disruptivas. De la misma forma, las descargas no disruptivas son aquellas que no producen ningún evento disruptivo, catalogando todas las señales como no disruptivas. Las descargas disruptivas intencionales son aquellas en las que la disrupción es forzada por los que realizan el experimento. La magnitud de cada señal es registrada en el tiempo, y duran hasta que termina el experimento o bien se produzca un evento disruptivo.

2.3.1 Base de datos JET

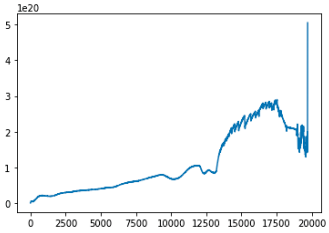
De las campañas realizadas en el Tokamak JET y facilitada por el Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), se obtuvo una base de datos con descargas disruptivas y descargas no disruptivas, con 7 señales de interés, las cuales son señaladas en la tabla 2.1. Las campañas a considerar son la C2830 y la C36. La campaña C2830 cuenta con 201 descargas disruptivas, 56 descargas disruptivas intencionales y 1036 descargas no disruptivas. La campaña C36 cuenta con 151 descargas disruptivas y 995 descargas no disruptivas. Todas las señales de estas descargas están muestreadas a 1[ms].



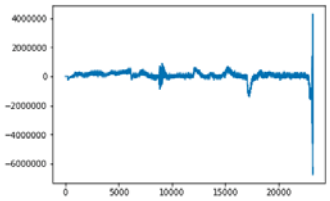
(a) Plasma current.

(b) Mode Lock.

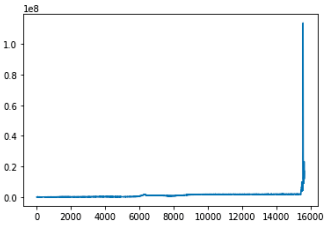
(c) Plasma internal inductance.



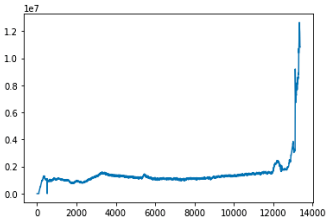
(d) Plasma electronic density.



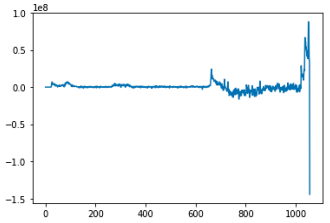
(e) Diamagnetic energy derivative.



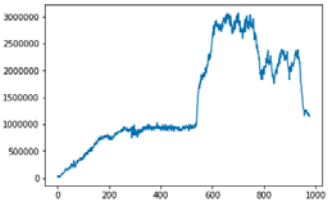
(f) Radiated power.



(g) Total input power.



(h) Temporal evolution of W influx level.



(i) Diamagnetic energy.

Figura 2.2: Señales de interés JET disruptivas.

2.4 Inteligencia artificial

Los dispositivos experimentales de fusión nuclear generan una gran cantidad de datos que intentan monitorear el comportamiento del plasma. Estas bases de datos obtenidas están compuestas por gran cantidad de descargas, que corresponden a experimentos realizados. Estas descargas están compuestas por un determinado número de señales que a su vez poseen una cantidad importante de datos, generando bases de datos de Gigas de información relativa al comportamiento del plasma. Este gran volumen de información no puede ser analizado de forma eficiente por una persona y se necesitan sistemas automatizados que puedan manejar tal información. Es por esto que se introduce la inteligencia artificial como un método para el análisis de estas grandes cantidades de información.

La inteligencia artificial es un campo que crece rápidamente y que ha demostrado resolver de forma eficiente problemas que serultan ser complicados para los humanos. La gran complejidad está en lograr que la inteligencia artificial aprenda a desarrollar y describir problemas de forma intuitiva y automática. El aprendizaje automático es una rama de la inteligencia artificial que permite a las computadoras aprender. En el contexto de las descargas de fusión

Tabla 2.1: Señales de interés JET

Señal	Nombre de la señal	Descripción
1	Plasma current	Intensidad de corriente de plasma
2	Mode Lock	Densidad de flujo magnético por Ampere
3	Plasma internal inductance	Inductancia interna del plasma
4	Plasma electronic density	Densidad electrónica del plasma
5	Diamagnetic energy derivate	derivada de energía diamagnética
6	Radiated power	Potencia radiada
7	Total input power	Potencia total de entrada
8	Temporal evolution of W influx level	Evolución temporal de la afluencia de W
9	Diamagnetic energy	Energía diamagnética

nuclear se refiere a identificar patrones complejos en gran cantidad de datos, y esto lo hace a través de algoritmos que revisan los datos, de esta forma la computadora es capaz de predecir comportamientos futuros. Estos algoritmos, si son bien implementados, pueden descubrir cómo realizar tareas generalizando a partir de ejemplos. Esto es muy útil en contraste a la construcción de estos programas de forma manual, haciendo de estos métodos un método factible y rentable donde la programación manual no lo es. Mientras más datos se posean, más problemas se pueden abordar, y en la última década el uso de aprendizaje automático se ha esparcido rápidamente más allá de la ciencia computacional, en parte gracias al gran desarrollo de la tecnología y a la capacidad de cómputo de los computadores modernos.

En el tiempo actual disponemos de gigantescas bases de datos con gran cantidad de señales de dispositivos de fusión que pueden ser totalmente analizados por algoritmos de inteligencia artificial. Estos pueden hacer uso de estos datos para lograr pronosticar varias series temporales en un periodo determinado de tiempo. Gracias a la posibilidad de abstracción de estas inteligencias es posible encontrar patrones, características o comportamientos que pueden describir la dinámica de las señales durante una disrupción para lograr estrategias de evitación y mitigación.

2.4.1 Series temporales

Una serie temporal se define como una secuencia de datos medidos en determinados momentos y ordenados cronológicamente. Al no seguir un modelo matemático en particular (aparentemente), se hace útil la aplicación de redes neuronales recurrentes para predecirlas. Los datos provenientes de los distintos experimentos realizados en los dispositivos de fusión nuclear son una serie de datos listados en un orden temporal, a un tiempo de muestreo igual para cada dato.

2.4.2 Reconocimiento de patrones

El reconocimiento de patrones un proceso que detecta e identifica estructuras ordenadas en datos. Estos patrones se obtienen del análisis de datos a partir de procesos de segmentación y extracción de características, con el objetivo de obtener información que permita establecer un conjunto de reglas que describan a los datos analizados. La primera etapa consiste en la entrada de datos que representan algún elemento físico, por medio de un sensor. Después, por medio de la extracción de características según un patrón de representación, se extrae la información relevante y se desecha aquella sin utilidad. Finalmente se tiene un clasificador encargado de etiquetar la información de entrada en una categoría de acuerdo a los patrones establecidos. Este proceso se muestra en la figura 2.3.

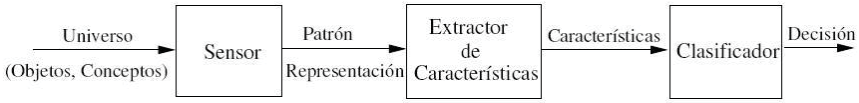


Figura 2.3: Reconocimiento de patrones.

Para lograr el modelo óptimo se debe analizar un cierto número de características para poder clasificar de manera satisfactoria los datos de entrada. Estos sistemas deben tener en cuenta la variabilidad de los datos, como lo son el ruido, cambio de escalas o modificaciones realizadas a las secuencias de entrada.

2.4.3 Pronóstico y clasificación

Para lograr prevenir eventos disruptivos y tomar acciones de evitación y mitigación será necesario poder pronosticar con cierto grado de acierto ventanas futuras de datos, y a la vez determinar si estos datos pronosticados tienen un carácter disruptivo o no, para así poder clasificar dicha señal y lanzar la alarma correspondiente.

2.5 Redes neuronales artificiales

Este trabajo contempla el uso de redes neuronales artificiales, también llamadas Multicapas de Perceptrones. Un perceptrón es un modelo de una neurona, precursor de redes neuronales más complejas. Esta área de la ingeniería investiga cómo modelos simples de cerebros biológicos pueden ser usados para resolver problemas computacionales. La utilidad de las redes neuronales radica en su habilidad para aprender representaciones en la información de entrenamiento para relacionar de mejor manera nueva información.

Las neuronas artificiales son los bloques de construcción de las redes neuronales. Son unidades computacionales que tienen señales ponderadas de entrada, las cuales son sumadas y producen una salida de acuerdo a una función de activación. La figura 2.4 muestra una

comparación entre la comunicación entre neuronas biológicas y neuronas artificiales.

Figura 2.4: neurona biologica y artificial

2.5.1 Pesos de las Neuronas

Cada neurona tiene entradas que son ponderadas por un factor. A este factor se le llama peso y existe uno para cada entrada de la neurona. De forma similar, cada neurona tiene un *bias*, el cual puede pensarse como un coeficiente de posición. Este bias tiene el valor de 1 y también tiene su peso asociado.

