UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA DE POSTGRADO Y EDUCACIÓN CONTINUA

FILTRO PREDICTIVO PARA STREAMS DE DATOS EN TIEMPO REAL EN RED DE GEODESIA GNSS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIA DE DATOS

DAVID ALEJANDRO PINEDA OSORIO

PROFESORA GUÍA:
NANCY HITSCHFELD KAHLER

PROFESOR CO-GUÍA:
JUAN BÁEZ SOTO

MIEMBROS DE LA COMISIÓN:
SERGIO BARRIENTOS PARRA
ENRIQUE CORDARO CÁRDENAS

SANTIAGO DE CHILE
2026

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIA DE DATOS
POR: DAVID ALEJANDRO PINEDA OSORIO
FECHA: 2026
PROFESORA GUÍA: NANCY HITSCHFELD KAHLER
PROFESOR CO-GUÍA: JUAN BÁEZ SOTO

FILTRO PREDICTIVO PARA STREAMS DE DATOS EN TIEMPO REAL EN RED DE GEODESIA GNSS

El monitoreo de deformaciones terrestres mediante sensores de posicionamiento geodésico (GNSS) es fundamental para la sismología, pero presenta desafíos debido al ruido en las señales y la complejidad de los datos. En la red del Centro Sismológico Nacional (CSN), se observa que los sensores cercanos comparten patrones de similitud, lo que ofrece una oportunidad para mejorar la predicción de desplazamientos geodésicos utilizando técnicas avanzadas de aprendizaje profundo.

El objetivo de esta tesis fue desarrollar un filtro predictivo de ruido y modelos de predicción basados en datos GNSS, evaluando su desempeño mediante arquitecturas de aprendizaje profundo. Para ello, se realizaron las siguientes tareas: análisis exhaustivo de las secuencias GNSS para identificar características subyacentes, selección y aplicación de modelos predictivos (CNN/MLP, RNN/LSTM y Transformers), definición de métricas como RMSE, AIC y BIC para evaluar su rendimiento, e implementación de estrategias de paralelización en entornos de alto rendimiento mediante GPU. Dynamic Time Warping (DTW) fue empleado para la alineación de las series temporales, mejorando el ajuste durante el entrenamiento.

Los experimentos confirmaron que los modelos RNN/LSTM sobresalen en la gestión de secuencias temporales, logrando una mayor reducción de ruido y precisión en predicciones. Los modelos CNN/MLP también demostraron ser efectivos, aunque con menor desempeño en comparación con RNN/LSTM. Por su parte, los modelos Transformers no alcanzaron los resultados esperados, resaltando la necesidad de ajustar su arquitectura para capturar patrones más complejos. Se comprobó que los modelos desarrollados son capaces de adaptarse a la geografía local, siendo útiles para aplicaciones prácticas como la detección temprana de sismos o el análisis de sismicidad de baja magnitud. Sin embargo, se identificaron limitaciones, como la sincronización de datos y restricciones de hardware, que abren oportunidades para trabajos futuros.

Como principales aportes, este trabajo destaca la capacidad de los modelos para reducir el ruido y generar predicciones en tiempo real, el desarrollo de métodos de ordenamiento para incorporar nuevas variables como velocidad y aceleración, y su potencial para aplicaciones más allá de la sismología, como el análisis de señales en sistemas industriales y biomédicos. En el futuro, se plantea integrar vectores adicionales, optimizar arquitecturas más complejas y explorar aplicaciones del sistema en monitoreo en tiempo real.

A todas esas mentes curiosas,
que ven las variables ocultas.

Agradecimientos

Agradezco el apoyo que he recibido para realizar este trabajo a mis padres, Elda Osorio y Pedro Pineda, habría sido mucho más difícil sin tenerlos a mi lado. Por el tiempo dedicado y la sabia guía en la construcción y desarrollo de esta tesis a Nancy Hitschfeld, Juan C. Baez Soto, Sergio Barrientos y Enrique Cordaro. Así como también a mis amistades y a cada persona con quien he podido encontrarme en este camino recorrido. ¡Muchas gracias!

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

Este trabajo consiste en implementar una solución mediante la aplicación de diferentes técnicas y algoritmos relacionados con aprendizaje de máquinas, es específico con modelos de aprendizaje profundo.

El sistema sobre el que se aplica este estudio es un conjunto de sensores operando en red, ubicados geográficamente, que producen una señal serie-tiempo de posicionamiento geográfico (GNSS). Esta red se ubica en la extensión norte a sur de Chile y es administrada por el Centro Sismológico Nacional (CSN) de la Universidad de Chile.

Si llamamos por *valor natural* a las serie-tiempo que se producen en cada una de las estaciones de esta red, se puede observar que presentan características y patrones que podrían ser abordables o procesados por una inteligencia artificial (IA). Principalmente se puede observar ruido y patrones en vecindad con las estaciones cercanas cuando el tramo observado está se comporta de manera normal (no hay presencia sismos).

El *enfoque* de este trabajo consistirá en analizar *streams* de datos de tipo GNSS, proponer soluciones que permitan agrupar por características similares y estudiar estrategias para la reducción del ruido, aplicando estas soluciones en la red de geodesia de CSN en tiempo real.

Se debe considerar que en conjunto la aceleración, velocidad y desplazamiento son los tres tipos de observaciones utilizados en sismología en la actualidad. Es un esquema de tres niveles que opera y mantiene la Red Sismológica Nacional (RSN) la que observa estos parámetros en **tiempo real**, que son usados en forma independiente y conjunta, mediante diferentes modelos, para estimar el hipocentro y magnitud de eventos sísmicos.

El objetivo principal de estas mediciones es localizar y estimar la magnitud de cada sismo observado. La localización consiste en encontrar el hipocentro (ubicación geográfica y profundidad). Siendo la intensidad una medida de energía con que ocurre el sismo.

Existen diferentes sensores que sufren igualmente de saturación de las señales, en el caso de eventos de magnitud superior a Mw7.0, para aquellos sensores de velocidad y aceleración que están cercanos al epicentro, el *desplazamiento* se torna una medida de importancia, ya que estos sensores no tienen problemas de saturación en la amplitud de su señal (Leyton y cols., 2018).

El desplazamiento es derivado de la posición estimada con receptores GNSS, con observaciones a 1 Hz en tiempo real (una muestra por segundo). Esta medición de desplazamiento presenta la mejor solución que puede ser estimada en la actualidad en tiempo real, con un error que puede variar entre 3 a 7 centímetros, con un valor medio de 4cm, medidos sobre

el ruido de la propia observación. Considera ruidos por ionósfera, troposfera y la fase de la onda emitida por el satélite al punto de recepción.

El desafío es poder caracterizar este *ruido*, dado que es generado por *efectos no corregidos* en la observación, los cuales se presentan por igual en las estaciones. No obstante esto, el valor estimado versus la desviación se hace significativo para desplazamientos de 5 centímetros, lo que representa un sismo de magnitud 6,9Mw (Ruiz y cols., 2017).

El ruido que se presenta en las estaciones puede ser caracterizado, lo cual nos plantea la pregunta **¿será posible diseñar un filtro predictivo de ruido, que permita otorgar una mayor sensibilidad a la detección?** Y, con este, habilitar un set de herramientas modulares para cálculos de aplicación sísmica.

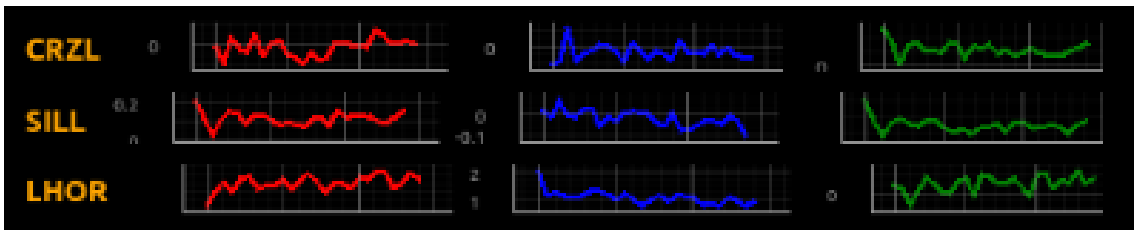

Figura 1.1: Serie tiempo GNSS para tres estaciones vecinas (E-N-U)

En la figura 1.1 se muestra la señal de desplazamiento descompuesta en componentes este-oeste, norte-sur y arriba-abajo. Es un grupo de estaciones de la red cercanas entre sí que, pese a sus diferentes curvas, presentan ciertas tendencias similares que podríamos asociar.

Las variaciones de desplazamiento son presentadas en función de serie-tiempo en que su curva presenta un ruido que podría caracterizar y filtrar. También, bajo una mirada general es posible observar patrones asociativos en estaciones cercanas.

Por último, dado que el *enfoque* de este trabajo es aportar en la creación de una solución que permita un monitoreo de sismicidad en tiempo real, será necesario crear las estrategias para generar modelos que puedan utilizar los recursos disponibles en tarjetas gráficas (GPU) (Navarro, 2014), la definición correcta de una arquitectura CPU/GPU es clave para generar una mejor solución dentro de lo posible.

1.1. Hipótesis

Es posible diseñar e implementar un filtro predictivo basado en modelos de aprendizaje profundo que permita reducir el ruido en señales GNSS en tiempo real, mejorando la sensibilidad para la detección de eventos sísmicos y habilitando herramientas modulares para aplicaciones en sismología.

1.2. Objetivos

En esta sección se definen los objetivos que este trabajo plantea a realizar.

1.2.1. Objetivo General

Desarrollar e implementar un filtro de ruido eficiente para su aplicación en los flujos de datos GNSS en tiempo real.

1.2.2. Objetivos Específicos

Los siguientes objetivos consisten en las tareas principales a realizar para este trabajo.

1. Realizar un análisis exhaustivo de las secuencias de datos provenientes de la red de estaciones GNSS para comprender sus características y patrones subyacentes.
2. Seleccionar y aplicar modelos predictivos adecuados que permitan el procesamiento y la interpretación efectiva de las secuencias de datos.
3. Definir métricas precisas y pertinentes para evaluar el rendimiento de los modelos en relación con los diferentes conjuntos de hiperparámetros considerados.
4. Diseñar estrategias de paralelización eficientes que permitan la implementación del filtro en entornos de procesamiento de alto rendimiento, como las tarjetas gráficas (GPU).

1.3. Metodología

Este trabajo consta de dos fases principales:

1. Análisis estático, tramos de datos fijos.
2. Propuesta de Modelos de aprendizaje profundo para filtrado de ruido y predicción.

Para abordar la *fase 1* se hace un análisis de datos en su estructura micro y macro. Este primera etapa consiste en un trabajo sobre los datos en estático (tramos de serie tiempo) para caracterizar y establecer una línea base.

En el trabajo para la *fase 2* Se diseñan e implementan diferentes modelos para agrupar y clasificar en base a patrones que se manifiestan en los datos de manera individual y grupal.

Se evalúan los rendimientos de los modelos en base a diferentes estrategias de paralelización a GPU. También se establece una tabla de recomendaciones y métricas definidas en el trabajo.

1.4. Estructura de la tesis

Los siguientes capítulos de esta tesis se estructuran de la siguiente manera.

2. Marco Teórico
   Contiene información teórica sobre los modelos, recursos ocupados, referencias académicas y estado del arte relacionado.

3. Análisis estático y diseño de modelos
   Contiene un estudio detallado del conjunto de datos con que se trabaja en esta tesis y se diseñan los modelos a entrenar.

4. Implementación de modelos de aprendizaje profundo

   Consiste en implementar modelos con diferentes arquitecturas internas, comparar sus resultados como filtro predictivo y escoger aquel que presente un mejor rendimiento según las métricas estudiadas.

5. Conclusiones.

Capítulo 2

Marco Teórico

2.1. Medir fenómenos sísmicos

El planeta Tierra se puede comprender como una masa heterogénea y dinámica, compuesta por una serie de capas de diferente composición bajo diferentes condiciones físicas y químicas, sean presión, temperatura, etc.

Esta amalgama de materiales y diversidad de condiciones generan una forma volumétrica del geoide, aglomeradas gracias a la gravedad.

Dentro de los fenómenos que ocurren en este planeta se incluye la tectónica de placas la *tectónica de placas* o desplazamiento de placas, que rozan y solapan unas con otras y, al ocurrir un deslizamiento entre ellas, produce una liberación de energía que genera ondas que se propagan a través del medio afectando todo lo que esté en contacto con este, deformando finalmente la figura de la Tierra.

La suma a lo largo del tiempo en escala geológica resulta en la deriva continental, teoría desarrollada por Wegener (1912), que explica la existencia de un principio de movilidad horizontal de los bloques continentales y oceánicos. En el *Cinturón del Pacífico* es posible reconocer este fenómeno como algo frecuente a través del tiempo y se refleja en una sismicidad que varía desde algo imperceptible, hasta grandes terremotos que cambian drásticamente el paisaje y las instalaciones humanas en cuestión de minutos, como el terremoto de Valdivia 1960. El tamaño de un terremoto se representa a través del Momento Sísmico, que es proporcional al área y al deslizamiento medio en esta área. La magnitud de un sismo es proporcional al logaritmo del Momento Sísmico.

Desde la fuente sísmica asociada a un sismo se generan principalmente dos tipos de ondas: ondas P, cuyo movimiento de partículas en el medio coincide con la dirección de propagación de la onda, y ondas S en que el movimiento de las partículas es perpendicular a la dirección de la onda. En un semi-espacio, bajo ciertas condiciones, la interacción entre estas ondas da origen a ondas superficiales. Las ondas P se propagan con mayor velocidad que las ondas S, pero en general poseen menor amplitud. La deformación estática asociada al paso de las ondas que se registra en el campo cercano debido a una dislocación ha sido resumida por Okada (1985).

Mediante el ejercicio de la **medición** de estos sucesos, tanto a posteriori como durante la ejecución misma del fenómeno, es posible desarrollar modelos científicos que permiten estimar la magnitud del sismo y la localización del origen.

La **onda sísmica** se registra con equipo científico que miden variables, como la posición, la velocidad y la aceleración. En conjunto permiten tener un mayor conocimiento del fenómeno y también desarrollar proyectos de prevención de catástrofes, entre otros.

También, el cambio físico de la posición de las placas es posible medirlo mediante herramientas geodésicas como equipos GPS. La medición del cambio, a lo largo del tiempo, permite describir la historia y proyectar al futuro. Esta deformación de la tierra ocurre de manera permanente, ocurre una y otra vez en diferentes lugares y con distintas magnitudes.

Cómo ejemplo se puede observar en la figura 2.1 en tres momentos del paso de la onda sísmica desde su fuente (o hipocentro) y luego su desplazamiento a través de una red de sensores (puntos en rojo).

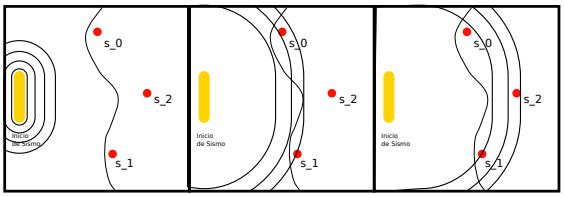

Figura 2.1: Evolución de una onda sísmica detectada por un conjunto de estaciones (simplificado)

En el caso de Chile, es posible mencionar el trabajo inicial de Plafker & Savage (1970), el cual relaciona los movimientos verticales estimados en los puntos de nivelación geodésica y en los desplazamientos horizontales de la red de triangulación con los desplazamientos generados por el gran terremoto de 1960. Mostrando que, mediante la observación de la posición de la tierra a lo largo del tiempo es posible conocer la deformación debido al efecto de los sismos.

Dada la complejidad y magnitud de este fenómeno, que ocurre a una macro escala, es necesario el uso de formas estandarizadas de medición, tanto del tiempo, la posición y otras variables. La creación de redes de sensores y diferentes capas de medición hacen posible mejorar la detección y caracterización de los sismos.

A medida que se ha ido adoptando el posicionamiento geodésico en los estudios sismológicos se ha hecho necesario definir un marco de trabajo en que destaca el glsitrs que es s un sistema espacial global de referencia que rota junto con la Tierra en su movimiento diurno en el espacio. Así como el Marco Internacional de Referencia Terrestre (ITRF) provee el marco de referencia en un conjunto de coordenadas. Con esta constitución general de la referencia geodésica es posible estudiar grandes zonas, siendo posible mantener una consistencia en las mediciones.