Los pesos son seleccionados de forma aleatoria (basados en una distribución Gaussiana). Estos pesos iniciales tendrán valores entre 0 y 1, los cuales serán ajustados una vez comience el entrenamiento (descrito en la sección Entrenamiento) [6].

2.5.2 Función de Activación

Las entradas ponderadas son sumadas y pasadas a través de una función de activación, a veces llamada función de transferencia. Se llama función de activación porque gobierna el umbral en el que se activa la neurona y la intensidad de la señal de salida. Los primeros trabajos contemplaban una función de activación de paso, la cual arroja una salida de 0 o 1 de acuerdo a un umbral. Por ejemplo, si este umbral se fija en 0.5, todas las sumas ponderadas sobre este valor generarán una salida de 1. En caso contrario la salida será 0.

Tradicionalmente, las funciones de activación no lineales son utilizadas. Esto permite a la red combinar las entradas en formas más complejas y mejorar el rendimiento. Funciones no lineales como la logística también es llamada *función sigmoide*, la cual produce una salida entre 0 y 1 de acuerdo a una distribución *en forma de s*. La función Tanh (tangente hiperbólica) genera la misma distribución, pero entre -1 y 1. Recientemente la función de activación rectificadora (RELU) ha demostrado mejores resultados [7].

2.5.3 Red de Neuronas

Las neuronas son organizadas en redes de neuronas. Una columna de neuronas se llama *capa*, y una red neuronal puede tener múltiples capas. Un esquema sencillo se muestra en la figura 2.5:

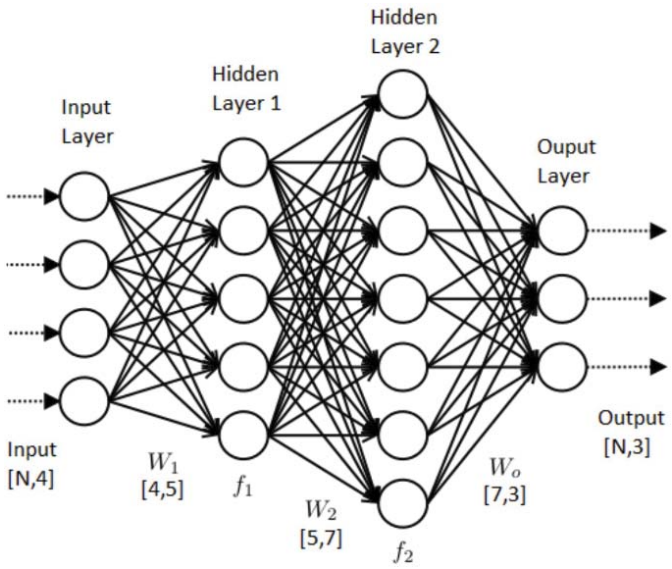


Figura 2.5: Ejemplo de red neuronal

Existen 3 tipos básicos de capas: capa de entrada, capa oculta y capa de salida.

2.5.3.1 Capa de Entrada

Es la capa que toma las entradas de la red neuronal. También es llamada capa visible porque es la parte expuesta de la red. Normalmente esta capa de entrada tiene tantas neuronas como entradas provenientes de la base de datos. Estas neuronas simplemente pasan la información a la siguiente capa.

2.5.3.2 Capas Ocultas

Las capas siguientes a la capa de entrada son llamadas capas ocultas ya que no están directamente expuestas a la información de entrada de la red. La estructura más sencilla es considerar una sola neurona en la capa oculta. Sin embargo, con la mejora de la tecnología computacional, redes neuronales muy profundas (es decir, con muchas capas ocultas) pueden ser construidas. Este es el fundamento del Deep Learning, el cual busca obtener mejores resultados basándose en modelos de redes neuronales más complejos (más capas ocultas) y con mayor capacidad de abstracción [8].

2.5.3.3 Capa de Salida

La última capa oculta es llamada *capa de salida* y es la responsable de generar el valor o vector de valores de salida que corresponden a los requerimientos del problema para el cual fue concebida la red neuronal. La elección de la función de activación para esta capa está fuertemente restringida al tipo de problema que se desea solucionar. Por ejemplo:

- Un problema de regresión puede tener una neurona de salida, la cual no está obligada a tener una función de activación.
- Un problema de clasificación binaria puede tener una neurona de salida con una funcion de activación sigmoide para que arroje un valor entre 0 y 1 para representar la probabilidad de predecir un valor. Se puede determinar un umbral que decida la predicción final.
- Un problema de multiclasificación puede tener múltiples neuronas en la capa de salida, una para cada clase. En este caso una función de activación softmax puede ser utilizada para arrojar la probabilidad para la predicción de cada clase. El resultado sería la clase con la mayor probabilidad.

En la figura 2.5 se puede apreciar una red neuronal con una capa de entrada de 4 neuronas, una primera capa oculta con 5 neuronas, una segunda capa oculta con 7 neuronas y una capa de salida con 3 neuronas. Este modelo podría representar un problema de clasificación de 3 clases.

2.5.4 Entrenamiento

Una vez definida la red neuronal, es necesario entrenarla con la información destinada a esta tarea.

2.5.4.1 Back Propagation

El algoritmo clásico de entrenamiento para redes neuronales es el llamado *Stochastic gradient descent*. Es cuando una columna de información es expuesta a la red como entrada. La red procesa esta entrada activando las neuronas y produciendo un valor de salida. A esto se le llama *forward pass* en la red. Esta es la misma forma en que se realizan predicciones con nuevos datos.

La salida de la red es comparada con el resultado esperado y se calcula el error. Este error es propagado hacia atrás a través de la red, una capa a la vez, y los pesos son actualizados acorde al valor que ellos contribuyen al error. A este algoritmo se le llama *Back Propagation*. Este proceso se repite para todos los ejemplos de la información de entrenamiento. Una ronda de actualización de pesos de la red para todo en conjunto de entrenamiento es llamado *época*. Dependiendo de la tarea a resolver, una red puede ser entrenada por decenas, centenas o miles de épocas.

2.5.4.2 Aprendizaje supervisado

En este algoritmo se usan datos que contienen características a través de un supervisor, y cada una asociada a una etiqueta. Esto hace que el algoritmo pueda aprender a clasificar diferentes problemas en base a esas características. Las tareas que caen dentro del aprendizaje supervisado son el reconocimiento de patrones (clasificación) y la regresión (aproximación de funciones), además se aplica a datos secuenciales. El aprendizaje supervisado se puede considerar aprendizaje con un "maestro", en el sentido que una función proporciona retroalimentación continua sobre la calidad de las soluciones obtenidas hasta el momento.

2.5.4.3 Aprendizaje no supervisado

En este algoritmo no existe un supervisor que controle el proceso de aprendizaje, se tiene mucha información que contiene muchas características y se aprende usando todas las propiedades de la base de datos, para luego aprender a clasificarlas. Es, por tanto, un método donde un modelo es ajustado en base a las observaciones. A diferencia del aprendizaje supervisado, no existe un conocimiento a priori. En el aprendizaje no supervisado, un conjunto de datos de objetos de entrada es analizado. Así, el aprendizaje no supervisado típicamente analiza los objetos de entrada como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos.

2.5.4.4 Predicciones

Una vez que la red está entrenada es posible realizar predicciones con nueva información. Normalmente se ocupa una parte de la base de datos para testear el modelo creado. La configuración de la red y el conjunto de pesos es todo lo que se necesita para poder utilizar un modelo. Las predicciones se realizan ingresando nueva información a la red y luego analizando el valor de salida.

2.6 Deep Learning

El Deep Learning, o aprendizaje profundo, es una rama del aprendizaje automático que contiene una serie de métodos o algoritmos que permiten a las máquinas aprender a través de ejemplos. Estos modelos de aprendizaje automático son entrenados con una gran cantidad de datos etiquetados con su categoría correspondiente, además de contar con arquitecturas de redes neuronales con muchas capas. Estas capas, conectadas en cascada, representan unidades de procesamiento no lineal que se van adaptando en el proceso de aprendizaje en términos de conceptos jerárquicos. Esta jerarquía de conceptos permite a la computadora aprender conceptos complicados al construirlos de conceptos más simples. Cada capa sucesiva usa la salida de la capa previa como entrada, este aprendizaje puede ser supervisado o no supervisado y contiene muchos niveles de representación dependiendo de los diferentes

niveles de abstracción (niveles de conceptos jerarquizados).

2.7 Redes neuronales recurrentes

Las redes neuronales recurrentes contienen lazos, permitiendo que se mantenga la información. Esta característica es especialmente útil cuando se pretende pronosticar series temporales. Un diagrama de una red neuronal recurrente se muestra en la figura 2.6.

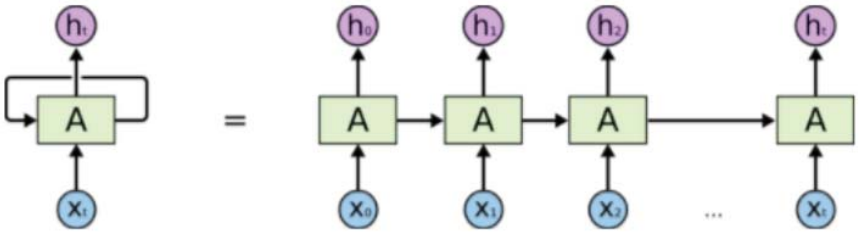


Figura 2.6: Red neuronal recurrente estándar [9].

El esquema anterior nos revela que las RNN se relacionan de mejor manera con las secuencias de datos. Sin ir más lejos, la gracia de estas redes es que pueden conectar información previa para una tarea presente. Sin embargo, no siempre se puede lograr este cometido. Cuando la brecha entre la información previa necesaria y la salida de la red es pequeña, la red puede aprender a ocupar esta información previa correctamente, tal como se ilustra en la figura 2.7

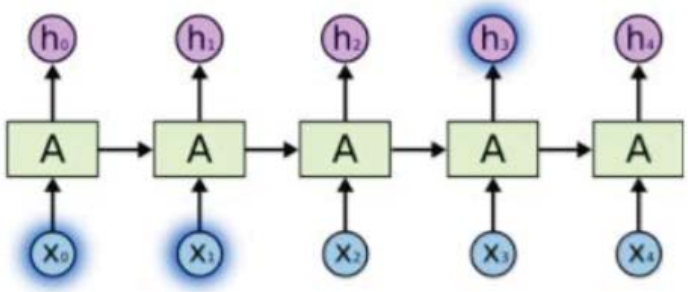


Figura 2.7: Brecha pequeña [6].

Sin embargo, hay casos donde es necesario más contexto, agrandando la brecha entre la información previa relevante y la salida correcta. Esto genera que la red neuronal recurrente sea incapaz de aprender a conectar la información, como se ejemplifica en la figura 2.8

El esquema de una red neuronal recurrente estándar se muestra en la figura 2.9.

Estas redes neuronales tienen que lidiar con el problema de las dependencias a largo plazo, es decir, deben ser capaces de conectar información que está mucho más atrás que el estado

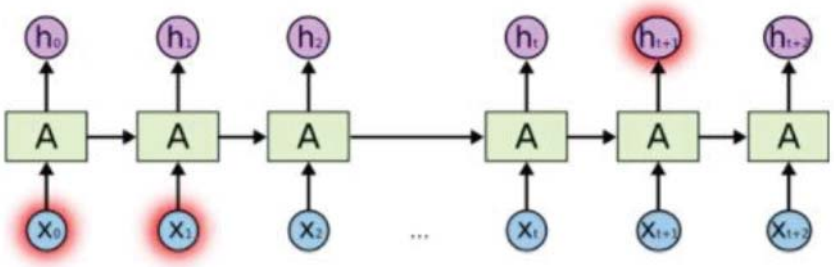


Figura 2.8: Brecha muy grande [6].

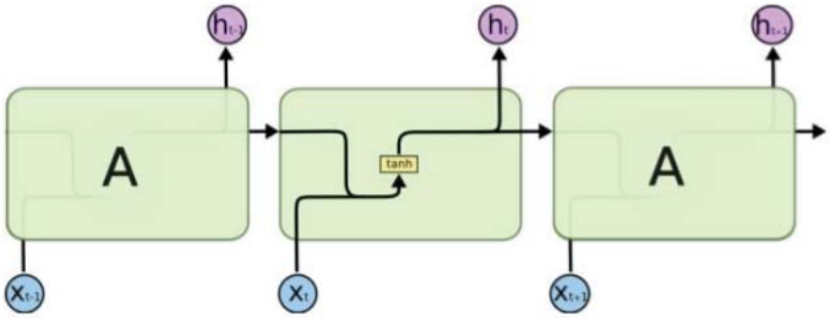


Figura 2.9: Red neuronal recurrente estándar [6].

actual. Esta tarea no siempre se consigue y es de este modo en que nacen las redes LSTM y GRU, las cuales pueden adaptarse al problema de las dependencias.

2.8 Long short-term memory

También llamadas redes LSTM, son un tipo especial de red neuronal recurrente, capaces de lidiar con el problema de las dependencias a largo plazo mencionado anteriormente. Son capaces de recordar información por periodos largos de tiempo, siendo una característica normal de ellas. Su esquema es similar al de una RNN estándar, pero con importantes modificaciones:

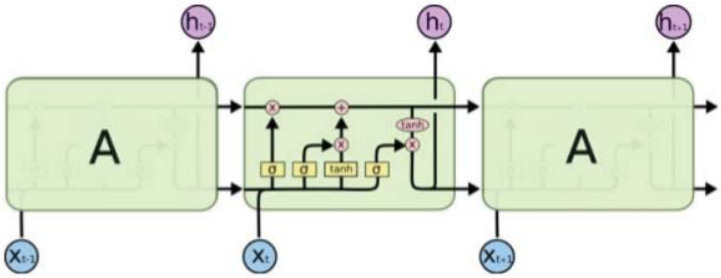


Figura 2.10: Red LSTM [6].

El módulo de una red LSTM tiene 4 capas, a diferencia del módulo de una RNN estándar (que tiene sólo una capa tanh).

Lo primero es la celda de estado (2.11a). Esta celda pasa a través de toda la cadena con pocas interacciones lineales, aunque la red puede agregar o remover información.

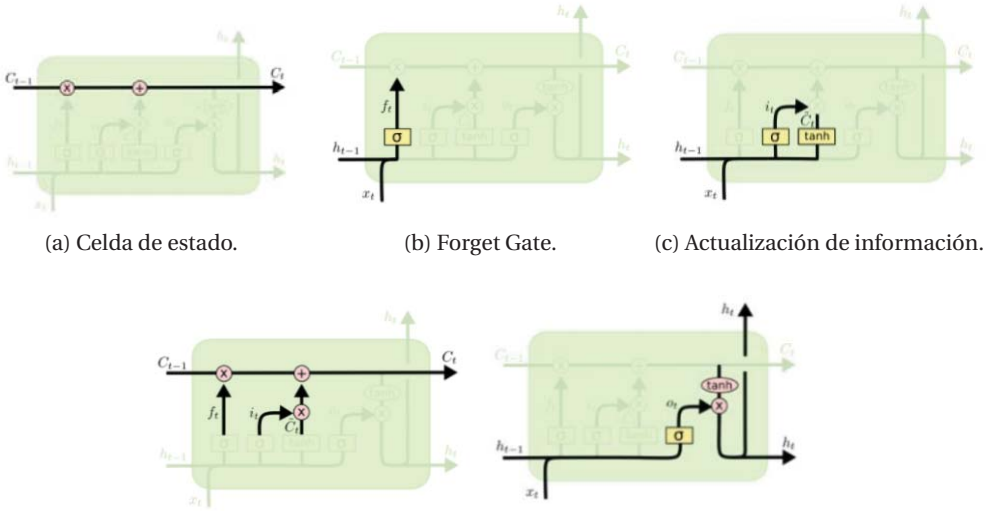
Después se debe decidir cuál información se desechará de la celda de estado. Esto se hace por la compuerta Forget Gate (2.11b), la cual es una capa con la función sigmoide. Esta función arroja un valor entre cero y uno, donde cero significa que no dejará pasar nada, y uno significa que dejará pasar todo hacia la celda de estado.

Luego tenemos que decidir cuál información guardaremos en la celda de estado. Esta etapa cuenta con dos partes. Primero, una capa sigmoide decide qué valores actualizaremos. Luego, una capa tanh (tangente hiperbólica) crea un vector de nuevos valores candidatos a ser actualizados, tal como se muestra en (2.11c).

Ahora debemos actualizar el estado antiguo de la celda, C_{t-1} , al nuevo estado C_t, como se muestra en la figura 2.11d. Primero multiplicamos el estado anterior por la salida de la compuerta Forget Gate, eliminando las cosas que queremos olvidar. Luego agregamos a la celda de estado el producto de las compuertas del segundo paso. Estos son los nuevos valores candidatos, ponderados de acuerdo a cuánto queremos que participen en la celda de estado.

Finalmente, debemos decidir cuál será nuestra salida. Esta salida será una versión filtrada de la celda de estado. Primero, una capa sigmoide decide qué partes de la celda de estado anterior dejaremos como salida. Luego, hacemos pasar la celda de estado por una capa tanh, para después multiplicar estas dos salidas y así dejar como salida final lo que hayamos decidido (2.11e).

La figura 2.11 muestra las etapas antes descritas.



(a) Celda de estado. (b) Forget Gate. (c) Actualización de información.