En conjunto el sistema general llamado *Geocentric Reference System of the Americas* (SIRGAS), (Báez y cols., 2007), se construye en base a Sistema Internacional de Referencia (ITRS) y Marco Internacional de Referencia Terrestre (ITRF), constituye una forma general de establecer las mediciones geográficos y los fenómenos asociados.

Desde el uso de metodologías locales topográficas hasta ahora en la actualidad que se utilizan herramientas como las redes satelitales de posicionamiento, se ha refinado la precisión con que se mide la deformación de la tierra.

Para los estudios geodésicos se establece el Marco Internacional de Referencia Terrestre (ITRF) que provee posicionamiento, determinación de órbita, orientación de la Tierra, entre otros factores (Jeffrey T. Freymueller & van Dam, 2019). Está conectado con el marco

de referencia celestial. Define el origen en el centro de masa del sistema terrestre, ejes de coordenadas y escala para medir distancias.

Dentro de las iniciativas destacadas se encuentra *Measuring our changing earth* de UNAVCO, que es un proyecto que gestiona la recolección de diferentes tipos de datos, software, estudios académicos y proyectos relacionados a la medición de observables que reflejan los continuos cambios de la superficie de la tierra.

2.2. Formas de expresar el tiempo

Pese a parecer una idea simple y ya definida, el tiempo como concepto es sujeto de especulación y de investigación científica. Se considera en algunas investigaciones el instrumento medidor de tiempo y su efecto en el instrumento mismo.

Dado que este trabajo considera el uso de datos generados por GNSS, sujetos a los efectos de la relatividad, el movimiento de los satélites y la traslación misma de las ondas con que se realiza el cálculo de cada medición, es un factor a considerar.

Más allá de esto, se considera en adelante que la unidad de medida elemental de tiempo es el segundo y se puede expresar como $[s]$.

Hay formas de expresar diferentes medidas de tiempo, aquellas que van por bajo el **segundo** ocupan la notación científica mili, micro, pico, etcétera.

También hay unidades de medida mayores al **segundo** y permiten registrar ventanas temporales más grandes. En esto tenemos la tabla 2.1 que describe estas relaciones.

Tabla 2.1: Equivalencias de unidades de tiempo

Unidad de tiempo	**Equivalencia**
Minuto	60 segundos
Hora	60 minutos
Día	24 horas
Mes	Pueden ser {28, 29, 30, 31} días
Año	Pueden ser {365, 366} días

De esta manera, para registrar un evento o la ocurrencia de un fenómeno se puede expresar el momento en que ocurre como:

$$ao - mes - da\ hora : minuto : segundo.millisegundos \tag{2.1}$$

La forma en que se expresa esta información que representa el tiempo se llama *formato de fecha-tiempo* o *estampa de tiempo*, ya que hay múltiples estilos que van determinados al uso o campo del conocimiento asociado.

Para establecer una referencia absoluta al tiempo que se mide en diferentes lugares del planeta Tierra, se establecen bandas temporales llamadas **husos horario** que mantienen la misma medida en esa zona.

Estos husos horarios se distribuyen longitudinalmente, en que el huso de referencia es **Coordinated Universal Time (Coordinated Universal Time (UTC))** que está asociado a la banda en que se ubica el meridiano Greenwish, dando como referencia al inicio del día.

Esta forma referencial de describir el tiempo fue diseñada y establecida por la **Unión Internacional de Telecomunicaciones (Unión Internacional de Telecomunicaciones (UIT))** con el fin de estandarizar la medición del tiempo y facilitar el uso en las comunicaciones y otras aplicaciones prácticas.

En conjunto con la determinación de la unidad de medida del *segundo* por el Sistema Internacional de medidas Sistema Internacional de Medidas (S.I.), en que es la cantidad de tiempo (longitud) contando las transiciones del átomo *cesio 133*.

> Un segundo es la duración de '9 192 631 770' oscilaciones de la radiación emitida en la transición entre los dos niveles hiperfinos del estado fundamental del isótopo 133 del átomo de cesio (133Cs), a una temperatura de 0° K. [1]

Este logro de determinar de manera exacta la medida de un **segundo** realizado en el siglo pasado (s. XX) ha permitido estandarizar la medición del tiempo y así generar aplicaciones de uso común, científicas e industriales.

En computación la medición del tiempo va asociada a la frecuencia del procesador y su capacidad en bits. El procesador o CPU habilita un contador de *ciclos* que permite controlar la secuencialidad de los procesos. Se puede expresar como una cadena de bits, que es como se almacena en memoria, en forma de **Tiempo Unix** como un número entero cuyo 0 inicia el 1 de enero de 1970, o como una cadena de texto bajo un formato específico [2].

El tiempo que muestra cada computadora se puede sincronizar con servidores que comunican el tiempo asociado a un reloj atómico. De esta manera es posible mantener a nivel global una medida del tiempo lo más exacta posible.

El Tiempo GPS es la medida de tiempo que registran los dispositivos sensores de posición geodésica. Estos consideran con mayor exactitud el tiempo medido, ya que no acumulan para un futuro el diferencial de tiempo que diariamente se acumula para generar el *día extra* que se ha definido cada cuatro años. La medida de tiempo **GPS** se puede convertir a una expresión de tiempo referencial a **UTC** considerando la semana del año, el tiempo en segundos de la semana y el *Tiempo Offset* [3].

$$T_{utc} = f(T_{gps}) + offset \tag{2.2}$$

Es fundamental considerar que la concepción del tiempo y la medición de éste debe estar sincronizada entre todo el conjunto de instrumentos que observan el fenómeno, de esta manera se hace posible determinar diversos patrones existentes, correlaciones y la creación de modelos representativos.

[1] Documento SI medidas https://www.cem.es/sites/default/files/siu8edes.pdf
[2] Tabla conversión para formatos de fecha https://docs.python.org/3/library/datetime.html
[3] Relación Tiempo GPS y UTC https://www.mobatime.com/article/utc-and-gps-difference/

2.3. Teoría Bayesiana

La existencia de fenómenos causales, en los que la probabilidad de que suceda algo dado que ya pasó otra cosa específica se puede describir matemáticamente mediante la expresión de inferencia bayesiana.

La distribución de probabilidad conjunta para A y B, según la regla de Bayes, se puede expresar por:

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(A)P(B|A)}{P(B)} \tag{2.3}$$

También se puede comprender visualmente gracias a un diagrama realizado por Akshay Pachaar en figura 2.2. En que se explica la relación probabilística de dos conjuntos que, al tener elementos en común, presentan una probabilidad de ocurrencia encadenada al otro conjunto.

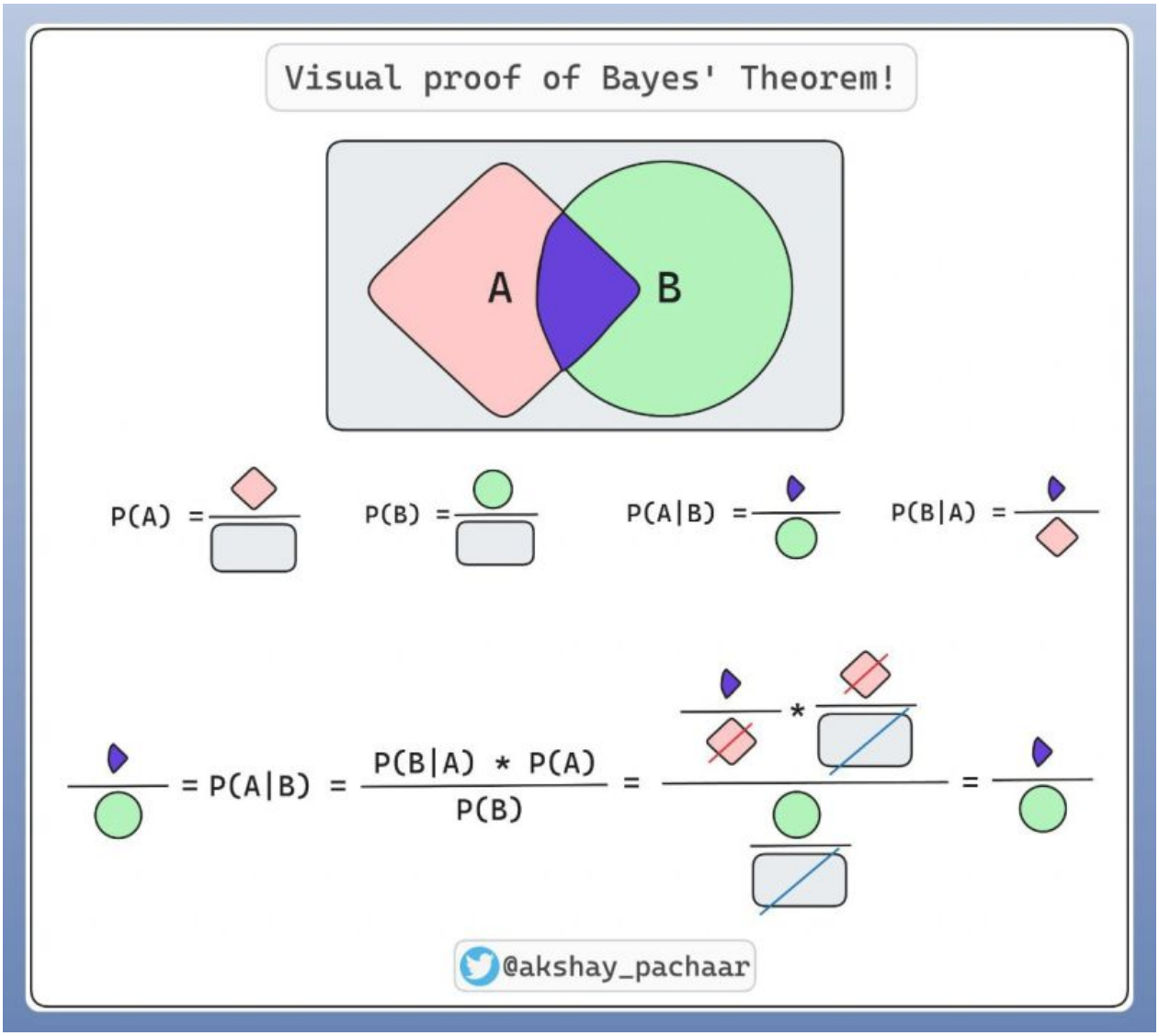

Figura 2.2: Prueba visual del teorema de Bayes

Dónde P(A) es el **prior** y P(B|A) la distribución de **muestreo**. Siendo P(B) la suma de todos los posibles valores de probabilidad para A.

Esta expresión permite modelar sistemas según Gelman et al (Gelman y cols., 2003) de una amplia variedad observable. Define tres pasos en el análisis bayesiano.

- Definir un modelo de probabilidad total
- Cálculo de las probabilidades condicionales o posteriori
- Evaluación de ajuste de modelo y las implicaciones a la distribución resultante.

El análisis bayesiano proporciona una base sólida para la deducción de teorías del aprendizaje basadas en la causalidad, como se discute en Goodman et al. (Goodman y cols., 2011). Esta metodología facilita la creación de abstracciones y generalizaciones del sistema observado.

La descripción de distintos fenómenos se potencia mediante la creación de un lenguaje que abarca diversas hipótesis explicativas. Esto permite una representación precisa y unificada de los fenómenos observados, utilizando un lenguaje de dominio general que puede adaptarse a diferentes contextos.

Definición de un modelo de probabilidad total Este paso implica la identificación de todas las variables relevantes y sus relaciones probabilísticas. Se establece un modelo que describe cómo se espera que las variables interactúen entre sí.

Cálculo de las probabilidades condicionales o a posteriori Utilizando la regla de Bayes, se actualizan las creencias sobre las variables del modelo en función de la evidencia observada. Esto permite recalibrar el modelo en función de nuevos datos, mejorando la precisión de las predicciones.

Evaluación del ajuste del modelo Se analizan los resultados obtenidos para verificar si el modelo representa adecuadamente la realidad observada. Este paso incluye la validación del modelo y la interpretación de las distribuciones resultantes para extraer conclusiones significativas.

La inferencia bayesiana no solo proporciona una estructura matemática robusta para el análisis de datos, sino que también habilita un marco conceptual para el entendimiento de fenómenos complejos. Al permitir la integración de nueva evidencia de manera continua, facilita la mejora constante de los modelos y su capacidad para reflejar la realidad.

En conclusión, la inferencia bayesiana es una herramienta poderosa para la modelización de fenómenos causales. Su capacidad para actualizar creencias en función de nueva evidencia y su flexibilidad para abarcar una amplia variedad de sistemas la convierten en una metodología esencial en el análisis de datos y la construcción de teorías científicas.

2.4. Secuencias serie-tiempo

Una secuencia *Serie Tiempo* consiste en una serie de valores que tienen asociado a cada uno de ellos una *marca de tiempo*. Entonces, es un conjunto de datos que describen un fenómeno durante un periodo de tiempo (R. H. Shumway & Stoffer, 2000).

El valor asociado a cada *marca de tiempo* corresponde a una medida de un mismo tipo de dato. Si se mide temperatura, todos los valores serán de temperatura, si se mide radiación todos los valores serán radiación. Es decir, una secuencia serie-tiempo de una medida será una serie de valores de un tipo de dato, cada uno asociado a una marca de tiempo.

Una secuencia serie-tiempo permite registrar la evolución de un fenómeno a lo largo del tiempo, podría ser el movimiento de un cuerpo o el cambio de temperatura para alcanzar la ebullición. Por lo que se puede decir que un fenómeno puede ser observado y, siendo posible, controlado mediante el estudio de su comportamiento.

Se puede expresar como una serie ordenada de valores.

$$Xs(t) = \{x_s(t=0), x_s(t=1), x_s(t=2), ..., x_s(t=n)\} \tag{2.4}$$

Un registro del tiempo con una mayor densidad significa mayor precisión o sensibilidad de lo que se observa o mide. Sin embargo, en sistemas sujetos a influencias o fenómenos naturales indeterminados, esta anotación se verá afectada por ruido, la medición contendrá además del fenómeno observado la influencia de factores externos no deseados que afectarán a la medición real como tendencias, periodicidades o efectos instrumentales.

La relación secuencial de un fenómeno entre un registro medido y el siguiente consiste en que los valores están relacionados de manera encadenada. Existe una continuidad que se puede expresar como la probabilidad de tomar un valor x_1 dado que el anterior fue x_0 que, en el contexto del fenómeno, se puede entender como un estado normal de operación o un estado sujeto a un estímulo que perturbaría la medición de la secuencia (Visser y cols., 2009).

Por ejemplo, para una secuencia que se describe con un modelo de primer orden, la probabilidad en S_t dados los valores anteriores será la probabilidad en S_t dado el anterior S_{t-1}.

Expresada esta idea en la forma bayesiana.

$$p(S_t|S_1, S_2, ..., S_{t-1}) = p(S_t|S_{t-1}) \tag{2.5}$$

A este modelo se le llama *modelo de cadenas de Markov ocultas (Modelo de Markov Oculto (HMM))* en que el *estado* no es directamente visible, pero las observaciones, que dependen del estado, si lo son. El modelo Modelo de Markov Oculto (HMM) se puede definir en general como una relación de parámetros.

$$\mu = (S, O, A, B, \pi) \tag{2.6}$$

En que $S = \{S_1, S_2, ..., S_n\}$ son los estados ocultos del modelo.

La variable $O = \{o_1, O_2, ..., o_n\}$ es el conjunto de observaciones del modelo.

A es la matriz de transición.

$$A = \{a_{i,j}\} \tag{2.7}$$

Cada parámetro posicional en la matriz es.

$$a_{i,j} = P(s_j|s_i) \tag{2.8}$$

i,j=1,2,...,n
B es la matriz de emisión,

$$B = \{b_{i,j}\} \tag{2.9}$$

Cada parámetro posicional en la matriz es.

$$b_{i,j} = P(o_j|s_i) \tag{2.10}$$

En que los índices corresponden a $i = 1, 2, ..., n$ y $j = 1, 2, ..., m$.
Y π es es el estado inicial, dónde π_i es la probabilidad del estado S_i al inicio.