(d) Actualización de memoria. (e) Salida final.

Figura 2.11: Red neuronal recurrente LSTM [6].

2.9 Gated recurrent unit

También conocida como GRU, es una variación de la red SLTM. Esta red combina la compuerta Forget Gate con la Imput Gate en una sola compuerta llamada Update Gate (compuerta de actualización). También combina la celda de estado con el estado oculto, entre otras variaciones. Representa un modelo más simple que una red LSTM estándar, sin perder popularidad.

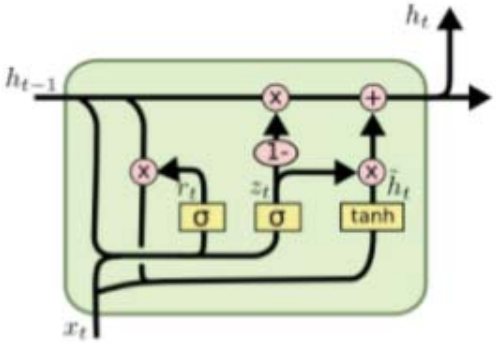


Figura 2.12: Red GRU [6].

Primero se realiza la activación de h_t , que corresponde a una interpolación lineal entre la activación previa h_{t-1} y el candidato de activación $\tilde{h}_t$. La actualización de la compuerta z_t decide cuánto actualiza la unidad de activación. Este procedimiento de tomar una suma lineal entre los estados existentes y el nuevo estado calculado es parecido al del bloque LSTM. Sin embargo, la GRU no tiene ningún mecanismo para controlar el grado al cual es expuesto el

estado, simplemente expone todo a la vez.

En resumen, una red neuronal recurrente simple presenta una multiplicación entre la entrada (Input x_t) y la salida anterior (Output h_{t-1}). Este resultado pasa a través de una compuerta con una función Tanh de activación. No existen celdas de estado.

En la Gated Recurrent Unit (GRU) se introduce una compuerta de actualización, la cual decide si pasar el Output anterior h_{t-1} a la compuerta siguiente o no pasarlo. La compuerta de olvido son simples operaciones matemáticas con un nuevo conjunto de pesos.

En los bloques Long-Short Term Memory (LSTM) se agregan dos compuertas más (Forget y Output), complementando la compuerta Update de la unidad GRU. Estas dos compuertas extras vienen con operaciones matemáticas con sus conjuntos de pesos respectivos.

Por lo tanto, la unidad GRU presenta mayor control en la red en comparación a una unidad recurrente simple, lo que se traduce en mayor control del flujo de información y mezcla con las entradas de la red, a medida que avanzan a través de los pesos de la misma. De la misma forma, las redes LSTM presentan aún más control sobre el sistema, presentando más parámetros para así darle más flexibilidad al control de la red global. Es necesario aclarar que al agregar más controlabilidad (y por ende, poder optar a mejores resultados) el sistema se hace más complejo y costoso en términos de operaciones computacionales.

Al ser ambas redes (GRU y LSTM) similares en complejidad en comparación a una red recurrente simple, es difícil determinar cuál de las dos se desempeñará de mejor manera a la hora de aplicarlas al problema propuesto. Es por esta razón que se utilizarán ambas redes y se analizarán los resultados obtenidos para decidir cuál es la que tiene mejor desempeño.

3 Implementación de algoritmos

En este capítulo se detalla el proceso de construcción de las distintas redes neuronales implementadas en este proyecto. En primer lugar, se especificará en detalle el proceso de pronóstico de series temporales, sus métricas de evaluación y los distintos experimentos que fueron llevados a cabo. En segundo lugar se explicará el proceso de clasificación de descargas disruptivas y no disruptivas.

3.1 Pronóstico de series temporales

La tarea de crear un modelo predictivo de series temporales es compleja ya que tienen la necesidad de analizar la información pasada en la secuencia de entrada. Las redes neuronales convencionales no son aptas para resolver este problema pues no tienen forma de considerar la variable del tiempo. Las redes neuronales recurrentes, por otra parte, sí tienen la habilidad de manejar la dependencia en la secuencia.

Las redes neuronales recurrentes contienen conexiones que tienen ciclos repetitivos, otorgándole realimentación y memoria a la red durante el tiempo. Esta característica importante las convierte en herramientas poderosas a la hora de de aprender y generalizar a través de secuencias de entrada. En particular se trabajará con las redes neuronales recurrentes LSTM y GRU, las cuales han resultado ser particularmente efectivas cuando se trabaja con series temporales.

La principal característica de las redes neuronales recurrentes de tipo LSTM o GRU es que la dependencia en los datos de entrada puede ser aprendido, y además no es necesario especificar un conjunto fijo de observaciones retardadas. Esto significa que la red puede aprender una dependencia temporal que varía según las circunstancias, como lo son las dinámicas presenten en las señales de fusión nuclear. Gracias a las capacidades descritas por estas redes neuronales recurrentes, se ha decidido probar estas redes para pronosticar el comportamiento futuro de los datos de una señal.

3.1.1 Rendimiento del pronóstico

Para la medición del rendimiento del pronóstico se utilizarán dos criterios, el promedio del error absoluto porcentual (MAPE) y la precisión. El error MAPE es un criterio de exactitud, es

decir, apunta a medir cuan cercanos son los valores pronosticados con respecto a sus valores reales y la precisión apunta a cuan repetible y reproducible es el pronóstico. La Figura 3.1 muestra con un ejemplo didáctico la diferencia entre exactitud y precisión.

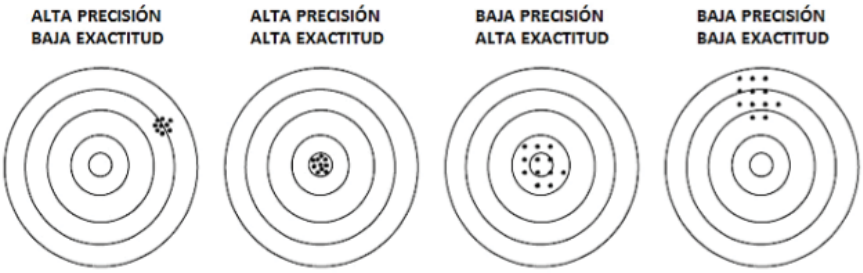


Figura 3.1: Exactitud y precisión.

Las ecuaciones 3.1 y 3.3 muestran cómo calcular el error MAPE y la precisión respectivamente, donde n es el número de muestras de los datos de la serie temporal, y_i es el valor real de la señal analizada e $\hat{y}_i$ es el valor pronosticado por la red.

$$MAPE\% = 100 \times \frac{1}{n} \times \sum_{i=0}^{n} |\frac{y_i - \hat{y}_i}{y_i}| \tag{3.1}$$

$$Exactitud\% = 100 - MAPE\% \tag{3.2}$$

$$\sigma\% = 100 \times \sqrt{\frac{1}{n} \times \sum_{i=0}^{n} (|\frac{y_i - \hat{y}_i}{y_i}| - MAPE)^2} \tag{3.3}$$

$$Precisión\% = 100 - \sigma\% \tag{3.4}$$

Para el pronóstico de series temporales se analizan 3 experimentos, estos son: análisis uno a uno, análisis de los últimos 2 segundos previos a la disrupción y el análisis considerando descargas completas.

3.1.2 Experimento uno a uno

Para este método de análisis se contempla estudiar cada una de las descargas, tanto safe como disruptivas, de forma individual, tomando en cuenta cada una de las 7 señales por

cada descarga. Para cada señal se toma el 60 % de la data para entrenar y el 40 % restante para testear.

3.1.3 Experimento 2 segundos previos a la disrupción

En esta oportunidad se trabajará con las descargas disruptivas no intencionales solamente. Lo que diferencia este análisis del anterior es que solamente serán considerados los últimos 2 segundos antes de la disrupción, y no la señal completa como en el caso anterior. Similarmente al caso anterior, se evaluarán todas las descargas disruptivas disponibles una a una.

3.1.4 Experimento considerando descargas completas

Para este análisis se propone considerar descargas completas para el entrenamiento, para luego testear con otras descargas, también completas. La diferencia radica en que esta vez no se tomará el 60 % de cada señal para entrenar y luego testear con el resto de la señal, sino que se tomarán el 60 % de las descargas para entrenar y se testeará con el 40 % restante de las descargas.

Este mismo procedimiento se aplicará a las descargas disruptivas, considerando solamente los últimos 2 segundos antes de la disrupción.

3.2 Clasificación de señales

La clasificación de descargas disruptivas y no disruptivas es una tarea complicada de resolver. Esta dificultad se debe, en parte, a la gran aleatoriedad que existe en las señales provenientes de los dispositivos. Si bien ciertas señales presentan un comportamiento similar en algunas de las descargas, en muchos casos las señales son muy diferentes entre sí, dificultando el reconocimiento de precursores disruptivos. Una serie temporal debe ser analizada ventana a ventana durante el periodo que dure la secuencia. La idea es lograr catalogar las señales como disruptivas o no disruptivas ventana a ventana con suficiente antelación para poder tomar acciones de mitigación y evitación en el caso de reconocer el aprontamiento de una disrupción. Nuevamente, al trabajar con series temporales, se construirán modelos basados en redes neuronales recurrentes LSTM y GRU.