Dada la naturaleza de este tipo de datos, las serie-tiempo presentan una alta dimensionalidad. Es un desafio clasificar correctamente debido a que manifiestan un basto número de características. Por lo que realizando una aproximación directa no siempre llevará a una mejor solución. En el trabajo de Esmael et al. (Esmael y cols., 2012) se aborda una propuesta por una representación de alto nivel de la secuencia, permitiendo extraer características de mayor orden.

En conjunto a la complejidad inherente de la medición misma de la serie-tiempo que es usual, esta se ve afectada por diferentes elementos naturales o artificales reflejados en un ruido que perturba la medición.

El ruido que agrega perturbaciones a la medición del fenómeno, se caracteriza porque se puede representar como la suma directa a la señal medida S(t). Siendo N(t) todo el aporte en ruido.

$$Y(t) = S(t) + N(t) \tag{2.11}$$

Este ruido puede ser modelado como un ruido blanco con media 0 y varianza σ_w^2. Definido como el valor medido de una variable aleatoria no correlacionada al fenómeno. $iid(0, \sigma_w^2)$.

Si consideramos además la tendencia general *Q(t)* y las periodicidades *W(t)*, se puede expresar de la siguiente manera.

$$Y(t) = Q(t) + W(t) + N(t) \tag{2.12}$$

Una clave importante a considerar es que una secuencia serie-tiempo es una composición de funciones definidas o caracterizadas por su comportamiento particular, por ejemplo la figura 2.3.

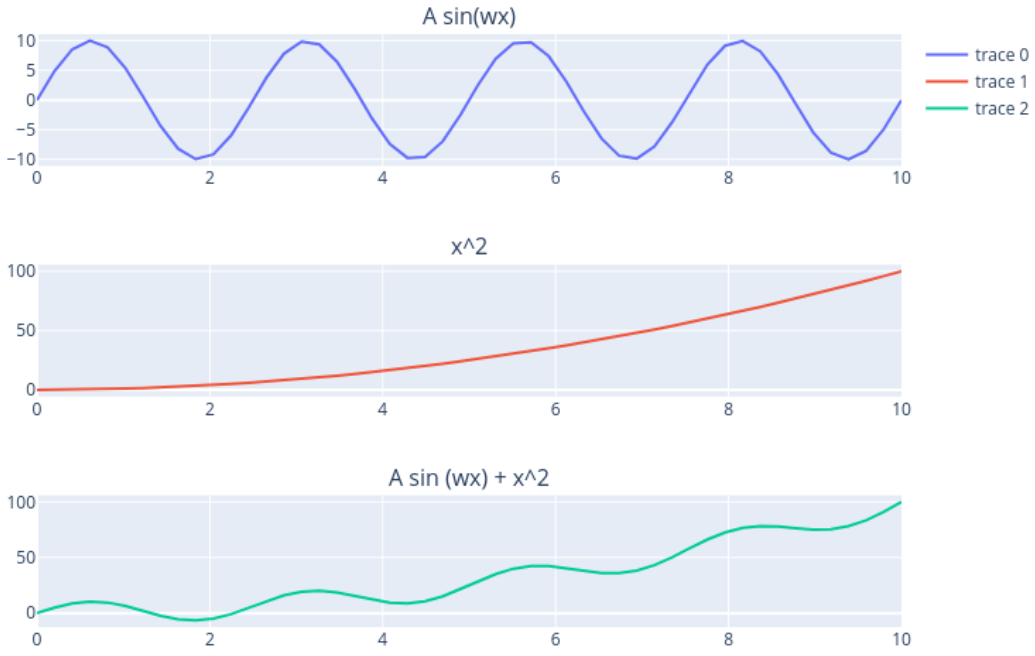

Figura 2.3: Composición de secuencias serie-tiempo

Para separar el ruido de la señal verdadera existen técnicas como la media móvil o técnicas matemáticas aplicadas a señales desarrolladas por Fourier. Este Filtro de Fourier parte de la premisa que una serie-tiempo se puede representar como una suma de secuencias simples desglosada en diferentes frecuencias y de ahí se hace posible filtrar algunas frecuencias específicas.

Entre los tipos de secuencias importantes se tiene que una serie-tiempo estacionaria cuando el comportamiento probabilístico de un tramo es idéntico al de otra tramo.

Esta primer secuencia.

$$S_0 = \{x_{t_1}, x_{t_2}, ..., x_{t_k}\} \tag{2.13}$$

Tiene un comportamiento idéntico al cambio en h momentos.

$$S_h = \{x_{t_{1+h}}, x_{t_{2+h}}, ..., x_{t_{k+h}}\} \tag{2.14}$$

Se relacionan de la siguiente manera:

$$P\{x_{t_1} \leq c_1, \ldots, x_{t_k} \leq c_k\} = P\{x_{t_{1+h}} \leq c_1, \ldots, x_{t_{k+h}} \leq c_k\} \tag{2.15}$$

En el caso de este trabajo, las secuencias serie-tiempo con que se trabaja consisten en que el valor medido es una composición de los valores correspondientes a los tres ejes (X, Y, Z) con una conversión de ejes a $\delta(Norte, Este, Up)$. Es decir, cada sensor tiene una posición de

referencia y respecto a esta se mide la diferencia

Un ajuste para trabajar con serie-tiempo de manera que permita generar estructuras operativas para un modelo de aprendizaje es crear una **Clusterización** por bloque de unidad de tiempo. Para eso se debe considerar la Frecuencia de generación de cada nuevo dato, así si la Frecuencia es un dato por segundo, y se generan datos con precisión de marca de tiempo al Milisegundo se ajusta la marca de tiempo al segundo.

Siendo **t** el tiempo y **h** los milisegundos extra de la medición.

$$Cluster((t+h, x_t)) = (t, x_t) \tag{2.16}$$

Esto es de utilidad, ya que permite crear una Agrupación Matricial dada la medida de un valor de un eje para varios sensores.

$$M_s = \begin{bmatrix} t_0 & x_{t_0} & y_{t_0} & z_{t_0} \\ t_1 & x_{t_1} & y_{t_1} & z_{t_1} \\ . & & & \\ . & & & \\ . & & & \\ t_k & x_{t_k} & y_{t_k} & z_{t_k} \\ t_{k+1} & x_{t_{k+1}} & y_{t_{k+1}} & z_{t_{k+1}} \end{bmatrix} \tag{2.17}$$

Aquí tendríamos una matriz de tres sensores clusterizada por la medición temporal. Es justamente una estructura de datos que puede ser de utilidad para el entrenamiento de modelos de aprendizaje.

2.5. Medidas de dependencia en Secuencias Series-Tiempo

Comprendiendo la serie-tiempo como una serie de valores encadenados, la expresión para la probabilidad de toda la secuencia se puede expresar como la probabilidad conjunta, según Shumway and Stoffer (R. Shumway & Stoffer, 2010).

$$F(c_1, c_2, \ldots, c_n) = P(x_{t_1} <= c_1, x_{t_2} <= c_2, , \ldots, x_{t_n} <= c_n) \tag{2.18}$$

Para el caso en que la ocurrencia de cada x sea independiente, dada una distribución de probabilidad ϕ, la expresión para cada caso se puede escribir como.

$$F(C) = \prod_{t=1}^{n} \phi(x_t) \tag{2.19}$$

Enfocándose en la expresión unidimensional.

$$F_t(x) = P(x_t <= x) \tag{2.20}$$

Y la función de densidad de probabilidad sería.

$$f_t(x) = \frac{\partial F_t(x)}{\partial x} \tag{2.21}$$

La **media** o **valor esperado** para la variable aleatoria.

$$\mu_{xt} = E(x_t) = \int_{-\infty}^{\infty} x f_t(x) dx \tag{2.22}$$

Se define también la Auto-covarianza como el segundo momento.

$$\gamma_x(s,t) = E((x_s - \mu_s)(x_t - \mu_t)) \tag{2.23}$$

Para todo s y t. Este valor mide la dependencia lineal entre dos puntos de la misma serie observada en diferentes tiempos. Aquellas series muy suaves el valor es grande pese a la distancia. Aquellas series que no lo son, presentan un valor bajo para este parámetro.

Otra medida de importancia para el conocimiento de las secuencias serie-tiempo es la **Auto-correlación (ACF)** (ACF), mide el factor de predictibilidad de la serie en un tiempo t, esta función tiene un recorrido de (-1,1).

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\{\gamma(s,s)\gamma(t,t)\}}} \tag{2.24}$$

Si conocemos el valor en algún punto específico s será posible conocer el valor en t mediante la relación $x_t = \beta_0 + \beta_1 x_s$.

La **Covarianza cruzada** de dos series x_t e y_t se puede calcular mediante la relación siguiente.

$$\gamma_{xy}(s,t) = E[(x_s - \mu_{xs})(y_t - \mu_{yt})] \tag{2.25}$$

La **Correlación cruzada** (CCF) para dos series x_t y y_t se puede calcular mediante la

relación siguiente.

$$\rho_{xy}(s,t) = \frac{\gamma_{xy}(s,t)}{\sqrt{\gamma_x(s,s)\gamma_y(t,t)}} \tag{2.26}$$

Estas dos últimas medidas pueden cambiar a lo largo de la serie. Porque dependen de la posición de cada uno, su separación. En caso que la medida de la separación entre ambas permanezca constante se presenta el caso de *Serie de Tiempo Estacionaria*.

2.6. Descomposición modular de una Secuencia Serie-Tiempo

La expresividad de la representación bayesiana de una secuencia serie tiempo como una cadena de Serie de Tiempo Estacionaria, da lugar a la creación de un lenguaje-sistema que permite describir mediante la composición de funciones base (o **kernels**) y operaciones elementales como suma o multiplicación, entre ellas.

Según Goodman et al. (Goodman y cols., 2011) al desarrollar una teoría del aprendizaje, mediante un lenguaje descriptivo basado en funciones, de la serie-tiempo se hace posible reconociendo sus componentes principales. Cada uno de estos componentes y sus combinaciones genera un efecto a considerar que permite modelar distintas características observadas de un fenómeno.

Se define un **kernel** como función de covarianza, es definida positiva y se basa en dos entradas: x y x'. Para generalizar se consideran como vectores en espacio euclideano.

$$Cov[f(x), f(x')] = k(x, x') \tag{2.27}$$

Estas funciones, como se ha mencionado en la sección anterior, son de utilidad para indicar la similaridad entre dos objetos. Es posible reconocer diferentes **kernels** base, tales como:

Exponencial Cuadrada - SE

$$\sigma_f^2 exp(\frac{(x-x')^2}{2l^2}) \tag{2.28}$$

Periodicidad - Per

$$\sigma_f^2 sin^2(2\pi\frac{x-x'}{p}) \tag{2.29}$$

Lineal-Lin

$$\sigma_f^2(x-c)(x'-c) \tag{2.30}$$

Constante-C

Solo es constante a un valor C_0 ...

La combinación de **kernels de covarianza** se fundamentan en las operaciones aditiva y multiplicativa.

Para la suma de kernels.

$$k_a + k_b = k_x(x, x') + k_b(x, x') \tag{2.31}$$

Para la multiplicación.

$$k_a x k_b = k_x(x, x') \times k_b(x, x') \tag{2.32}$$

Con eso, es posible describir secuencias mediante la operación entre diferentes funciones de Covarianza. Permitiendo acciones, como por ejemplo realizar una regresión polinomial, determinar funciones periódicas, observar el crecimiento de amplitud, entre otras características de la secuencia.

Desde la teoría bayesiana, es posible utilizar los kernels para definir **puntos de cambio** que modelan cambios en la secuencia entre distintas estructuras. Por ejemplo, pueden ser definidos mediante el uso de la función *Sigmoidal*

$$CP(k1, k2)(x, x') = \sigma(x)k_1(x, x')\sigma(x') + (1 - \sigma(x))k_1(x, x')(1 - \sigma(x')) = k_1 x\sigma + k_2 x\sigma \tag{2.33}$$

Mediante el uso de kernels también es posible representar características específicas de una secuencia. Dado que son funciones definidas positivas, se pueden expresar como:

$$k(x, x') = h(x)^T h(x) \tag{2.34}$$

Por ejemplo, un kernel estacionario puede ser descrito como una serie de senos y cosenos mediante una representación de Fourier.

Se pueden expresar simetrías e invarianzas. Al codificar en este lenguaje de kernels, es posible mejorar la precisión de los modelos mediante la definición de un correcto *prior*.

Para realizar regresiones sobre secuencias serie-tiempo Duvenaud articula un lenguaje que se describe a partir de los kernels específcios.

Racional Cuadrática-RQ

$$\sigma_f^2(1 + \frac{(x - x')^2}{2\alpha l^2})^{-\alpha} \tag{2.35}$$

Cosenos-cos

$$\sigma_f^2 cos(2\pi\frac{(x - x')}{p}) \tag{2.36}$$

Ruido Blanco-WN

$$\sigma_f^2\delta(x - x') \tag{2.37}$$

Diferentes aproximaciones al problema de la regresión se pueden abordar mediante diferentes expresiones utilizando el mismo lenguaje o conjunto de kernels.

Por ejemplo, en la tesis de Duvenaud se realiza un descripción progresiva del lenguaje para modelar el dataset *Mauna Loa* [fig. 4.3].

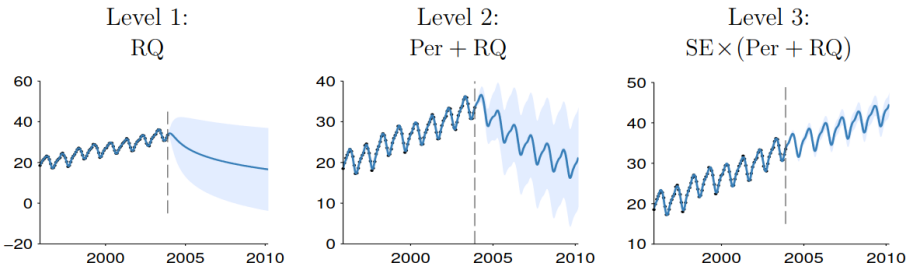

Figura 2.4: Modelación del dataset Mauna Loa mediante la expresión progresiva del uso de kernels

La creación de modelos basados en un lenguaje de kernels, necesita de un proceso de ajuste de parámetros. Si bien tradicionalmente hay parámetros y formas representativas de modelos se determinan por conocimiento experto o experimental, también es posible realizar una aproximación mediante modelos generativos de procesos gausianos.

El uso de redes neuronales profundas de diferentes tipos se pueden comprender como una forma de escoger la función de distribución a priori de un modelo (no lineal). El proceso de aprendizaje se puede interpretar en terminos de conocer la distribución de probabilidad posterior, Mackay, 1998 (MacKay, 1998). Estas redes neuronales se pueden agrupar dentro de la clase de distribuciones de tipo **Procesos Gausianos**

El efecto de cada **Kernel** modifica el modelo de una forma reconocida.

Tabla 2.2: Operaciones y sus efectos

Operación	**Descripción**
Multiplicar por SE	Remueve correlaciones de largo alcance debido al decrecimiento monotónico a 0, convirtiendo cualquier correlación global en local.
Multiplicar por Lin	Es equivalente a multiplicar la función modelada por una función lineal. Causa que la desviación estándar del modelo varíe linealmente, sin afectar la correlación.
Multiplicar por σ	Equivale a multiplicar la función modelada por un sigmoid; la función va a 0 luego de cierto punto.
Multiplicar por Per	Remueve correlación entre pares de valores no cercanos a una distancia periódica, permitiendo variación con cada periodo, pero manteniendo la correlación entre periodos.
Multiplicar por un kernel	Modifica la covarianza en la misma razón de multiplicar por una función al prior.

De manera expresiva, por ejemplo, $PerxLinx\sigma_{1700}$ se puede expresar como función periódica, con variación linear en la amplitud, que aplica hasta 1700.