3.2.1 Alarmas para mitigación y evitación

Los criterios que se ocuparán para validar cada resultado obtenido se harán a partir de alarmas, es decir, el tiempo donde el clasificador etiqueta un dato de la secuencia de la serie temporal como disruptiva o no disruptiva. Para activar cada tipo de alarma, es necesario un tiempo que varía según el tiempo de activación de la alarma, tendremos entonces distintos tipos de alarmas las cuales son las siguientes: Alarma Válida (VA) la cual tendrá valores sobre 30[ms] y

bajo 1[s](ecuación 3.5), Alarma Prematura (PA) la cual tendrá valores sobres 1[s](ecuación 3.6), y Alarma Tardía (TD) la cual tendrá valores mayores de 0[s] y menores a 30[ms](ecuación 3.7). La suma de VA, PA y TA nos proporcionará la Tasa de Acierto (SR) (ecuación 3.8) y junto con Alarma Perdida (MS) (ecuación 3.9) aquellas no detectadas, obtendremos el porcentaje total de elementos testeados. También a partir de los tipos de alarma en el testeo de descargas no disruptivas tendremos las alarmas Falsa Alarma (FA) cuando se detecte una señal no disruptiva como disruptiva.

$$VA = 30[ms] < WT < 1[s] \tag{3.5}$$

$$PA = WT > 1[s] \tag{3.6}$$

$$TD = 0 < WT < 30[ms] \tag{3.7}$$

$$SR = VA + PA + TD \tag{3.8}$$

$$MS = 1 - SR \Rightarrow WT \leq 0[s] \tag{3.9}$$

3.3 Entorno de trabajo

Para el diseño y construcción del modelo de red neuronal recurrente de tipo LSTM y tipo GRU se ha utilizado el lenguaje de programación Python junto con la biblioteca TensorFlow y la Interfaz de programación de aplicaciones (API) Keras [10].

3.3.1 Tensorflow

Es una biblioteca de código abierto, desarrollado y liberado por Google, que se basa en un sistema de redes neuronales. Esto significa que puede relacionar varios datos en una red simultáneamente, de la misma forma que lo hace un cerebro humano, por medio de estructuras de datos llamados tensores. Por ejemplo, puede reconocer varias palabras del alfabeto ya que relaciona letras y fonemas. Otro caso es el de imágenes y textos que se pueden

relacionar entre si rápidamente gracias a la capacidad de asociación del sistema de redes neuronales.

3.3.2 Keras

La API Keras es una librería de Aprendizaje Profundo (Deep Learning) de Python, es una interfaz de programación de aplicaciones de alto nivel, escrita en Python y capaz de ejecutarse en TensorFlow, CNTK o Theano. Nos proporciona la posibilidad de trabajar con Redes Neuronales Recurrentes gracias a sus librerías, donde el modelo LSTM y modelo GRU ya vienen implementados, permitiendo al usuario ajustar los parámetros acordes a sus necesidades. Keras fue desarrollado con un enfoque que permite la experimentación rápida y sencilla [11].

Figura 3.2: Keras, interfaz simplificada de TensorFlow.

3.4 Red neuronal para pronóstico

En esta sección se describe el proceso de implementación de la red para pronóstico. Los pasos a seguir son los siguientes:

1. Selección de datos de entrada
2. Preprocesamiento de los datos de entrada
3. División de datos en entrenamiento y testeo
4. Diseño y construcción de red neuronal
5. Implementación de la red neuronal
6. Resultados del pronóstico
7. Criterios de evaluación

Primero se debe adquirir los datos provenientes de la campaña C2830 del JET. Esta base de datos cuenta con 201 descargas disruptivas, 56 descargas disruptivas intencionales y 1036 descargas no disruptivas. Cada descarga contienen 7 señales, descritas en la tabla 2.1. Las señales que componen la base de datos del JET están muestreadas a 1[ms], las cuales fueron remuestreadas a ventanas de 10[ms]. Esto se hace tomando el promedio de cada 10 ventanas y dejar ese valor como nueva muestra.

Además, todas las señales de la base de datos fueron normalizadas antes de ingresar a la red.

Esta normalización se realizó entre 0 y 1. La base de datos es dividida en dos categorías: datos de entrenamiento con el 60 % de las señales y la base de datos de testeo con el 40 % restante.

3.4.1 Diseño de la red

El modelo de la red neuronal de tipo LSTM y GRU se muestra en el listado 3.4.1 y se concibe de la siguiente manera: 5 neuronas de entrada, 20 neuronas de estado oculto, 1 neurona de salida, 200 épocas de entrenamiento, función de activación sigmoide (por defecto), dropout recurrente de 0.1, batch size de 2, error de red RMS y optimisador RMSProp. Este diseño es experimental y está sujeto a modificaciones de acuerdo a los resultados que se obtendrán. Las 5 neuronas de entrada implica que se analizarán 5 muestras consecutivas para predecir la siguiente. Las 20 neuronas en la capa oculta se eligieron de forma tentativa, y en la capa de salida hay una sola neurona pues se pronosticará un solo dato futuro.

Listado 3.1: Implementación red LSTM/GRU Pronóstico

```
model_LSTM = Sequential()
model_LSTM.add(LSTM(20, recurrent_dropout=0.1, input_dim=look_back))
model_LSTM.add(Dense(1))
model_LSTM.compile(loss='mean_squared_error', optimizer='RMSProp')
model_LSTM.fit(trainX, trainY, nb_epoch=200, batch_size=2, verbose=2)

model_GRU = Sequential()
model_GRU.add(GRU(20, recurrent_dropout=0.1, input_dim=look_back))
model_GRU.add(Dense(1))
model_GRU.compile(loss='mean_squared_error', optimizer='RMSProp')
model_GRU.fit(trainX, trainY, nb_epoch=200, batch_size=2, verbose=2)
```

Este modelo es utilizado para los 3 experimentos de pronóstico de series temporales mencionados anteriormente. Se implementa la red descrita y se obtienen los resultados del pronóstico que son evaluados según los criterios antes establecidos.

3.5 Red neuronal para clasificación

En esta sección se describe el proceso de implementación de la red para clasificación de descargas en disruptivas o no disruptivas. Los pasos a seguir son los siguientes:

1. Selección de datos de entrada
2. Preprocesamiento de los datos de entrada
3. División de datos en entrenamiento y testeo
4. Etiquetado de señales
5. Diseño y construcción de red neuronal
6. Implementación de la red neuronal
7. Resultados de la clasificación

8. Verificación de alarmas
9. Tasas de acierto

Al igual que para el pronóstico, primero se debe adquirir los datos provenientes de la campaña C2830 del JET. En esta oportunidad tampoco se considerarán las descargas disruptivas intencionales. Todas las señales de la base de datos fueron normalizadas antes de ingresar a la red. Esta normalización se realizó entre 0 y 1. La base de datos es dividida en dos categorías: datos de entrenamiento con el 70 % de las señales y la base de datos de testeo con el 30 % restante.

Es importante aclarar que se cuentan con 201 datos de descargas disruptivas y con 1036 datos de descargas no disruptivas. Esto puede presentar problemas a la hora de considerar un porcentaje fijo de descargas para el entrenamiento, pues se tiene mucha más información no disruptiva que disruptiva, lo que puede generar un modelo sobre ajustado para las descargas seguras. Cabe destacar que los datos son normalizados entre 0 y 1 antes de entrar a la red neuronal. Además, los datos originales están muestreados a ventanas de 1 [ms], por lo que fueron remuestreados a ventanas de 10 [ms] tomando el promedio de cada 10 muestras.

En esta red se contempla el análisis simultáneo de todas las señales al momento de clasificar una descarga como disruptiva o no disruptiva. Esto quiere decir que el modelo será entrenado considerando como datos de entrada las 7 señales mencionadas anteriormente al mismo tiempo. La evaluación se llevará a cabo de la misma manera.

El modelo es modificado para que analice una muestra a la vez para cada descarga, para que de esta forma el modelo pueda ser implementado para predecir disrupciones en tiempo real. Esto complejiza la construcción del modelo, pero lo transforma en una herramienta realmente útil.