2.7. Modelos de aprendizaje en Ciencia de Datos

El campo de la ciencia de datos dispone de una amplia colección de modelos de aprendizaje que se usan para abordar una variedad de problemas. Estos modelos se distinguen por su capacidad para ajustar una serie de parámetros durante el proceso de entrenamiento, lo cual está influenciado por varios factores clave:

Tabla 2.3: Aspectos clave en el análisis de modelos de aprendizaje automático

Aspecto	**Descripción**
Características del `dataset`	La cantidad de datos y la estructura de cada dato en particular.
Tipo de aprendizaje	Supervisado, no supervisado o de refuerzo.
Naturaleza del problema	Puede ser clasificación, regresión, clusterización, generativo, entre otros.
Métricas, métodos de optimización y funciones de pérdida	Aspectos cruciales para la evaluación y ajuste del modelo.

Los resultados obtenidos por estos modelos se basan en una interpretación heurística de las relaciones estadísticas entre los datos de entrada y los valores reales. Esta interpretación permite valorar las métricas y determinar la eficacia del modelo para resolver un problema específico.

Las métricas, adaptadas al problema en cuestión, pueden evaluar la precisión o la similitud entre los resultados predichos y los valores reales. Un modelo bien entrenado tiene la capacidad de predecir con cierta certeza los resultados, siempre y cuando la entrada de datos sea similar a la utilizada durante su entrenamiento.

La metodología para desarrollar un modelo de aprendizaje de máquina generalmente sigue estos pasos:

1. Identificación el problema asociado a un fenómeno en un sistema.
2. Extracción y estructuración de datos.
3. Definición de características relevantes.
4. División del dataset en conjuntos de entrenamiento, validación y pruebas.
5. Seleccionar métodos de optimización y funciones de pérdida.
6. Entrenamiento y validación del modelo
7. Evaluación a través de métricas y comparación de resultados.

Esta metodología proporciona un marco de comparación entre diferentes modelos que abordan un mismo problema, facilitando la selección del enfoque más adecuado.

La complejidad intrínseca de un modelo está determinada por la teoría que lo sustenta y su implementación específica en la tecnología disponible. Se observa un progreso significativo en el campo de la ciencia de datos a medida que se incrementa la capacidad computacional disponible.

Los ajustes de regularización son técnicas fundamentales en el entrenamiento de modelos de aprendizaje automático y desempeñan un papel crucial en la prevención del sobreajuste (Overfitting). La regularización busca controlar la complejidad del modelo al agregar términos adicionales a la función de pérdida durante el entrenamiento.

Dos métodos de regularización comunes son la regularización L1 (Regularización Lasso) y la regularización L2 (Regularización Ridge). La regularización L1 tiende a generar modelos dispersos al penalizar algunos coeficientes y forzarlos a cero, lo que facilita la selección de características importantes. Mientras tanto, la regularización L2 reduce la magnitud de los coeficientes, previniendo valores extremos. Además, existen técnicas como la disminución de la tasa de aprendizaje, la detención temprana (Early Stopping) y la regularización por *Dropout* en redes neuronales, todas destinadas a evitar el sobre ajuste al controlar la complejidad del modelo y mejorar su capacidad de generalización en datos no vistos.

Una distinción fundamental se presenta entre los modelos de Deep Learning y aquellos que no dependen de esta técnica está en que los modelos tradicionales de aprendizaje automático, como las máquinas de vectores de soporte (SVM (Support Vector Machine)), árboles de decisión, regresión lineal y métodos de vecinos más cercanos (K-NN (K-Nearest Neighbors)), se basan en técnicas que no requieren de redes neuronales. Estos modelos tradicionales tienden a ser más interpretables y suelen funcionar bien con conjuntos de datos pequeños o medianos.

Por otro lado, el Aprendizaje Profundo (Deep Learning), con su arquitectura de redes neuronales profundas, ofrece la capacidad de aprender representaciones jerárquicas complejas de los datos y ha demostrado ser especialmente efectivo en tareas de reconocimiento de imágenes, procesamiento de lenguaje natural, y en conjuntos de datos de gran escala. Sin embargo, el alto requerimiento computacional y la necesidad de grandes cantidades de datos para un entrenamiento eficaz son consideraciones clave al optar por modelos de **aprendizaje profundo** en comparación con enfoques más convencionales.

2.8. Redes Neuronales

La técnica fundamental que se aplica en este trabajo es la de aprendizaje profundo sobre redes neuronales, por lo que está sección se avoca principalmente en describir las características y conceptos principales, además de las variantes abordadas en esta tesis.

2.8.1. Perceptrón

El Perceptrón es la unidad básica de una red neuronal. Es un concepto que permite hacer una analogía de una función representada como una neurona.

Permite asociar linealmente una serie de entradas $\vec{X} = \{x_1, x_2, ..., x_N\}$ respectivamente con un parámetro ajustable $\vec{W} = \{W_1, W_2, ..., W_N\}$, cuya suma con un valor constante o bias $\vec{b}$ de dimensión 1 (en este caso), permite obtener una salida de la red o predicción.

Las N entradas corresponden en sí a diferentes características de un dato, es decir son N características (Features) que operan sobre la neurona.

Esta salida se opera con una función de activación que permitirá controlar el resultado y ajustarlo al caso de estudio (clasificación, regresión, etc.)

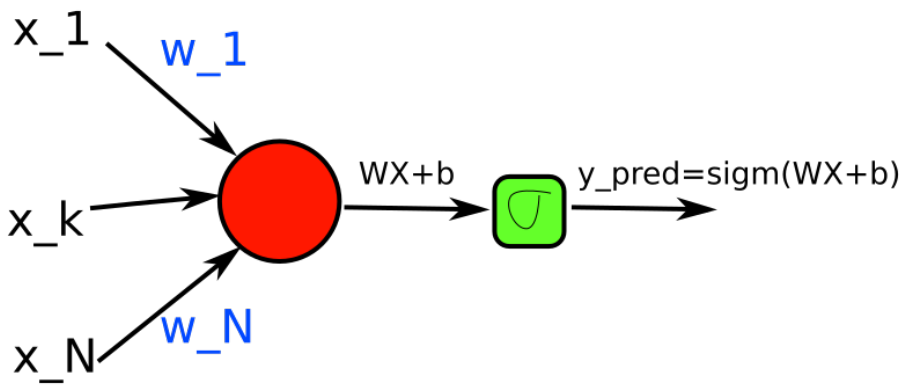

Figura 2.5: Modelo de perceptron con N entradas y función de activación

Este modelo atómico está limitado a entregar un solo valor, que tanto puede ser binario como un número real en el rango de la función de activación, visualmente se puede observar en la figura 2.5.

Dado que la mayoría de los problemas son de una complejidad mayor a la que puede proveer el perceptrón único como solución, es posible componer un arreglo de perceptrones (neuronas o nodos) en lo que que se puede llamar *capa de red*.

2.8.2. Función de activación

La función de activación se puede interpretar como un elemento regulador de la red que se ubica como operador a la salida de diferentes etapas de una red. Permite transformar una operación lineal en no lineal, además existen algunos no derivables y otras derivables.

Las funciones de activación más comunes incluyen la función Sigmoide, que mapea los valores de entrada en un rango entre 0 y 1, adecuada para problemas de clasificación binaria. La función Tangente Hiperbólica (Tanh) mapea los valores de entrada en un rango entre -1 y 1, similar a la función Sigmoide pero con un rango más amplio. Por otro lado, la función ReLU (Rectified Linear Unit) activa neuronas si la entrada es mayor que cero, acelerando el proceso de entrenamiento al evitar problemas asociados al desvanecimiento del gradiente. Estas funciones de activación no solo introducen no linealidad en las redes neuronales, sino que también afectan la capacidad de convergencia, el rendimiento y la estabilidad durante el proceso de aprendizaje.

Entre las funciones de activación conocidas se tienen, entonces:

Entre otras y cada una tiene su uso según el problema en particular.

En resumen, las funciones de activación son elementos esenciales en las redes neuronales, ya que permiten a estas redes aprender y modelar relaciones no lineales en los datos. La elección de una función de activación apropiada depende del problema que se esté abordando y puede tener un impacto significativo en el rendimiento y la capacidad de la red para generalizar patrones complejos en los datos de entrada.

Tabla 2.4: Funciones de activación y sus características

Función	**Descripción**
Sigmoid	Provee resultados de 0 a 1. Fórmula: $\sigma(x) = \frac{1}{1+e^{-x}}$
Tanh	Provee resultados de -1 a 1. Fórmula: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
ReLU	Activa neuronas solo para valores positivos. Fórmula: $\text{ReLU}(x) = \max(0, x)$

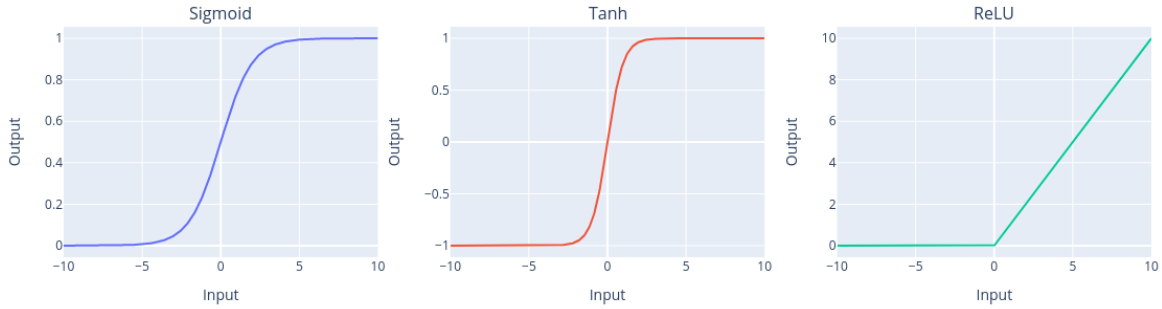

Figura 2.6: Funciones de activación no lineales: sigmoid, tanh, relu

2.8.3. Capa de Red

Una capa de red es una composición en un arreglo ordenado de 'M' neuronas, permitiendo operar M veces las N entradas.

Será necesario identificar cada parámetro como único, ya que la calibración resultante del entrenamiento considera cada uno de estos como un valor específico ya que atiende alguna característica en particular del problema.

La intuición matemática permite construir una representación matricial de esta estructura ordenada observada en 2.7.

La matriz de parámetros.

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,1} & w_{k,2} & \dots & w_{k,N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M,1} & w_{M,2} & \dots & w_{M,N} \end{bmatrix} \tag{2.38}$$

El arreglo de *Bias*.

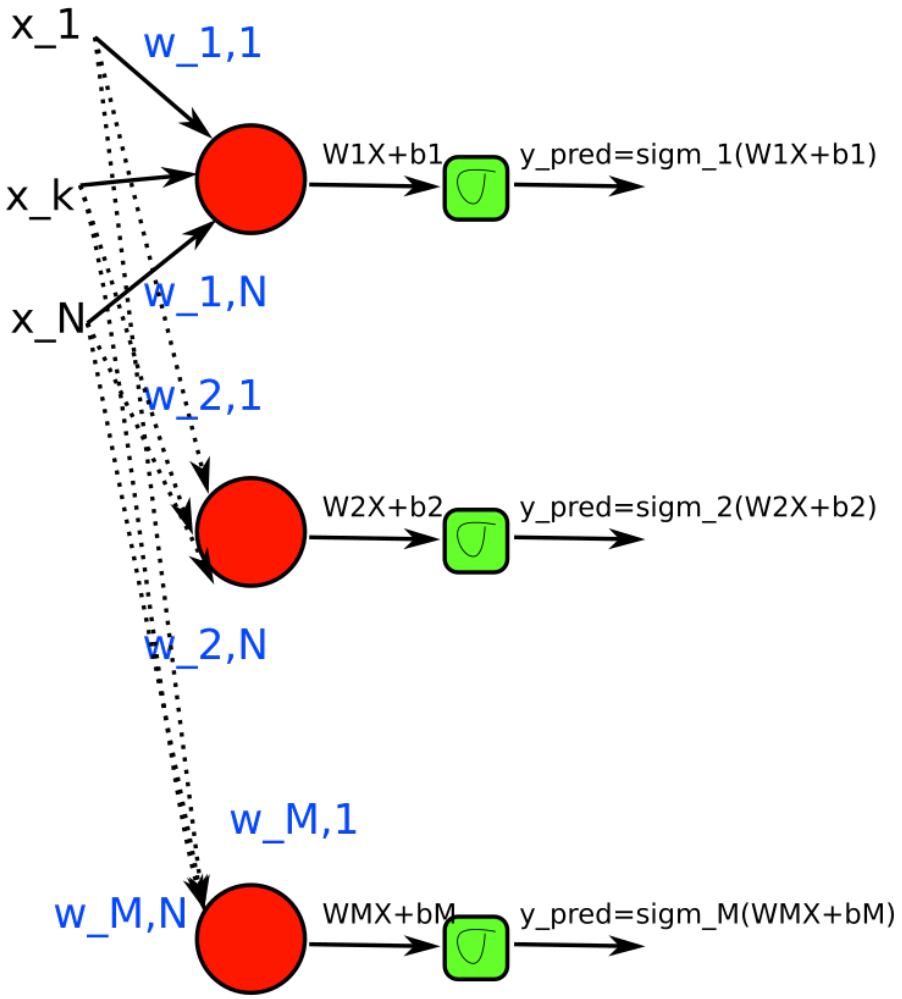

Figura 2.7: Capa de red con M neuronas

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_M \end{bmatrix} \tag{2.39}$$

Con esto tendremos una expresión equivalente matricial a una operación lineal.

$$\vec{z} = W\vec{X} + \vec{b} \tag{2.40}$$

En que, siendo la función de activación la misma para todo el arreglo.

$$\hat{\vec{y}} = \sigma(z) \tag{2.41}$$

Teniendo así una salida no lineal de una capa de red neuronal.

Con este arreglo de neuronas ya es posible crear una red neuronal de una o mas capas, cada una con un conjunto propio de neuronas.

2.8.4. Red neuronal 'Fully Connected'

Una de las redes neuronales más sencillas, dentro del contexto, se puede considerar como la composición de varias capas de arreglos de neuronas interconectadas.

Esto permite operar la entrada en varias etapas y, según los parámetros W y Bias b que se tenga en cada capa se tendrá una salida como resultado.

Cómo se observa en la red (figura 2.8), cada capa tendŕa una cantidad determinada de neuronas, además se pueden caracterizar las capas por su ubicación en:

1. Capa de entrada, toma directamente cada una de las características del dato
2. Capa oculta, son las capas intermedias de la red.
3. Capa de salida, es la capa que entrega el resultado final a la función de activación.

2.8.5. Entrenamiento del modelo

El entrenamiento de un modelo de Deep Learning es un proceso iterativo en el que se ajustan los parámetros del modelo para minimizar una función de pérdida o error, con el fin de hacer que el modelo se adapte mejor a los datos y pueda realizar predicciones precisas.

Aquí hay una descripción detallada del proceso de entrenamiento:

Inicialización : Los parámetros del modelo, como los pesos y los sesgos (bias) en las capas neuronales, se inicializan generalmente de manera aleatoria o utilizando esquemas de inicialización específicos. Esta etapa es crítica ya que la elección de los valores iniciales puede influir en la convergencia y el rendimiento final del modelo.

Paso de forward (propagación hacia adelante) : Durante esta fase, los datos de entrada se introducen en el modelo y se calcula la salida predicha. Las características se propagan a través de las diferentes capas de la red, cada una aplicando transformaciones no lineales mediante funciones de activación, generando así una salida que se compara con la salida real para calcular la pérdida.