Para desarrollar un buen modelo predictivo es necesario considerar ciertas cosas. Para empezar, es necesario definir qué parte de la señal es considerada disruptiva. De la base de datos disponible se cuenta con 201 descargas disruptivas, donde los últimos instantes de cada señal tienen un comportamiento disruptivo. Esto quiere decir que el comportamiento previo a dichos instantes debe ser catalogado como seguro (etiqueta = 0). Entonces una señal será considerada segura desde su inicio hasta que se reconozca un comportamiento disruptivo (etiqueta = 1). Por esta razón es importante determinar dónde comienza el comportamiento disruptivo para cada descarga. Se proponen 3 tiempos disruptivos para abordar este problema. Estos tiempos se reflejan en las últimas 100, 50 y 25 muestras de cada descarga de entrenamiento, generando de esta forma 3 tipos de modelos. La figura 3.3 representa lo expuesto.

Luego, es importante considerar el desbalance que existe entre las muestras disruptivas y no disruptivas, siendo las descargas seguras mayores en una proporción de 5 a 1. El modelo toma aleatoriamente muestras disruptivas y seguras para el entrenamiento de acuerdo a una variable que representa la cantidad de muestras disruptivas entrenadas versus la cantidad de muestras seguras, procurando un mayor entrenamiento para las descargas disruptivas. De esta

forma se intenta compensar el desbalance existente en la información de entrenamiento. Esta variable considera los valores 2, 4, 8, 16 y 32. Cada uno de los 3 tipos de modelos anteriores contempla esta variable, llegando a un total de 15 experimentos para cada tipo de red (LSTM y GRU).

Finalmente es necesario aclarar que las descargas deben tener una cantidad mínima de muestras para que puedan entrar a la red. Esta cantidad depende de las muestras que se consideren disruptivas y de la cantidad mínima de muestras catalogadas como seguras al comienzo de cada descarga. Este largo mínimo fue fijado en 500 muestras, lo que representan 5[s](recordar que las descargas fueron remuestreadas a ventadas de 10 [ms]). Por esta razón, las descargas que no contaban con esta cantidad mínima de muestras (13 descargas seguras y 8 descargas disruptivas) fueron descartadas del análisis.

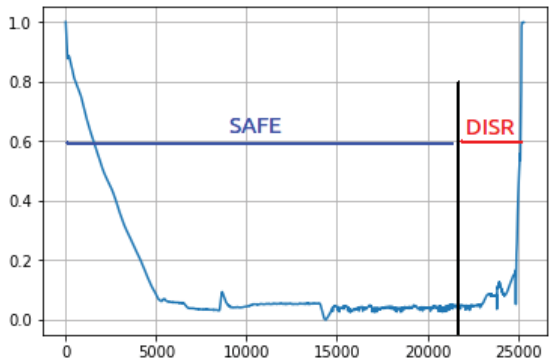


Figura 3.3: Etiquetado para cada señal.

3.5.1 Diseño de la red

El diseño de la red neuronal tipo LSTM y tipo GRU se concibió de la siguiente manera: los datos de entrenamiento con retardo ingresan a la red LSTM/GRU, con las siguientes características: 7 neuronas de entrada (una por cada señal), 128 neuronas ocultas, 2 neuronas de salida (función de activación softmax), función de error Mean squared error, Optimizador Adam y 100 épocas. En el listado 3.5.1 se muestra la implementación para la red LSTM.

Listado 3.2: Implementación red LSTM para clasificación

```
inputs = Input([None, len(signal)])
input_ = []
for i in range(len(signal)):
input_.append(BatchNormalization()(Lambda(lambda x: x[..., i:i+1])(inputs
  )))
x = Concatenate()(input_)
x = LSTM(128)(x)
out = Dense(2, activation='softmax')(x)
model_LSTM = Model(inputs, out)
op = Adam(1e-4, amsgrad=True)
model_LSTM.compile(op, 'categorical_crossentropy', metrics=['acc'])
```

De igual manera, en el listado 3.5.1 se muestra la red tipo GRU

Listado 3.3: Implementación red GRU para clasificación

```
inputs = Input([None, len(signal)])
input_ = []
for i in range(len(signal)):
input_.append(BatchNormalization()(Lambda(lambda x: x[..., i:i+1])(inputs
  )))
x = Concatenate()(input_)
x = LSTM(128)(x)
out = Dense(2, activation='softmax')(x)
model_GRU = Model(inputs, out)
op = Adam(1e-4, amsgrad=True)
model_GRU.compile(op, 'categorical_crossentropy', metrics=['acc'])
```

Esta configuración de red contempla, como se mencionó con anterioridad, 128 neuronas en la capa oculta. Estas neuronas representan bloques LSTM y GRU. Como se puede apreciar, sólo se tiene una capa oculta. Esta configuración fue probada en diferentes oportunidades con mayor cantidad de capas ocultas, donde se fue variando la cantidad de neuronas en cada caso. Las configuraciones implementadas fueron las siguientes:

1- 3 capas ocultas, con 128 neuronas cada una.

2- 5 capas ocultas, con 64 neuronas en la primera y última capa oculta, y 128 en el resto.

3- 7 capas ocultas, con 32 neuronas en la primera y última capa oculta, 64 neuronas en las contiguas y 128 en las capas del medio.

4- 10 capas ocultas, agregando capas de 128 neuronas al caso anterior, junto a las otras capas de 128 neuronas.

Estas configuraciones, al igual que para el diseño de la red para pronóstico, fueron tentativas y puestas a prueba. Las que obtuvieron mejor desempeño, tanto para las redes LSTM y GRU, fueron las que se mencionan en los listados 3.2 y 3.3 respectivamente.

4 Resultados

En esta sección se mostrarán y comentarán los resultados obtenidos de la implementación de los modelos basados en redes neuronales recurrentes LSTM y GRU propuestos tanto para el pronóstico de series temporales y para la clasificación de descargas disruptivas y no disruptivas. Se presentan tablas y gráficas de resultados, todo comentado.

En primera instancia, se mostrarán los resultados obtenidos del pronóstico de series temporales provenientes de la campaña C2830 del JET. En esta oportunidad se evaluarán los 3 experimentos propuestos. Para todos los experimentos se utiliza una red LSTM y una GRU.

En segunda instancia se mostrarán los resultados obtenidos para la clasificación de descargas disruptivas y no disruptivas. Para esta parte del proyecto también se cuenta con la base de datos del JET, así como también es evaluado el desempeño de la red LSTM y GRU. Es necesario recordar que para la clasificación se varió la cantidad de datos etiquetados como disruptivos en 3 experimentos. Primero se evaluó la clasificación considerando las últimas 100 ventanas (1[s]) como disruptivas. Luego se realizó el mismo procedimiento considerando las últimas 50 ventanas como disruptivas. Finalmente se repitió el proceso considerando las últimas 25 ventanas como disruptivas. Para cada experimento se varió la razón de descargas disruptivas versus descargas no disruptivas en 5 factores (2,4,8,16 y 32). Estos 15 experimentos fueron evaluados tanto para la red LSTM como la GRU.

4.1 Pronóstico de series temporales

A continuación se muestran los resultados para el pronóstico en cada uno de los 3 experimentos realizados.

4.1.1 Análisis uno a uno

Como ya se mencionó, este análisis fue realizado descarga por descarga, en cada una de las 7 señales, tanto para descargas disruptivas como no disruptivas. Los resultados son los siguientes:

En la tabla 4.1 se aprecia que el mejor pronóstico para descargas no disruptivas, analizando cada señal de forma independiente, fue para la densidad electrónica del plasma con la red

Tabla 4.1: Exactitud método *uno a uno* no disruptivo

Señal	LSTM %	GRU %
1 Corriente de plasma	99.15	98.69
2 Amplitud de "Mode Lock"	92.94	93.63
3 Inductancia interna del plasma	97.58	97.01
4 Densidad electrónica del plasma	98.96	99.20
5 Derivada de la energía diamagnética	42.34	32.65
6 Potencia radiada	93.58	91.54
7 Potencia total de entrada	93.13	93.37

GRU con un 99,2 % de exactitud. El mejor pronóstico para la red LSTM fue para la señal corriente de plasma con un 99,1 % de exactitud. Por otra parte, los peores pronósticos fueron para la señal derivada de la energía diamagnética, con un 42,34 % para la red LSTM y un 32,65 % para la red GRU.

Tabla 4.2: Exactitud método *uno a uno* disruptivo

Señal	LSTM %	GRU %
1 Corriente de plasma	95.65	96.3
2 Amplitud de "Mode Lock"	90.28	89.66
3 Inductancia interna del plasma	95.58	96.14
4 Densidad electrónica del plasma	95.58	96.22
5 Derivada de la energía diamagnética	32.76	37.23
6 Potencia radiada	84.89	83.47
7 Potencia total de entrada	92.34	93.45

Para el análisis de las descargas disruptivas, por su parte, el mejor pronóstico lo obtuvo la red GRU con un 96,3 % de exactitud y un 95,65 % para la red LSTM, ambos para la señal corriente de plasma. Nuevamente el peor pronóstico fue para la señal derivada de la energía diamagnética, con un 32,76 % y un 37,23 % para las redes LSTM y GRU respectivamente.