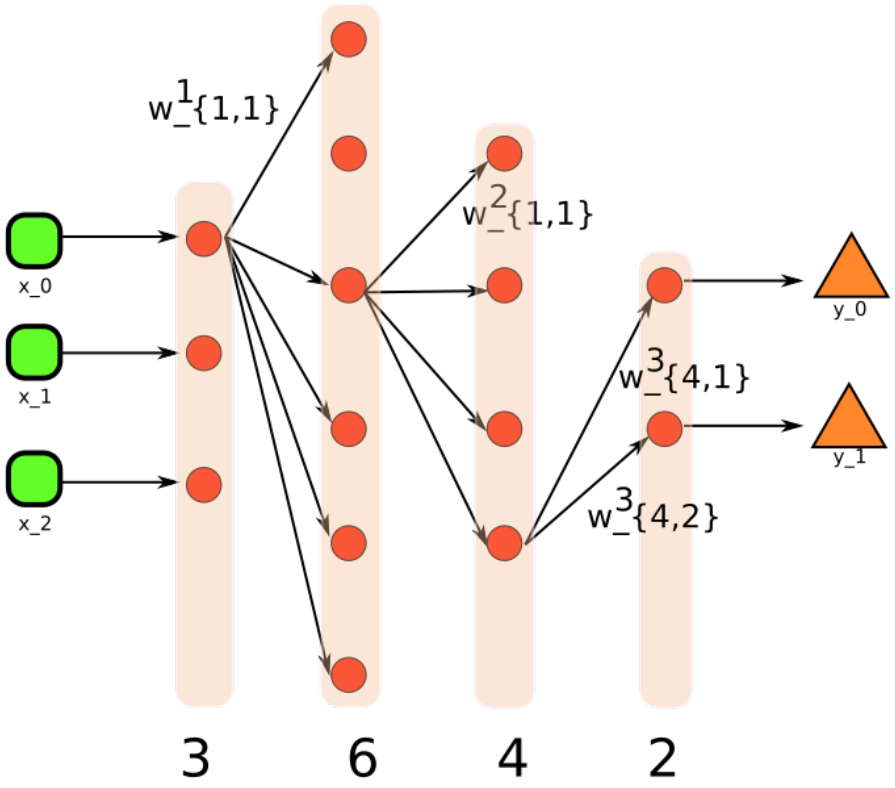

Figura 2.8: Modelo de perceptron con N entradas y función de activación

Cálculo de la pérdida : Se evalúa la discrepancia entre la salida predicha por el modelo y los valores reales utilizando una función de pérdida, que cuantifica la diferencia entre las predicciones y los valores observados. El objetivo del entrenamiento es minimizar esta función de pérdida.

Paso de backward (propagación hacia atrás) : Este es el proceso en el que se calculan las derivadas parciales de la función de pérdida con respecto a cada parámetro del modelo, utilizando el algoritmo de retro propagación del gradiente. El gradiente resultante indica la dirección y magnitud en la que se deben ajustar los parámetros para reducir la pérdida.

Actualización de parámetros : Los parámetros del modelo se actualizan utilizando un algoritmo de optimización, como el descenso de gradiente estocástico (SGD), Adam, RMSProp, entre otros. Estos algoritmos ajustan los parámetros en la dirección opuesta al gradiente, con el objetivo de minimizar la función de pérdida.

Iteraciones (epochs) : El proceso de forward, cálculo de pérdida, backward y actualización de parámetros se repite durante múltiples iteraciones o épocas. En cada época, el modelo ajusta gradualmente sus parámetros para mejorar su capacidad de hacer predicciones precisas.

Evaluación y ajuste : Se evalúa el rendimiento del modelo en un conjunto de datos de validación para verificar su capacidad de generalización. Basándose en esta evaluación, se pueden realizar ajustes en Hiperparámetros, arquitectura del modelo o técnicas de regularización para mejorar el rendimiento.

Finalización del entrenamiento : El entrenamiento se detiene cuando se alcanza un criterio predefinido, como un número fijo de épocas, cuando la pérdida converge o cuando se observa un rendimiento satisfactorio en los datos de validación.

En resumen, el entrenamiento de un modelo de Deep Learning implica ajustar iterativamente los parámetros del modelo utilizando datos de entrenamiento y un algoritmo de optimización para minimizar la función de pérdida, con el objetivo de lograr un modelo que generalice bien a datos no vistos. Este proceso requiere selección cuidadosa de hiperparámetros, regularización adecuada y una observación constante del rendimiento del modelo para obtener resultados óptimos.

2.8.6. Regularización por Dropout

La *regularización* es una técnica que permite aplicar un factor de penalización al cálculo del error. Se puede abordar de diferentes formas y afecta positivamente en los ajuste del modelo.

El llamado *dropout* es un ajuste de regularización que se aplica sobre las neuronas de manera aleatoria.

En pocas palabras, lo que hace es desactivar o no la salida de la neurona en particular, teniendo un efecto positivo en el modelo general ya que entrega un grado de variabilidad, previniendo el sobreajuste (Overfitting).

Esta desactivación ocurre en cada iteración durante el entrenamiento, otorgando una mayor robustez a la red.

En una mirada mas cercana permite que cada neurona aprenda características de una manera mas independiente a las otras. Ayudando así a evitar el sobre ajuste a patrones específicos. Es una herramienta que contribuye a la generalización del modelo.

2.8.7. Función de pérdida

Una vez definida la estructura del modelo, que puede consistir tanto en una simple red fully conected, o una arquitectura compuesta con otros elementos. Un punto de importancia que define la calidad del modelo será la función de pérdida.

La función de pérdida de*Cross Entropy Loss* generalmente se utiliza en problemas de clasificación, donde se tienen múltiples clases y se busca comparar las distribuciones de probabilidad predichas por el modelo con las distribuciones reales de las clases.

En este trabajo la función de pérdida debe expresar la diferencia entre lo que entrega el modelo y el valor real con que se está comparando. Una función de pérdida muy utilizada, dado que tiene una interpretación directa es la del **error cuadrático medio** (MSE) que se expresa de la manera siguiente.

$$C(y, y_{pred}) = \frac{1}{2M} \sum_{\alpha=1}^{M} (y_\alpha - y_{pred,\alpha})^2 \tag{2.42}$$

Que tiene características especiales:

- Cuadrática
- parábola
- sumandos >= 0
- tiene un mínimo

Sin embargo, dependiendo del problema, las funciones que pueden entregar una mejor aproximación a resolver la *similaridad* entre la salida real y la predicha.

Para problemas de serie-tiempo, se dispone de las siguientes funciones de pérdida.

En su lugar, para problemas de regresión en series temporales, algunas opciones comunes de funciones de pérdida son:

Mean Squared Error (MSE) Es la pérdida más común en problemas de regresión. Calcula el promedio de las diferencias al cuadrado entre las predicciones y los valores reales.

Mean Absolute Error (MAE) Calcula el promedio de las diferencias absolutas entre las predicciones y los valores reales.

Huber Loss Es una combinación de MSE y MAE, que es menos sensible a valores atípicos que el MSE.

Smooth L1 Loss Similar al Huber Loss, es una versión suave del error absoluto, que se comporta de manera similar al MSE pero con una respuesta menos extrema a los valores atípicos.

Classic Dynamic Time Warping Classic Dynamic Time Warping (CDTW) Compara las curvas real y predicha, entrega un valor de distancia.

2.8.8. Técnica de Backpropagation

Esta técnica considera que el modelo permite expresar los cambios en sus parámetros como derivadas parciales y usando regla de la cadena.

El primer paso hacia adelante *forward* consiste en obtener los resultados frente a una entrada y calcular la *pérdida o loss*.

$$\begin{aligned} z &= WX + b \\ y_{pred} &= \sigma(z) \\ C = loss(y, y_{pred}) &= MSE(y, y_{pred}) \end{aligned}$$

Siendo derivable la función de pérdida, teniendo además en consideración un *factor de aprendizaje o learning rate*, es posible ajustar los parámetros W de cada neurona. Esta acción de reajuste en base la diferencial se llama *backward*.

Entonces, el ajuste para la neurona i será usando un enfoque de descenso del gradiente:

$$\begin{aligned} \frac{\partial C}{\partial W_i} &= -(y - y_{pred})\sigma'(z)x_i \\ W_i^{t+1} &= W_i - \eta \frac{\partial C}{\partial W_i} \end{aligned}$$

Este procedimiento se realiza de manera iterativa y sobre todo el conjunto en cada etapa. De esta manera, en conjunto con la *loss* el modelo va ajustando su precisión hasta un factor aceptable.

2.9. Algoritmo de optimización

El algoritmo de optimización es la forma que se realiza el ajuste de los parámetros para la siguiente iteración.

2.9.1. Descenso de Gradiente Estocástico (S.G.D.)

El Descenso de Gradiente Estocástico es un algoritmo de optimización utilizado en el entrenamiento de redes neuronales y otros modelos de aprendizaje automático. Funciona ajustando iterativamente los pesos de la red neuronal en la dirección opuesta al gradiente de la función de pérdida con respecto a esos pesos. La idea principal es encontrar mínimos locales o globales en la función de pérdida, lo que representa la convergencia del modelo.

Proceso de actualización El algoritmo comienza con pesos aleatorios y utiliza muestras de datos individuales o mini-batches (subconjuntos pequeños de los datos de entrenamiento) para calcular los gradientes y actualizar los pesos. Esto reduce la carga computacional en comparación con el cálculo del gradiente en el conjunto de datos completo.

Tasa de aprendizaje (Learning Rate) El SGD utiliza un parámetro conocido como tasa de aprendizaje, que controla el tamaño de los pasos que se dan en la dirección del gradiente. Un learning rate demasiado grande puede hacer que el algoritmo oscile alrededor del mínimo, mientras que uno demasiado pequeño puede ralentizar la convergencia.

Desafíos El SGD puede enfrentar desafíos como la convergencia lenta o quedarse atrapado en mínimos locales. Para abordar estos problemas, se han desarrollado variantes del SGD, como el SGD con momentum, que incorpora un término de momentum para acelerar la convergencia y reducir oscilaciones.

2.9.2. Adam (Adaptive Moment Estimation)

Estimación Adam es un algoritmo de optimización adaptativo que combina las ideas del Descenso de Gradiente Estocástico con estimaciones adaptativas de los momentos de primer y segundo orden de los gradientes. Adam ajusta la tasa de aprendizaje para cada parámetro individualmente, lo que lo hace más adaptable a diferentes tasas de convergencia para cada parámetro.

Momentum y adaptación de la tasa de aprendizaje Adam utiliza un momento de primer orden (media móvil de los gradientes) y un momento de segundo orden (media móvil de los gradientes al cuadrado) para adaptar la tasa de aprendizaje. Esto permite que el algoritmo adapte la tasa de aprendizaje para cada peso de la red.

Ventajas Adam suele converger más rápidamente que el SGD estándar en muchos casos y se adapta bien a una amplia gama de problemas de optimización. Sin embargo, su rendimiento puede variar según la naturaleza del problema y la elección de los hiperparámetros.

2.10. Convolución Matricial

La *Convolución* consiste en una operación matemática entre dos funciones ($f_a y f_b$), en que f_b se desplaza en el tiempo operando sobre f_a. El resultado de esto se integra o suma (en caso discreto) para obtener la convolución. La función que se desplaza opera como filtro o *kernel* sobre la función que opera.

La definición formal de convolución está dada por:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.43}$$

En el ámbito del procesamiento de imágenes y redes neuronales convolucionales (CNN), la convolución se emplea principalmente para extraer características de las imágenes. En una convolución entre una imagen y un filtro (también conocido como kernel), el filtro se desliza a través de la imagen aplicando una operación de multiplicación y suma en cada posición. Este proceso genera un mapa de características que resalta patrones específicos, como bordes, texturas o características más complejas, dependiendo del filtro utilizado.

La operación de convolución se lleva a cabo calculando la suma ponderada de los elementos de la imagen y el filtro que se superponen en cada posición. La convolución se caracteriza

por ser una operación local y compartida, lo que significa que los mismos pesos del filtro se utilizan repetidamente en diferentes ubicaciones de la imagen.

En términos más técnicos, en una convolución entre una imagen de entrada y un filtro, se multiplica cada elemento del filtro por los elementos correspondientes de la imagen en una región específica, y luego se suman estos productos para obtener un solo valor en la salida. Esta operación se repite a lo largo de toda la imagen para generar un mapa de características.

Una convolución es una operación entre un bloque de entradas, en este caso el conjunto de señales de la estación y sus vecinas, con uno o más *kernels* que operan como filtros.

Esta operación, en conjunto con las siguientes etapas de la red neuronal, permiten realizar una operación de extracción de características a la entrada. En términos matemáticos, permite pasar desde una matriz $N * M$ a un vector Wx1, en que necesariamente $W >= N * M$,

Cada uno de los *kernel* consiste en una matriz cuyos valores se van ajustando con el entrenamiento del modelo, la convolución sobre cada punto (para no llamarlo Píxel que es algo asociado específicamente a imágenes) de la matriz entregará un valor compuesto que caracteriza la localidad. En este caso se compone de valores en el tiempo de si misma y de las estaciones vecinas en el mismo tramo de tiempo.

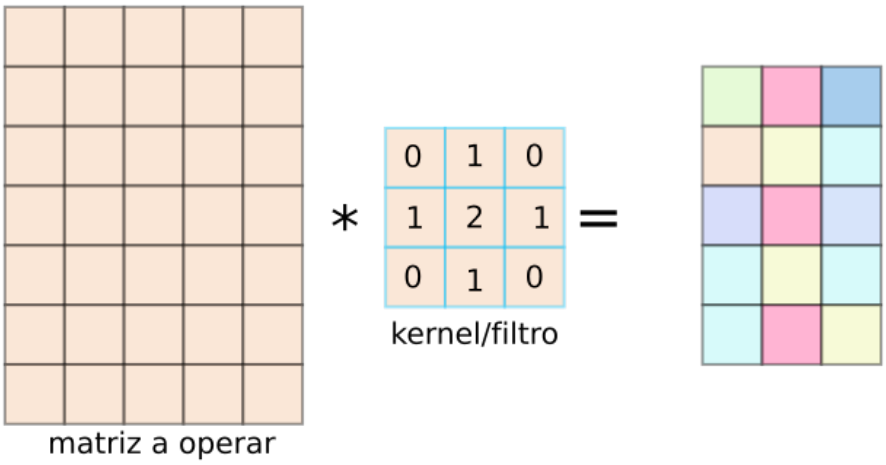

Figura 2.9: Ejemplo de operación de convolución matricial

En el caso discreto, para convolución matricial 2.9, se tiene que cada valor resultante es un punto que se expresa de la siguiente manera:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]) = H * F \tag{2.44}$$

La etapa de Convolución produce un *mapa de activación*, y este puede ser el resultado de una o más etapas de convolución en cascada. Se espera que estas características tengan relación con la periodicidad, comportamientos locales y otros patrones que podrían existir. El resultado final de esta operación debe ser un vector de valores de entrada para la siguiente etapa.

2.11. Redes Neuronales aplicadas a secuencias

El dataset de entrada puede tener diferentes estructuras de datos, una estructura particular es aquella que consiste en una secuencia, ya que cada elemento tiene una posición en un orden. Una secuencia serie-tiempo es de este tipo de datos de entrada, consiste en una tupla de valores **(tiempo, medición)**, en que el orden o posición va dado por el valor del tiempo. También es posible tener secuencias en análisis de lenguaje natural, en que el texto analizado es representado como una secuencia de valores numéricos antes de entrenar el modelo.

Dada una secuencia de entrada, como una serie-tiempo, es posible construir redes neuronales que consideren las correlaciones con el contexto, lo que permite construir modelos robustos frente a sistemas que entreguen este tipo de datos estructurados en secuencias.

2.11.1. Redes recurrentes (RNN)

Una red recurrente que tiene memoria de contexto. Este tipo de redes permite considerar el contexto cercano y lejano en que el valor se posiciona, permitiendo extraer características de interés entre relaciones del valor con los otros valores cercanos, es decir permite capturar regularidades estadísticas, Elman (Elman, 1990).

Implementa una maquina de estados en retroalimentación, de manera que en la entrada considera los nuevos valores y el resultado de lo anterior, permitiendo establecer una estructura de control que mejora la estabilidad y las condiciones de predicción.

En el capítulo 14 de **Neural Network Methods in Natural Language Processing** (Goldberg, 2017) hay una descripcción detallada de esta clase de redes y diferentes configuraciones. La expresión base que define una RNN es la siguiente:

$$\begin{aligned} RNN*(x_{1:n};s_o) &= y_{1:n} \\ y_i &= O(s_i) \\ s_i &= R(s_{i-1},x_i) \end{aligned}$$

En que cada vector se caracteriza por:

$$\begin{aligned} x_i &\in \mathbb{R}^{d_in} \\ y_i &\in \mathbb{R}^{d_out} \\ s_i &\in \mathbb{R}^{f(d_out)} \end{aligned}$$

Lo esencial de una red recurrente es que está construida de tal manera que tiene memoria sobre la secuencia. Se puede expresar como en el siguiente diagrama.

En que R y O son son las mismas funciones a lo largo de la secuencia, pero manteniendo el estado de la computacióna través del vector s_i.

Entre los defectos que tiene esta arquitectura es que acumula el error.

Por esto es que, para mantener un mejor control del entrenamiento en un contexto cuyo valor de la secuencia no solo depende de valores cercanos, sino también lejanos, se creó la arquitectura LSTM con memoria a largo plazo.

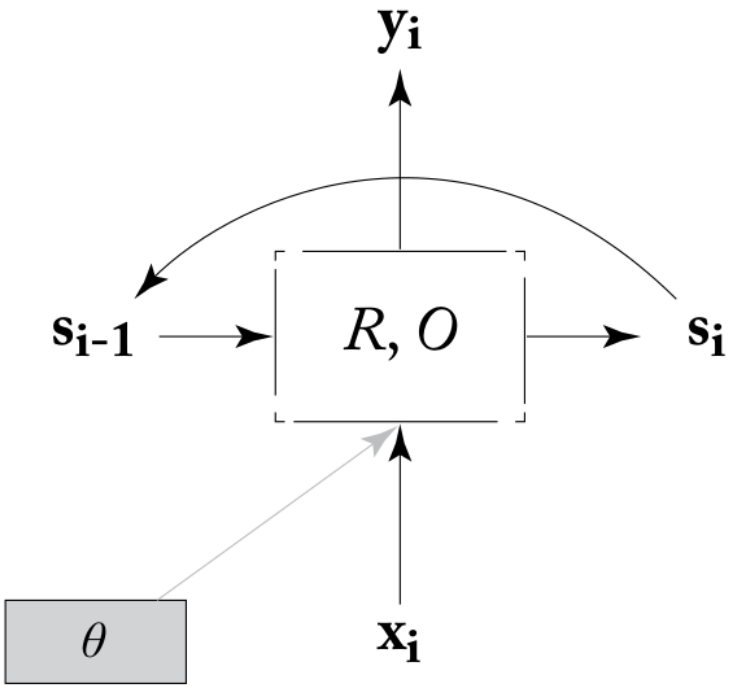

Figura 2.10: Diagrama representación de una RNN (Goldberg, 2017)

2.11.2. Redes RNN/LSTM

Esta arquitectura de red neuronal RNN implementa una solución al problema de *explosividad* o *implosividad*, permitiendo controlar las dependencias lejanas tanto como las cercanas (Hochreiter & Schmidhuber, 1997).

Garantiza que puede aprender a conectar intervalos de tiempo grandes (sobre 1000 pasos), incluso en casos de secuencias ruidosas e incomprensibles, sin perdidas de desplazamiento local, figura (**?**).

Su algoritmo asegura que el error de arrastre se mantenga constante, siendo su característica central que le da ventaja sobre la RNN básica.

Se definen una serie de ecuaciones que determinan el comportamiento de la red.

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}H_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}H_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}H_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}H_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

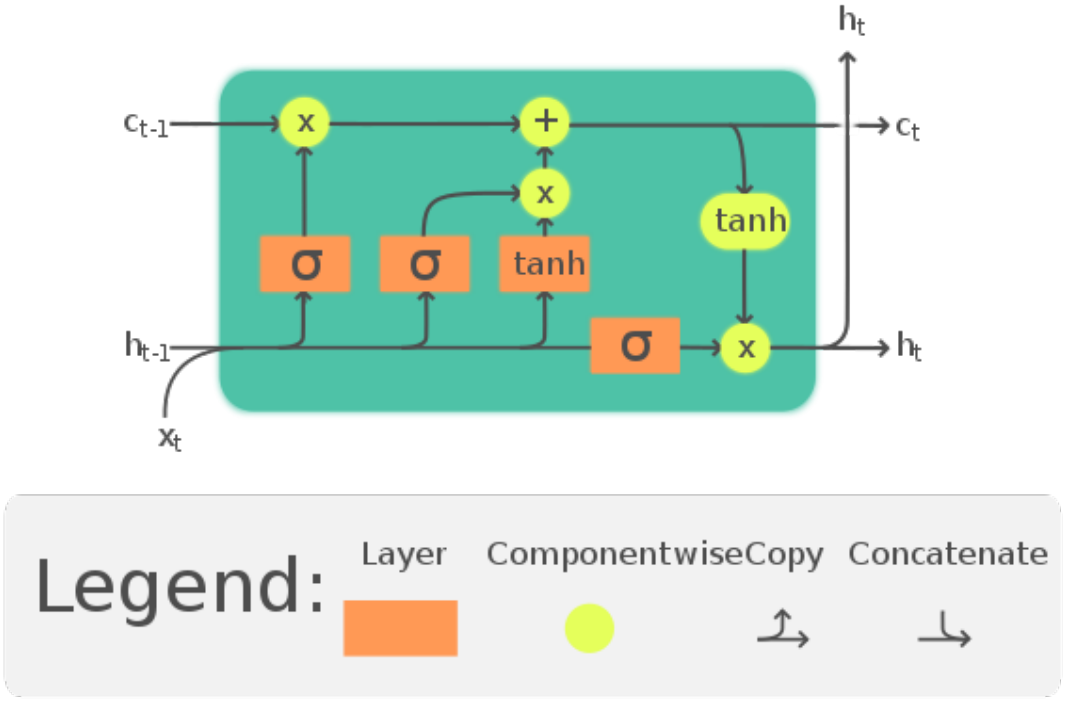

Figura 2.11: Diagrama RNN con LSTM

En que cada variable corresponde a:

h_t es el estado oculto en tiempo 't'.

c_t es la estado de la celda en tiempo 't'

x_t es la entrada en tiempo 't'

h_{t-1} es el estado oculto de la capa en tiempo 't-1' o el estado inicial en '0'

f_t, g_t, o_t son las celdas para entrada y borrado de memoria, y la puerta de salida.

σ es la función sigmoide.

$\odot$ es el Producto de Hadamard

En una red *LSTM* multilayer la entrada x_t^l en la capa l (l>=2) es el estado oculto h_t^{l-1} de la capa previa multiplicada por un **Dropout** (δ_t^{l-1} dónde este es una variable aleatoria de tipo Bernoulli.

Aún así, con las redes LSTM estándar existe un problema que aplica para casos en que se entrena sobre serie-tiempo continua, ocurre que las celdas de memoria tienen un crecimiento lineal, lo que repercute en el deterioro de la celda LSTM.

En año 2000 se presentó una solución a este problema recientemente detectado (Gers y cols., 2000), presentan la inclusión a la celda LSTM de un módulo que *aprende a olvidar*. Es decir, aprende a autoregular el crecimiento lineal de la memoria, permitiendo así entrenar flujos de secuencias serie-tiempo.

2.11.3. Redes Profundas con Transformers

El problema general de maquinas de aprendizaje aplicadas a secuencias de datos que tienen una correlación en su orden es definitivamente amplio. En las RNN, al ser recurrentes, como su nombre indica Recursión, repercute en que la estructura de esta red en la parte recurrente *no puede ser paralelizable*, lo que reduce las capacidades de aplicar modelos usando un alto poder de cómputo en un tiempo razonable.

Con la publicación de *Attention is all you need* (Vaswani y cols., 2023) el año 2017, se abre un nuevo camino con un método que si puede se puede paralelizar, el uso de los llamadas **transformers** harán posible otro abordaje al modelamiendo de redes profundas,.

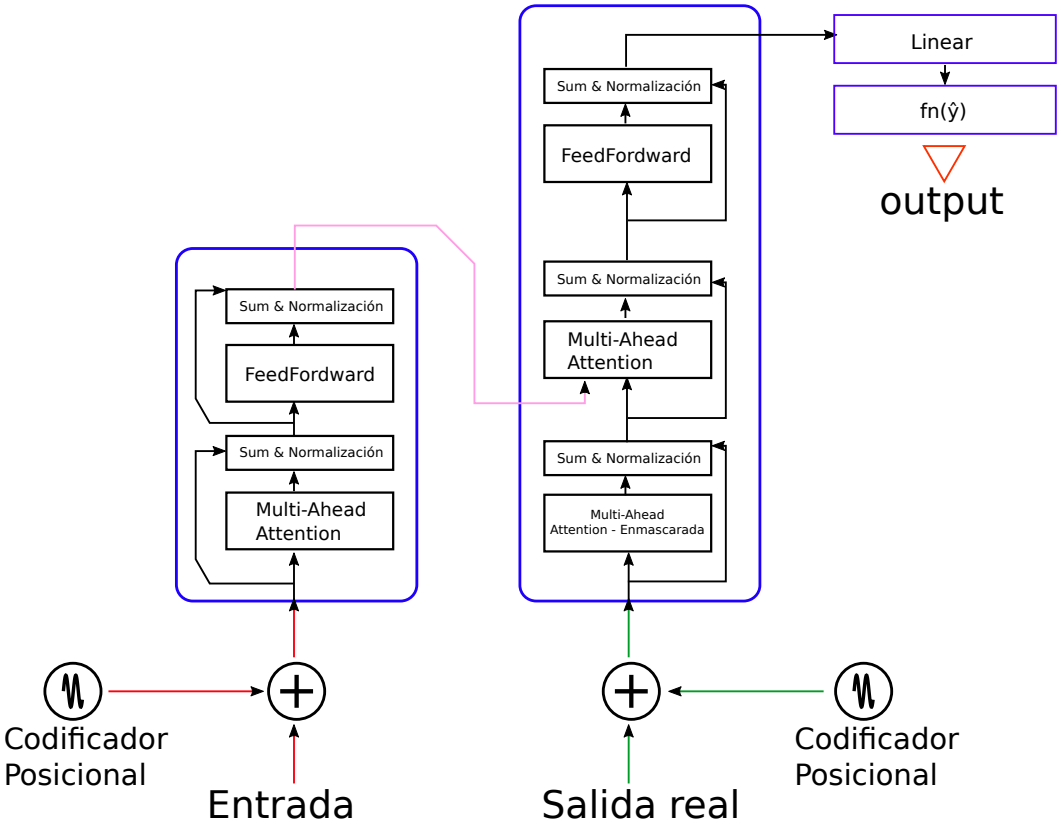

Figura 2.12: Diagrama base de 'attention' con transformer

Entre los componentes de importancia hay que destacar los **codificadores posicionales** que proveen la información posicional de cada elemento, tanto en la entrada como en el arreglo objetivo. De esta manera se alimenta al modelo con los parámetros suficientes para que aprenda el orden y las relaciones de la secuencia [4].

En el caso de estudio se debe entregar una mayor importancia a los últimos segundos de medición, por lo que el Encoder posicional debe además entregar información de la cercanía a los últimos datos, que se puede determinar con una exponencial exp^x añadiendola como una columna extra a las columnas sin/cos básicas, además de agregar una capa entrenable

[4] Introducción a Encoder Posicional https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

reguladora que da un peso a cada parámetro.

Se distinguen con claridad dos elementos en este modelo:

Encoder El bloque que procesa la secuencia de entrada codificándola y entregandola a varias capas de 'attention' que permiten correlacionar en contexto. La definición base consiste en N=6 capas que procesan la secuencia.

Decoder Este bloque incluye lo que resulta del **encoder** en conjunto con la aplicación de 'attention' a la secuencia de salida real. Entregando la salida estimada.La definición base consiste en N=6 capas que procesan la secuencia.

2.11.4. El elemento 'attention'

Consiste en una función que relaciona una consulta 'Q', con un conjunto de pares de llave-valor a una salida. Dónde estos elementos son vectores.

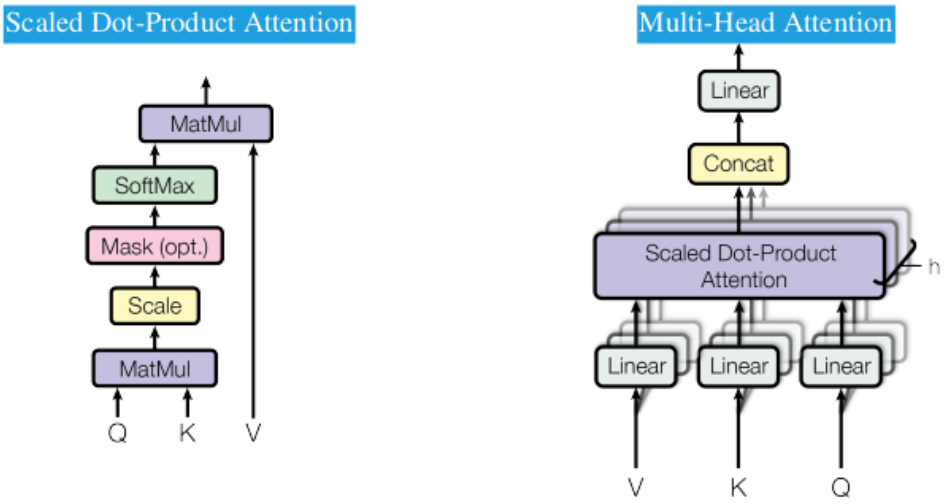

Figura 2.13: Función 'attention'

Siendo d_k el valor de la dimensión de los vector. Se computa el producto punto de la consulta 'Q' con las llaves 'keys'. La 'attention' se calcula como sigue:

Para attention escalada por producto punto:

$$Attention(Q, K, V) = \frac{QK^T}{\sqrt{d_k}}V \tag{2.45}$$

Una forma progresiva de esta misma estrategia es 'multihead-attention', es decir aplicada en paralelo en diversos puntos. Se define por la relación:

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{2.46}$$

En donde $head_h = Attention(QW_i^Q, KW_i^K, VW_i^V)$

Las proyecciones son matrices de parámetros ajustables.

En que cada matriz de parámetros:

- $W_i^Q \in \mathbb{R}^{d_{model} x d_q}$
- $W_i^K \in \mathbb{R}^{d_{model} x d_k}$
- $W_i^V \in \mathbb{R}^{d_{model} x d_v}$

En este modelo las consultas (Queries) vienen de una capa previa 'decoder', las llaves y valores de la memoria vienen de la salida del encoder. Esto permite que cada posición en el decoder atienda todas las posiciones de la secuencia de entrada. Replicando los mecanismos de atención para otros modelos secuencia a secuencia.

Entre los desarrollos contemporáneos que usan transformer y se aplican a soluciones de serie-tiempo para regresión y predicción es posible encontrar el trabajo de Grigsby et al. 2021 (Grigsby y cols., 2021) en que desarrolla predicciones sobre serie-tiempo de multiples variables la red de tipo **Long-Range Transformers** tiene un espectro de aplicación tanto sobre el tiempo como en el espacio, como se observa en la figura 2.14.

Esta investigación interpreta la arquitectura de Transformer como un sistema de paso de mensajes que aprende. Aprendiendo las relaciones espaciales y temporales, al segregar de manera adecuada las celdas de tiempo con las variables que definen las relaciones espaciales. Extrae las ventajas de redes *Secuencia a Secuencia*, así como también redes basadas en *grafos*. También es de destacar que permite establecer relaciones significativas entre los elementos.