4.1.2 Análisis de los 2 segundos previos a la disrupción

Como se mencionó con anterioridad, este análisis es similar al anterior, con la diferencia de que sólo entrará a la red los últimos dos segundos antes de la disrupción. Sólo las descargas disruptivas son analizadas. Las señales son analizadas una por una. Los resultados obtenidos son los siguientes:

Tabla 4.3: Exactitud método de los *últimos dos segundos*

Señal	LSTM %	GRU %
1 Corriente de plasma	99.81	99.77
2 Amplitud de "Mode Lock"	93.11	93.31
3 Inductancia interna del plasma	99.46	99.18
4 Densidad electrónica del plasma	98.99	99.37
5 Derivada de la energía diamagnética	89.30	89.78
6 Potencia radiada	94.99	96.29
7 Potencia total de entrada	97.14	96.92

Este análisis obtuvo su mejor pronóstico para la red LSTM con un 99,81 % de exactitud en la señal corriente de plasma, siendo hasta el momento el mejor resultado obtenido entre todos los experimentos. Para la red GRU, el mejor resultado también fue para la corriente de plasma, con un 99,77 % de exactitud. Los peores pronósticos fueron nuevamente para la señal derivada de la energía diamagnética, con un 89,3 % y un 89,78 % para las redes LSTM y GRU respectivamente. Cabe destacar que estos han sido los mejores pronósticos para esta señal.

4.1.3 Análisis señales completas

En esta oportunidad se estudia el entrenamiento a base de señales completas, para luego testear con otras señales, también completas. En el caso de las descargas disruptivas, se considera, al igual que en el apartado anterior, los últimos 2 segundos antes de la disrupción. Los resultados son los siguientes:

Tabla 4.4: Exactitud método de *señales completas* no disruptivas.

Señal	LSTM %	GRU %
1 Corriente de plasma	96.70	96.40
2 Amplitud de "Mode Lock"	94.30	92.76
3 Inductancia interna del plasma	94.71	94.54
4 Densidad electrónica del plasma	96.22	96.70
5 Derivada de la energía diamagnética	43.45	43.92
6 Potencia radiada	93.98	91.73
7 Potencia total de entrada	57.15	58.38

Como se puede apreciar, el mejor pronóstico lo obtuvo la red LSTM con un 96,7 % en la señal corriente de plasma. El mismo resultado lo obtuvo la red GRU para la señal densidad electrónica del plasma. Una vez más, los peores pronósticos fueron para la señal derivada de la energía diamagnética, con un 43,45 % y un 43,92 % para las redes LSTM y GRU respectivamente.

Tabla 4.5: Exactitud método de *señales completas* disruptivas.

Señal	LSTM %	GRU %
1 Corriente de plasma	99.63	99.74
2 Amplitud de "Mode Lock"	99.24	99.34
3 Inductancia interna del plasma	99.59	99.56
4 Densidad electrónica del plasma	99.38	99.28
5 Derivada de la energía diamagnética	29.26	56.27
6 Potencia radiada	97.03	96.25
7 Potencia total de entrada	98.85	99.26

Para el análisis de las descargas disruptivas, por su parte, el mejor pronóstico lo obtuvo la red GRU con un 99,74 % de exactitud y un 99,63 % para la red LSTM, ambos para la señal corriente de plasma. Nuevamente el peor pronóstico fue para la señal derivada de la energía diamagnética, con un 29,26 % y un 56,27 % para las redes LSTM y GRU respectivamente.

4.2 Clasificación de descargas disruptivas y no disruptivas

A continuación se muestran los resultados para la clasificación de descargas en disruptivas y no disruptivas. Como se mencionó en el capítulo anterior, se propone un sistema de alarmas. Las redes implementadas están programadas para que analicen cada señal ventana a ventana, etiquetando cada una de ellas como disruptiva o no disruptiva. Sucede en muchos casos que algunas descargas tienen un breve comportamiento disruptivo y luego se comportan de forma segura. Para hacer un sistema más eficiente y evitar una alta tasa de falsas alarmas, se determinó que una descarga completa será catalogada como disruptiva cuando presente 10 ventanas consecutivas etiquetadas como disruptivas. Cuando esto suceda, se lanzará una alarma, la cual llamaremos Warning Time (WT). en los resultados siguientes se muestran los porcentajes relacionados a los criterios de evaluación mencionados en el capítulo anterior, además del valor promedio de los tiempos de advertencia y su respectiva desviación estándar. La idea principal es obtener el mayor porcentaje de alarmas válidas posible.

4.2.1 Clasificación con red LSTM

Las tablas presentadas en este capítulo presentan la tasa de acierto (SR), las alarmas válidas (VA), las alarmas prematuras (PA), alarmas tardías (TD), las alarmas perdidas (MS), las falsas alarmas (FA), el promedio de los tiempos de las alarmas detectadas antes de la disrupción en [ms] (Mean) y la desviación estándar de los tiempos (STD).

Los resultados de clasificar con la red neuronal recurrente tipo LSTM se muestran a continuación.

Tabla 4.6: Resultados con red LSTM considerando 100 muestras disruptivas

Razón	LSTM							
	SR%	VA%	PA%	TD%	MS%	FA%	Mean [ms]	STD[ms]
2	16	0	16	0	84	34	4045	1453
4	100	0	100	0	0	26	10370	3970
8	52	0	52	0	48	28	4845	3879
16	20	0	20	0	80	20	5349	2087
32	100	0	100	0	0	22	12164	4785

En la tabla 4.6 se muestran los resultados del primer experimento de esta sección. Si bien en algunos casos se obtubo un 100 % de tasa de acierto, ninguna de estas alarmas fue válida, pues los tiempos de advertencia fueron considerados prematuros. Los promedios de los tiempos de advertencia son demasiado elevados, evidenciando que esta clasificación no es la óptima.

Tabla 4.7: Resultados con red LSTM considerando 50 muestras disruptivas

Razón	LSTM							
	SR%	VA%	PA%	TD%	MS%	FA%	Mean [ms]	STD[ms]
2	0	0	0	0	100	30	0	0
4	100	0	100	0	0	24	9370	4528
8	94	10	84	0	6	26	4293	1546
16	100	8	92	0	0	18	6660	1922
32	0	0	0	0	100	16	0	0

En la tabla 4.7 se puede observar un porcentaje de alarmas válidas en 2 experimentos. Si bien sigue siendo un porcentaje demasiado pequeño, presenta una mejora con el experimento anterior. Las alarmas prematuras siguen siendo demasiadas, dejando en evidencia que el sistema aún no funciona correctamente.

Tabla 4.8: Resultados con red LSTM considerando 25 muestras disruptivas

Razón	LSTM							
	SR%	VA%	PA%	TD%	MS%	FA%	Mean [ms]	STD[ms]
2	100	8	92	0	0	24	6601	2021
4	0	0	0	0	100	22	0	0
8	100	8	92	0	0	26	2856	849
16	98	10	88	0	2	18	1253	687
32	100	44	56	0	0	12	967	285

En la tabla 4.8 se muestran los resultados para el último experimento de esta sección. Se obtuvo un porcentaje de alarmas válidas notoriamente más elevado (44%). La tasa de acierto fue de 100% con un promedio de 967[ms] en los tiempos de advertencia.

En este experimento se busca encontrar una relación entre la cantidad de datos disruptivos y la tasa de acierto para las alarmas válidas. Como se puede apreciar en los resultados obtenidos para red LSTM, la tasa de acierto para las alarmas válidas va aumentando a medida que se catalogan menos datos disruptivos. Esto tiene sentido si se toma en cuenta que el experimento se detiene en el momento de la disrupción, considerando que dentro de un margen razonable (el cual se está testeando) cada descarga se comporta de manera no disruptiva hasta el momento de la misma disrupción. Al considerar menos datos disruptivos en el entrenamiento nos aseguramos que dicha información, en su totalidad, contiene mayor porcentaje de datos disruptivos, generando así una mejor clasificación. Este supuesto se evidencia con los resultados crecientes de la tasa de acierto para las alarmas válidas.

De igual manera, se aprecia un aumento de alarmas válidas al considerar mayor número de descargas disruptivas en comparación con las no disruptivas al momento de entrenar la red.

4.2.2 Clasificación con red GRU

Los resultados de clasificar con la red neuronal recurrente tipo GRU son los siguientes.

Tabla 4.9: Resultados con red GRU considerando 100 muestras disruptivas

Razón	GRU							
	SR%	VA%	PA%	TD%	MS%	FA%	Mean [ms]	STD[ms]
2	18	0	18	0	82	20	5035	2453
4	70	0	70	0	30	18	8470	1970
8	50	0	50	0	50	20	3245	1859
16	60	0	60	0	40	26	2349	1067
32	100	0	100	0	0	28	4154	1785

En la tabla 4.9 se muestran los resultados del primer experimento de esta sección. Si bien en un caso se obtubo un 100% de tasa de acierto, ninguna de estas alarmas fue válida, pues los tiempos de advertencia fueron considerados prematuros. Los promedios de los tiempos de advertencia son demasiado elevados, evidenciando que esta clasificación no es la óptima.