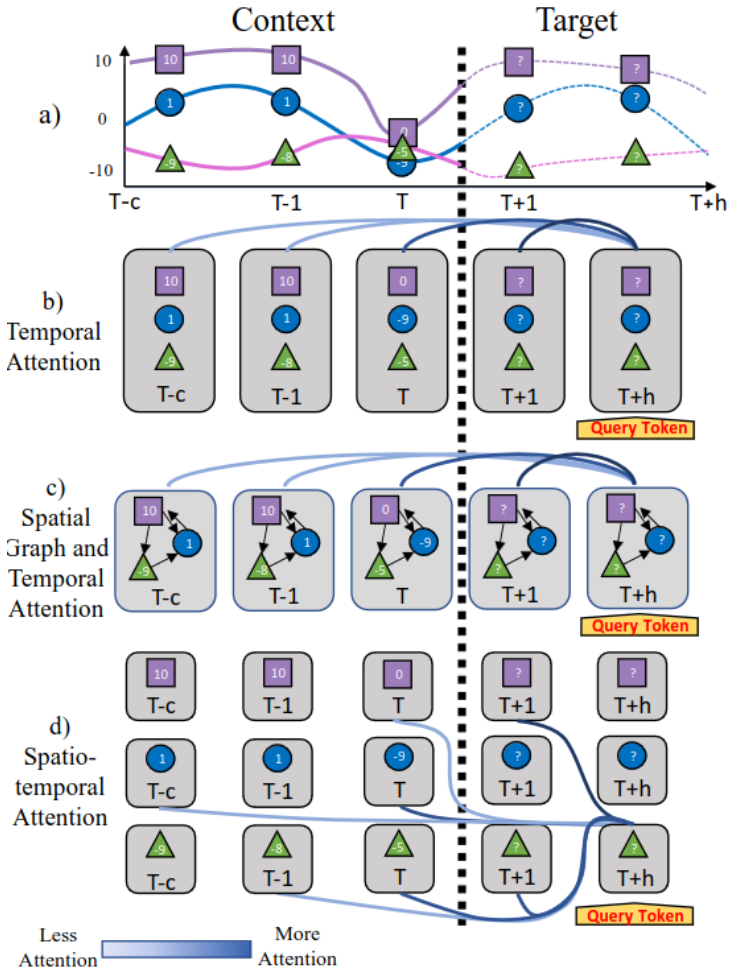

Figura 2.14: Transformer de alcance Espacio-Temporal (Grigsby y cols., 2021)

2.12. Transformada rápida de Fourier

La aplicación de un filtro por transformada de Fourier a la serie tiempo se puede expresar matemáticamente como:

$$_j = \sum_{k=0}^{n-1} x_k e^{\frac{2\pi i}{n} jk}, \quad \forall j = 0, ..., n-1 \tag{2.47}$$

Esto permite representar la señal en el espacio complejo, mediante la suma a groso modo de funciones seno, habiendo un coeficiente por frecuencia presente en la señal.

Esto se aplica a todo el Dataset por cada estación y por cada eje, permitiendo obtener la señal filtrada por FFT.

- $f = \frac{1}{dn}[0, 1, \ldots, \frac{n}{2} - 1, -\frac{n}{2}, \ldots, -1]$ Si es par.
- $f = \frac{1}{dn}[0, 1, \ldots, \frac{n-1}{2}, -\frac{n-1}{2}, \ldots, -1]$ Si es impar

Algoritmo de filtrado.

- Aplica **fft** a la señal
- Computa las frecuencias asociada a cada coeficiente
- Mantener frecuencias suficientemente bajas
- Computar la inversa de fft.

En código python, con **numpy**, esto se puede expresar en la siguiente función.

```
from numpy.fft import *

def filter_signal(signal, threshold=1e8):
    fourier = rfft(signal)
    frequencies = rfftfreq(signal.size, d=20e-3/signal.size)
    fourier[frequencies > threshold] = 0
    return irfft(fourier)
```

Este algoritmo[5] se aplica entonces a cada dataset de estaciones disponible y se almacenará el resultado de la señal filtrada, con el objetivo de usarla para el entrenamiento como la señal decodificada.

2.13. Métricas para Series-Tiempo

Estas métricas utilizan una distancia no euclideana, ya que esta se adaptan mejor al tipo de dato y su estructura. Entre las métricas desarrolladas para serie tiempo se encuentra la llamada *Dynamic Time Warping* (DTW), que consiste en una familia de variantes que abordan de manera adecuada una comparación entre series-tiempo, como se puede observar en el trabajo de Sakoe y Chiba (Sakoe & Chiba, 1978), Salvador, et all (Salvador & Chan, 2004) y últimamente con el de Blondel y Mensch (Blondel y cols., 2021). Esta familia de métricas DTW se diferencian esencialmente en el modo de implementar el algoritmo que permite comparar las series.

[5] Estudio FFT limpieza de ruido https://www.kaggle.com/code/theoviel/fast-fourier-transform-denoising

2.13.0.1. Classic Dynamic Time Warping (CDTW)

Esta es una métrica que permite estimar el error entre dos secuencias serie-tiempo calculando un índice de similitud en base a la forma de cada serie, observando que tan parecidas son.

Dados dos serie-tiempo X=$(x_1, ..., x_n)$ e Y=$(y_1, ..., y_n)$, la matriz de costo C es definida como el costo para cada par de valores combinados [6].

$$C_{i,j} = f(x_i, y_j), \forall i \in 1, .., n, j \in 1, ..., m \tag{2.48}$$

Dónde f es la función de costo, típicamente el cuadrado de la diferencia, $f(x, y) = (x-y)^2$. Una ruta *warp* es una secuencia $p = (p_q, ..., p_L)$ tal que:

- Condición de valor $p_l = (i_l, j_l) \in 1...n \times 1...m, \forall l \in 1, ...L$
- Condición de borde $p_1 = (1, 1)$ y $p_L = (n, m)$
- Monotonía y tamaño del paso, es decir la diferencia entre dos puntos próximos:

$$\begin{aligned} p_{l+1} - p_l \in (0,1), (1,0), (1,1) \\ \forall l \in 1, ..., L-1 \end{aligned} \tag{2.49}$$

El costo asociado a la ruta *warp* se denota por C_p como la suma de elementos de la matriz de costos que pertenece a cada ruta *warp*.

$$C_p(X, Y) = \sum_{l=1}^{L} C_{i_l, j_l} \tag{2.50}$$

El puntaje DTW es definido como el mínimo de estos costos entre todos las rutas *warp* posibles.

$$DTW(X, Y) = min_{p \in P} C_p(X, Y) \tag{2.51}$$

Dónde P es un conjunto de rutas *warp*. Este puntaje puede ser computado utilizado la matriz acumulada, denotada como D, y se define por:

$$\begin{aligned} D_{1,j} &= \sum_{k=1}^{j} C_{1,k}, \forall j \in 1...m \\ D_{i,1} &= \sum_{k=1}^{i} C_{k,1}, \forall i \in 1...n \end{aligned} \tag{2.52}$$

En que, también, para el caso (i,j) se tendrá:

[6] Classic Dynamic Time Warping compared https://tslearn.readthedocs.io/en/stable/auto_examples/autodiff/plot_soft_dtw_loss_for_pytorch_nn.html

$$D_{i,j} = minD(i-1, j-1), D(i-1, j), D(i, j-1) + C_{i,j} \quad \forall i \in 2, ..., n, j \in 2, ..., m \tag{2.53}$$

De aquí, se tendrá que la matriz de costos es el puntaje *Dynamic Time Warping*.

$$DTW(X, Y) = D_{n,m} \tag{2.54}$$

Sin embargo, esta métrica tiene dos desventajas importantes.

- Compleja computacionalmente, $O(nm)$
- No responde a inecuaciones triangulares, por lo que para algoritmos de tipo k-nearest-neighbors debe usarse fuerza bruta.

De aquí, con este método, se derivan diferentes implementaciones que son específicas a su dominio, estas son:

clásica uso genérico para serie-tiempo

sakoechiba define un camino en torno a la curva

itakura define una zona que cambia de ancho según la curva

multiescale hace un seguimiento escalonado de la curva

En el trabajo de Geler (Geler y cols., 2019) se realiza una comparación de las implementaciones Sakoe-Chiba e Itakura. Mientras la primera se encarga de hacer una análisis por banda, Itakura propone una implementación mediante un zona-rombo de detección. Mostrando resultados superiores al primero (ver figura 2.15).

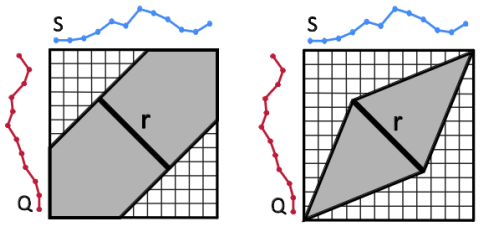

Figura 2.15: Comparativa DTW Sakoe-Chiba vs Itakura. Geler (Geler y cols., 2019)

Con esta técnica como referencia, considerando que si bien es útil para la comparación de series-tiempo, tiene algunas debilidades ya que en sus enfoques más clásicos no es derivable en todo su dominio, además de que puede llegar a locales no-óptimos si se ocupa como función de pérdida. La propuesta de Blondel at al (Blondel y cols., 2021) nombrada como **Soft-DTW divergence** aborda estos problemas y ajustes a la **Soft-DTW** original ya que esta asocia al bias una regularización entrópica.

Características de *Soft-DTW divergence*:

- Es una variante válida de Soft-DTW
- No negativa
- Llega a un mínimo si y solo si ambas curvas son iguales

Esta variante tiene implementaciones en software para **R** y **python**, en la web de Tavenar [7] es posible observar el análisis teórico y de implementación, contiene animaciones y explicaciones de las principales implementaciones de esta familia de métricas.

2.14. Uso de GPU para la resolución de problemas computacionales

Dada la naturaleza compleja de los modelos, estructuras de datos y operaciones matemáticas en que incurren los diferentes algoritmos utilizados tanto para la creación y el uso de modelos de aprendizaje, se hace necesario hacer uso de las herramienta computacionales. Conociendo además las ventajas y desventajas en que se incurre cuando se usa una u otra.

En esta sección se evalúan los principios teóricos para la utilización de computación paralela con procesadores gráficos (GPU). En principio este tipo de procesadores ofrecen la capacidad de realizar cálculos matemáticos matriciales en paralelo, a diferencia de las tradicionales CPU que requieren un procesamiento secuencial. Esto impacta directamente en el tiempo que conlleva realizar una operación incluida en un algoritmo (**?**).

En el presente trabajo se considera el entrenamiento de modelos de máquinas de aprendizaje modeladas como una serie de operaciones que considera matrices, es factible realizar una aproximación al entendimiento del uso de las GPU para mejorar el rendimiento en la resolución del algoritmo.

Teniendo claridad en el uso del recurso GPU, aporta con los fundamentos para de las herramientas de análisis de procesamiento paralelo para lograr una mejor comprensión de la solución propuesta. Esto permite abordar con éxito el problema de procesamiento en tiempo real de un conjunto de datos o problemas que si puedan resolverse bajo un enfoque paralelo.

Para abordar la solución de un problema con el uso de GPU es necesario *identificar el tipo de problema*, ¿se puede usar la GPU o, dada su naturaleza, solo la CPU? Y, a partir de aquí, cómo diseñar la paralelización de alguna etapa del algoritmo.

Entre los conceptos básicos a considerar son:

Concurrencia
Dos o más tareas progresan de manera simultánea

Paralelismo
Dos o más tareas se ejecutan de manera simultánea

Con esto, es posible comprender el concepto de **Computación Paralela** que consiste en resolver un problema de tamaño **n** bajo un dominio de **k** partes ($k \geq 2, k \in \mathbb{N}$) en una cantidad **p** de recursos o procesadores disponibles.

[7] An introudction to Dinamic Time Warping https://rtavenar.github.io/blog/dtw.html

Al ser el problema P paralelizable en D elementos, nombramos a la solución paralela como P_D, como una composición de k problemas.

$$D = d_1 + d_2 + \cdots + d_k = \sum_{i=1}^{k} d_i \tag{2.55}$$

Ahora, la solución aplicada de cada problema significa la aplicación de una **función** a cada parte, determinando que P_D sea un problema de tipo **data paralelo**. Este tipo de problemas habilita una solución con GPU.

$$f(D) = f(d_1) + f(d_2) + \cdots + f(d_k) = \sum_{i=1}^{k} f(d_i) \tag{2.56}$$

De manera análoga, si el problema se resuelve aplicando en cada uno de los diferentes módulos al mismo flujo (stream) de datos, el problema es de tipo **tarea paralelo**. Este tipo de problemas habilita una solución con CPU.

$$D(S) = d_1(S) + d_2(S) + \cdots + d_k(S) = \sum_{i=1}^{k} d_i(S) \tag{2.57}$$

Para medir cómo afecta al rendimiento una solución específica se determina una medida de *rendimiento*, dado un problema de tamaño n y p procesadores.

$$T(n, p) \tag{2.58}$$

Las métricas de **trabajo** (work) y **duración** (span), definen las bases para computar otras métricas como speedup y eficiencia.

La calidad de un algoritmo paralelo se define con las métricas *work* y *span*.

El concepto de *work* se fundamente en el tiempo total para ejecutar un algoritmo paralelo usando un procesador, denotado por T(n,1). El concepto de *span* se define como el tiempo más largo que se necesita para computar con un hilo, denotado por $T(n, \inf)$, siendo equivalente a la medida de tiempo usando infinitos procesadores.

Con estos conceptos, se define una relación que es la *work law* en que se define como el tiempo mínimo de ejecución del algoritmo.

$$T(n, p) \geq \frac{T(n, 1)}{p} \tag{2.59}$$

Y también la *ley de duración* (span law), que establece la relación. Y significa que el tiempo del algoritmo no debe ser menor a la duración para la ejecución mínima con un hilo.

$$T(n, p) \geq T(n, \inf) \tag{2.60}$$

El **speedup** es la medición del cambio, que tan rápido llega a ser, con respecto a una ejecución secuencial.

$$S_p = \frac{T_s(n, 1)}{T(n, p)} \leq p \tag{2.61}$$

En que el numerador es el tiempo para el mejor algoritmo secuencial, y el denominador es el tiempo que toma en ejecutar para p procesadores.

En caso que un problema se componga internamente de diferentes partes, cada una con soluciones secuenciales o paralelas. Se define la ley **Amdahl**:

$$S(p) = \frac{1}{(1-c) + \frac{c}{p}} \tag{2.62}$$

En que c es la fracción del problema que corre en paralelo y (1-c) la parte de se ejecuta secuencial, con p procesadores.

Una implementación adecuada será aquella que compense ejecución secuencial-paralela en cada etapa de manera correcta con los tiempos necesarios de respuesta requeridos.

2.15. Grafos

Para establecer un modelo de Grafos, es crucial comprender conceptos fundamentales dentro de la teoría matemática. En términos generales, un grafo es una estructura conceptual que permite relacionar elementos mediante aristas que definen dichas conexiones. Cada uno de estos elementos se denomina **Nodo** o **Vértice**, y representa la abstracción de un objeto que puede contener información relevante.

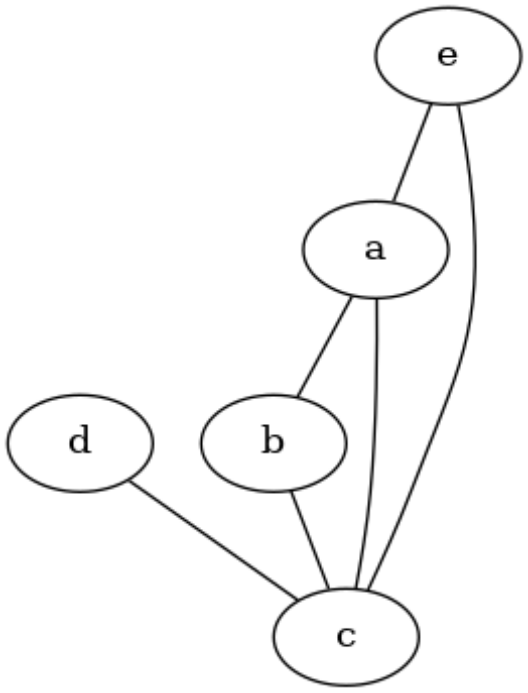

Figura 2.16: Ejemplo de grafo G=(V,E)

Un grafo se puede expresar matemáticamente como G = (V, E), donde G es el grafo, V es el conjunto de vértices, y E es el conjunto de aristas que conectan un nodo con otro en G. La forma en que estos vértices se interconectan en un grafo define diferentes tipos y se caracteriza por propiedades como vecindad, caminos entre nodos, entre otras. Una explicación más detallada de estos conceptos se puede encontrar en la obra "Graph Theory"de Diestel (Diestel, 2017).

En el caso del ejemplo anterior, se tendría:

Para los vértices ($V = \{a, b, c, d, e\}$)

Para las aristas ($E = \{\{a, b\}, \{a, c\}, \{a, e\}, \{b, c\}, \{d, e\}\}$

Esto nos permite definir "nodos vecinos"según un criterio específico, el cual debe ser elegido en función del caso de estudio. Este criterio puede estar basado en *similaridad*, *cercanía por distancia* u otro valor compartido.

Identificar el criterio más adecuado para el caso de estudio puede presentar un desafío interesante. De esta manera, se puede construir un grafo que sea efectivo para el propósito planteado.