Tabla 4.10: Resultados con red GRU considerando 50 muestras disruptivas

Razón	GRU							
	SR%	VA%	PA%	TD%	MS%	FA%	Mean [ms]	STD[ms]
2	0	0	0	0	100	30	0	0
4	100	0	100	0	0	18	5360	2528
8	34	0	34	0	66	22	4253	1266
16	100	8	92	0	0	16	3360	1520
32	90	0	90	0	10	10	3432	1783

En la tabla 4.10 se puede observar un porcentaje de alarmas válidas en 1 experimento. Si bien sigue siendo un porcentaje demasiado pequeño, presenta una mejora con el experimento anterior. Las alarmas prematuras siguen siendo demasiadas, dejando en evidencia que el sistema aún no funciona correctamente.

Tabla 4.11: Resultados con red GRU considerando 25 muestras disruptivas

Razón	GRU							
	SR%	VA%	PA%	TD%	MS%	FA%	Mean [ms]	STD[ms]
2	92	0	92	0	8	18	4231	1123
4	0	0	0	0	100	12	0	0
8	100	4	96	0	0	20	1856	872
16	98	18	80	0	2	12	1341	567
32	100	40	60	0	0	10	977	381

En la tabla 4.11 se muestran los resultados para el último experimento de esta sección. Nuevamente se obtuvo un porcentaje de alarmas válidas notoriamente más elevado (40 %). La tasa de acierto fue de 100 % con un promedio de 977[ms] en los tiempos de advertencia.

Los resultados obtenidos para la red GRU son similares a los obtenidos con la red LSTM. Como se puede apreciar en esta oportunidad, la tasa de acierto para las alarmas válidas va aumentando a medida que se catalogan menos datos disruptivos. Nuevamente el supuesto de que considerar menos datos disruptivos en el entrenamiento nos asegura un mejor desempeño se evidencia con los resultados crecientes de la tasa de acierto para las alarmas válidas.

Igual que en análisis anterior, se aprecia un aumento de alarmas válidas al considerar mayor número de descargas disruptivas en comparación con las no disruptivas al momento de entrenar la red.

Discusión y conclusiones

La fusión nuclear promete ser una fuente de generación de energía comprometida con el medio ambiente y abundante en producción. Su principal desventaja es la complejidad de replicar las condiciones óptimas para su generación, siendo la disrupción del plasma la gran limitante para pasar de un dispositivo a un reactor. El principal objetivo de este proyecto fue implementar una red neuronal capaz de analizar la gigantesca cantidad de datos generados por un dispositivo de fusión nuclear, análisis que es prácticamente imposible de realizar por un operario.

Estos datos a analizar están constituidos por series temporales asociadas a las distintas variables monitoreadas en el dispositivo de fusión nuclear Tokamak JET. Este trabajo conllevó el estudio de redes neuronales recurrentes, ya que tienen la capacidad de analizar secuencias de datos. Se plantean entonces dos tareas fundamentales: el pronóstico de series temporales asociadas al dispositivo y la clasificación de estas señales entre disruptivas o no disruptivas.

Las redes neuronales recurrentes estándar presentan problemas con las dependencias a largo plazo, por lo que fue necesario estudiar arquitecturas más complejas. Los modelos de estudio elegidos fueron las redes neuronales recurrentes LSTM y GRU, ya que ambas redes puede lidiar con el problema de la dependencia a largo plazo.

La característica principal que tienen en común ambas redes es la celda de estado. Dicha celda no está presente en la red neuronal recurrente tradicional, la cual cuenta con una sola capa oculta con función de activación tangente hiperbólica. Tanto la red LSTM como la GRU son capaces de mantener, agregar o quitar información relevante de esta celda de estado, resolviendo de esta forma el problema de las dependencias de datos muy pasados.

La diferencia entre estas redes está en que en la unidad LSTM la cantidad de contenido de la memoria que es visto, o usado por las otras unidades en la red, es controlada por la compuerta de salida. La unidad GRU expone todo el contenido sin ser controlado. Otra diferencia importante es la localización de la compuerta de entrada, o la correspondiente compuerta de reseteo. La unidad LSTM calcula el nuevo contenido de la memoria sin ningún control por separado de la cantidad de información que fluye desde los pasos de tiempos previos. Por otra parte, la unidad GRU controla la información que fluye desde pasos previos de activación mediante el cálculo de nuevos candidatos de activación, pero no controla de forma independiente la cantidad de activación candidata que se agrega.

Debido a sus similitudes, ambas redes fueron utilizadas en todos los experimentos implementados en este proyecto. En la gran mayoría de los casos se observan resultados muy similares entre la red LSTM y la GRU, por lo que es difícil decidir cuál de las dos redes funciona mejor.

Para el pronóstico de series temporales te realizaron 3 experimentos. Se utilizó la base de datos proveniente de la campaña C2830 del Tokamak JET y se pronosticó para una ventana de tiempo. En general las tasas de exactitud fueron bastante elevadas. En el primer experimento, que consistió en analizar cada señal de forma independiente, el mejor pronóstico para descargas no disruptivas fue para la señal densidad electrónica del plasma con la red GRU con un 99,2 % de exactitud. El mejor pronóstico para la red LSTM fue para la señal corriente de plasma con un 99,1 % de exactitud. Para el análisis de las descargas disruptivas, por su parte, el mejor pronóstico lo obtuvo la red GRU con un 96,3 % de exactitud y un 95,65 % para la red LSTM, ambos para la señal corriente de plasma.

Para el análisis de los dos segundos previos a la disrupción se obtuvieron los mejores resultados del proyecto (en cuanto a pronóstico) con un 99,81 % de exactitud en la señal corriente de plasma para la red LSTM. Para la red GRU, el mejor resultado también fue para la corriente de plasma, con un 99,77 % de exactitud.

En el análisis de las señales completas, para las descargas no disruptivas, el mejor pronóstico lo obtuvo la red LSTM con un 96,7 % en la señal corriente de plasma. El mismo resultado lo obtuvo la red GRU para la señal densidad electrónica del plasma. Para el análisis de las descargas disruptivas, por su parte, el mejor pronóstico lo obtuvo la red GRU con un 99,74 % de exactitud y un 99,63 % para la red LSTM, ambos para la señal corriente de plasma.

En todos los experimentos el peor pronóstico fue para la señal derivada de la energía diamagnética, alcanzando tasas de exactitud bastante bajas en comparación a los mejores resultados.

En cuanto a la clasificación, los resultados obtenidos no fueron muy alentadores. Si bien en varias oportunidades se alcanzó una tasa de acierto de 100 %, el mayor porcentaje fue de alarmas prematuras. Esto no es bueno porque la disrupción ocurre al final de cada descarga disruptiva, lo que significa que todas las ventanas previas a una disrupción debe ser etiquetadas como no disruptivas. Una alarma demasiado prematura significa que la red está catalogando una zona segura como disruptiva. Lo ideal sería tener un 100 % de alarmas válidas (tiempo de advertencia entre 30[ms] y 1[s]). Los mejores resultados se dieron con la red LSTM considerando los 25 últimos datos como disruptivos con una tasa de acierto de 44 % de alarmas válidas. La red GRU alcanzó un 40 % de alarmas válidas bajo las mismas condiciones, siendo este su mejor resultado. Esto deja en evidencia que queda mucho por mejorar para alcanzar los estándares permitidos por ITER(TPR >95 % y FPR <5 %).

Bibliografía

[1] G. E. S. Yearbook, *Global Energy Statistical Yearbook*, 2018. [En línea]. Disponible en: https://yearbook.enerdata.net/

[2] C. S. Nuclear, *Consejo de Seguridad Nuclear*, 2018. [En línea]. Disponible en: https://www.csn.es/

[3] A. T. Universo, *A través del Universo*, 2008. [En línea]. Disponible en: https://inigo.sendino.org

[4] EUROfusion, "Eurofusion," 2018. [En línea]. Disponible en: http://www.euro-fusion.org.

[5] ITER, "Iter," 2018. [En línea]. Disponible en: http://www.iter.org.

[6] S. Miller, *How to Build a Neural Network*, 2015. [En línea]. Disponible en: https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/

[7] Y. LeCun, K. Müller, L. Bottou, y G. Orr, *Efficient BackProp*. Springer, 1998.

[8] T. D. Science, "Why deep learning over traditional machine learning?" 2018. [En línea]. Disponible en: https://towardsdatascience.com

[9] C. Olah, "Understanding lstm networks," 2015. [En línea]. Disponible en: http://colah.github.io

[10] Tensorflow, "Tensorflow," 2018. [En línea]. Disponible en: https://www.tensorflow.org/guide/keras

[11] J. Brownlee, *Deep Learning with Python*. Machine Learning Mastery, 2016.