En el ejemplo de la figura 2.16, para el grafo G, tendríamos los siguientes conjuntos de "nodos vecinos":

$$
\begin{aligned}
N(a) &= b, c, e \\
N(b) &= a, c \\
N(c) &= b, d, e \\
N(d) &= c \\
N(e) &= a, c
\end{aligned}
\tag{2.63}
$$

De esta manera, si observamos los vecinos al nodo **a** se podrá observar la vecindad, como en la figura 2.17

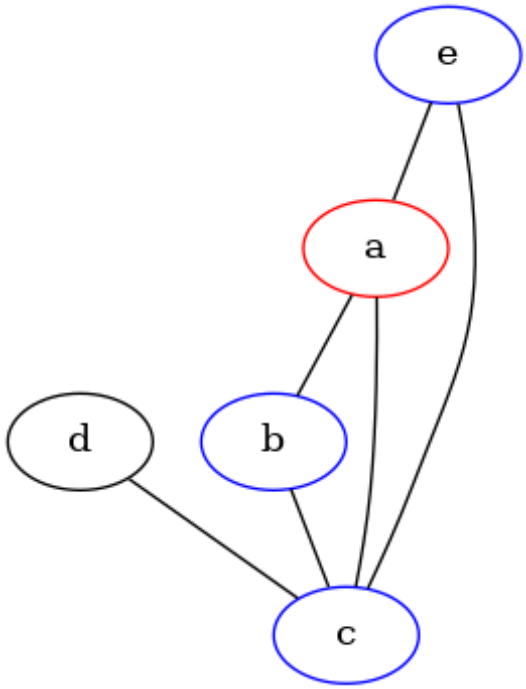

Figura 2.17: Ejemplo de vecinos a 'a' en grafo G=(V,E)

Con estos conceptos fundamentales de la teoría es posible definir las relaciones entre los elementos de un dominio, como la distancia entre las estaciones y definir un concepto de vecindad basado en esta.

2.16. Predicción al futuro cercano o 'forecasting'

Es de interés que el análisis de fenómenos que puedan registrarse mediante la observación de secuencias serie-tiempo permita realizar la predicción del futuro cercano a continuación del tramo registrado. Es decir, basado en los datos de la secuencia, sus patrones, es viable estimar lo que sucede a futuro, en inglés se le llama *forecasting*, como referencia ver el artículo de Boshnakov, Georgi (Boshnakov, 2016).

Una utilidad aplicable al monitoreo sismológico es poder levantar alertas de emergencias frente al gatillante de un posible evento. Comparando la predicción con lo que sucede en el momento se puede evaluar la divergencia, este parámetro se debe establecer en base a una medida y definir valores *checkpoint* que gatillen diferentes mensajes.

Realizar *forecasting* cuantitativo considera la elaboracíon de un modelo que comprenda los diferentes patrones expresados en los datos y las relaciones entre ellos. Mientras más exhautivo sea el registro se puede obtener un modelo más preciso. De esta manera se permite al modelo proyectar a futuro, basado en estos patrones, la estimación de lo que podría suceder.

Entre los conceptos de importancia en la realización de una predicción *forecasting* es posible encontrarse en tabla 2.5.

Tabla 2.5: Definiciones relacionadas con la predicción

Término	**Definición**
Error de predicción	Es la estimación de la precisión de la predicción.
Horizonte de predicción	Es la definición de los periodos en que se realiza.
Intervalo de predicción	Es la frecuencia, cada tanto tiempo, se realiza la predicción.

El proceso de construir e implementar el modelo de predicción consiste en:

1. Definición del problema
2. Recolección de los datos
3. Análisis de los datos
4. Selección de modelo y ajustes
5. Validación del modelo
6. Despliegue del modelo
7. Monitoreo de rendimiento del modelo

Cada uno de estos pasos se puede visualizar en un diagrama (figura 2.18).

2.16.1. Definición del problema

Debe desarrollar un conocimiento de cómo la predicción será utilizada. Se define la periodicidad, métricas y factores de interés para la toma de decisiones que permitan llevar una comprensión acabada de la predicción.

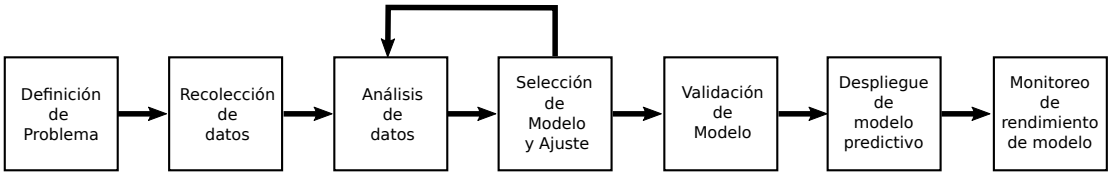

Figura 2.18: Diagrama del proceso de predicción 'forecasting' (Boshnakov, 2016)

2.16.2. Recolección de los datos

Consiste en la obtención de las variables relevantes que permitan definir la predicción. Sean secuencias de una estación o un grupo, datos de posicionamiento y otra información relacionada.

2.16.3. Análisis de los datos

Previo a la selección del modelo predictivo a utilizar se hace necesario realizar un análisis visual y estadístico de los datos, establecer correlaciones, agrupaciones o *clustering*, reconocer patrones importantes, componentes cíclicos y evoluciones del fenómeno observado, etc.

2.16.4. Selección del modelo y ajustes

Consiste en escoger uno o más modelos y ajustes de modelos a los datos. Ajustar significa estimar ciertos parámetros que definen el modelo, Hiperparámetros en el caso de máquinas de aprendizaje.

2.16.5. Validación del modelo

Consiste en la evaluación del modelo para determinar si se comporta con un rendimiento adecuado o un cierto nivel de rendimiento mediante alguna métrica conocida.

Una técnica recomendada es tomar todo el corpus de datos y dividirlo en grupos para entrenamiento, pruebas y validación. Es clave aquí reconocer que, en el caso de redes de aprendizaje profundo, la retropropagación (*backpropagation*) toma una importancia fundamental en el ajuste del modelo durante el entrenamiento.

2.16.6. Despliegue del modelo

Consiste en preparar el modelo para operar y ser utilizado con el fin que fue diseñado. Es importante asegurarse que el o la usuario tengan un entendimiento del uso del modelo y cómo producir las predicciones. El mantenimiento del modelo permite asegurar que se continue la operación.

2.16.7. Observación del rendimiento del modelo

Una vez el modelo entra en operación se debe asegurar que el rendimiento de las predicciones se mantengan, sea con un plan de re-entrenamiento periódico, extensión del entrenamiento, complementar con herramientas de visualización, generación de notificaciones o

alertas.

2.16.8. Consideraciones estadísticas

Un valor de importancia para evaluar la precisión de la predicción es el **error** y se calcula mediante la diferencia entre el valor predicho (y en un tiempo anterior **t -** τ) y el valor real en **t**.

$$e_t(\tau) = y_t - \hat{y}_t(t - \tau) \tag{2.64}$$

Entonces, el error entre la medida actual y un paso atrás (1).

$$e_t(1) = y_t - \hat{y}_t(t - 1) \tag{2.65}$$

La diferencia entre la observación y_t y el valor obtenido por ajuste $\hat{y}_t$ es llamado valor **residual**.

$$e_t = y_t - \hat{y}_t \tag{2.66}$$

Ambos conceptos de **error de predicción** y **valor residual** se diferencian en que este último corresponde a la diferencia de predicción del modelo con datos históricos, que se ajustan mejor a una predicción real.

Usualmente, como se ha mencionado en las secciones anteriores, una serie tiempo presenta ruido, por lo que es posible realizar un filtrado de este mediante un suavizado. Permitiendo así suavizar la curva. Diversas técnicas se pueden aplicar, pero una de las más conocidas y sencillas de utilizar es el promedio por ventana móvil.

$$M_T = \frac{1}{N} \sum_{t=T-N+1}^{T} y_t \tag{2.67}$$

Esta técnica de análisis, junto con otras de mayor complejidad, permiten ir descubriendo diversos patrones que se manifiestan en el registro de la secuencia, como periodicidad o estacionalidad.

La Estacionalidad en Series de Tiempoimplica un cierto equilibrio o estabilidad en los datos. En el caso de datos de posicionamiento geodésico, una estación fija emitirá constantemente una serie-tiempo estacionaria, salvo casos especiales como movimientos sísmicos.

Se determina una media constante por.

$$\mu_y = E(y) = \int_{-\infty}^{+\infty} y f(y) \partial y \tag{2.68}$$

Y la varianza constante, definida cómo.

$$\sigma_y^2 = Var(y) = \int_{-\infty}^{+\infty} (y - \mu)^2 f(y) \partial y \tag{2.69}$$

Estos valores, extraídos de una muestra, se utilizan para estimar los parámetros. Si las observaciones en la serie tiempo son discretas, como es en el caso más común para sensores digitales.

Se determina una media constante por.

$$\bar{y} = \mu_y = \frac{1}{T}\sum_{t=1}^{T} y_t \tag{2.70}$$

Y la varianza constante, definida cómo.

$$s^2 = \hat{\sigma}_y^2 = \frac{1}{T}\sum_{t=1}^{T}(y_t - \mu)^2 \tag{2.71}$$

2.16.9. Autocovarianza y autocorrelación

Generar un modelo que pueda predecir a futuro se sustenta en que los datos de entrenamiento están entrelazados, esto se puede conocer matemáticamente al establecer la **autocovarianza** y **autocorrelación** de la secuencia, en sí misma y con otras variables.

Primero, si la serie-tiempo es estacionaria significa que la distribución de probabilidad conjunta de dos observaciones, sean y_t e y_{t+k} será la misma para dos periodos de tiempo separados por k.

La covarianza dada por esta relación en **k** se puede expresar con la siguiente fórmula.

$$\gamma_k = Cov(y_t, y_{t+k}) = E[(y_t - \mu)(y_{t+k} - \mu)] \tag{2.72}$$

La colección de los diferentes valores γ_k, con k=0,1,2,3..., se llama **función de autocovarianza**. Siendo para k=0, la varianza.

El **coeficiente de correlación** con el *lag* k se define por.

$$\rho_k = \frac{E[(y_t - \mu)(y_{t+k} - \mu)]}{\sqrt{E[(y_t - \mu)^2]E[(y_{t+k} - \mu)^2]}} = \frac{\gamma_k}{\gamma_0} \tag{2.73}$$

Y la colección de valores ρ_k, se llama la **función de autocorrelación (ACF)**. Esta función es independiente de la escala de la medida de la serie, siendo adimensional.

Para estimar la autocovarianza de manera discreta y obtener las ACFs para una serie-tiempo, la relación se expresa por.

$$c_k = \hat{\gamma}_k = \frac{1}{T}\sum_{t=1}^{T-k}(y_t - \bar{y})(y_{t+k} - \bar{y}) \tag{2.74}$$

En que la ACF es estimada por la función de muestra de autocorrelación.

$$r_k = \hat{\rho}_k = \frac{c_k}{c_0} \tag{2.75}$$

Para establecer matemáticamente que una secuencia es estacionaria, se define **variogram**, que es la medida de las varianzas de las diferencias entre las observaciones con lag k. Expresado en la ecuación.

$$G_k = \frac{Var(y_{t+k} - yt)}{Var(y_{t+1} - y_t)} \tag{2.76}$$

Estos valores pueden ser visualizados en función del lag k. Si se de el caso de estacionalidad, la función G_k converge asintóticamente.

$$G_k = \frac{1-\rho_k}{1-\rho_1} \to \frac{1}{1-\rho_1} \tag{2.77}$$

Estimando las varianzas, teniendo en consideración la serie tiempo discreta.

$$d_t^k = y_{t+k} - y_t\bar{d}^k = \frac{1}{T-k}\sum d_k$$

Se estima la varianza de la diferencia por.

$$s_k^2 = \frac{\sum_{t=1}^{T-k}(d_t^k - \bar{d}^k)^2}{T-k-1} \tag{2.78}$$

De aquí que el la muestra del variogram estará dada po.

$$\hat{G}_k = \frac{s_k^2}{s_1^2} \tag{2.79}$$

2.16.10. Ajustes de tendencias y fenómenos no estacionarios

.

En el caso de que se observe una secuencia serie-tiempo que sea una composición de tendencias y estacionalidad, es posible realizar una separación de ambos efectos permitiendo modelar el fenómeno.

Una herramienta de utilidad para extraer las tendencias o fenómenos no estacionarios es la diferenciación.

$$x_t = y_t - y_{t-1} = \nabla y_t = (1-B)y_t \tag{2.80}$$

En que **B** se considera un operador de retroceso y permite definir diferenciación a diferentes pasos operando la expresión en potencia.

$$x_t = \nabla^2 y_t = (1-B)^2 y_t = y_t - 2y_{t-1} + y_{t+2} \tag{2.81}$$

Si se desea elimitir la periodicidad, entonces conocida por **d**.

$$\nabla_d y_t = (1-B^d)y_t = y_t - y_{t-d} \tag{2.82}$$

Esta herramienta matemática permite descomponer la serie tiempo en secuencias identificables por sus características de manera separada. Así como también permite realizar diversas operaciones de desplazamiento.

$$\nabla^d = (1-B)^d \tag{2.83}$$

Usualmente basta con aplicar una a dos diferencias para desacoplar las tendencias más importantes. Así como en casos de periodicidad es posible extraer la diferencias por.

$$\nabla_d y_t = y_t - y_{t-d} \tag{2.84}$$

2.16.11. Evaluación y monitoreo del modelo de predicción

Para evaluar el rendimiento del modelo se establecen diversas métricas estadísticas que operan sobre el error entre lo predicho y la medición real.

Utilizando el error hacia un paso.

$$e_t(1) = y_t - \hat{y}_t(t-1) \tag{2.85}$$

La media del error está definida por.

$$ME = \frac{1}{n}\sum_{t=1}^{n} e_t(1) \tag{2.86}$$

El error de media absoluto.

$$MAD = \frac{1}{n}\sum_{t=1}^{n} |e_t(1)| \tag{2.87}$$

El error cuadrático medio.

$$MSE = \frac{1}{n}\sum_{t=1}^{n} [e_t(1)]^2 = \hat{\sigma}^2_{e(1)} \tag{2.88}$$

El valor esperado de MSE para un modelo correcto, debe ser cero o cercano a cero. Dado que la técnica de producción debe producir predicciones imparciales.

El MSE es un estimador directo de la varianza para un paso del error. Si el error está normalmente distribuido, hay una relación entre MSE y MAD.

$$MSE = \sqrt{\frac{\pi}{2}} MAD \approx 1.25 MAD \tag{2.89}$$

Estas métricas son dependientes de la precisión de la predicción. Se expresan en términos de la unidad de medida. Existe una métrica adimensional que es el **error de predicción relativo.**

$$re_t(1) = (\frac{y_t - \hat{y}_t(t-1)}{y_t})100 \tag{2.90}$$

Con este parámetro definido se puede calcular el valor medio del error de predicción relativo (MPE).

$$MPE = \frac{1}{n}\sum_{t=1}^{n} re_t(1) \tag{2.91}$$

O la media absolute del error de predicción relativo (MAPE).

$$MAPE = \frac{1}{n}\sum_{t=1}^{n} |re_t(1)| \tag{2.92}$$

2.16.12. Aproximación general a la selección del modelo

La base para seleccionar el mejor modelo es definir correctamente el erro, luego establecer las métricas adecuadas según el caso de estudio.

El procedimiento general para definir un modelo predictivo consiste en los siguientes pasos.

1. Graficar serie-tiempo y determinar características principales, como tendencias o estacionalidad.
2. Separar las tendencias y estacionalidades, ajustando el modelo
3. Desarrollar el modelo predictivo sobre los residuales
4. Validar el rendimiento del modelo
5. Obtener el error al diferenciar entre la medición real y la predicción
6. Analizar las periodicidades
7. Si la intervalos de predicción es la deseada, construir los intervalos de predicción para los residuales y aplicar la reversa de las transformaciones.
8. Desarrollar e implementar un sistema de monitoreo del modelo de predicción.

2.17. Comparación de modelos

Al momento de evaluar entre dos o más modelos que realizan la predicción sobre el mismo sistema, se hace necesario establecer algunos parámetros o medidas de comparación. Puede ser que un modelo tenga un buen ajuste a datos históricos pero que no produzca una buena predicción. Ajustarse con exactitud a los datos históricos produce un sobre ajuste, así como también incluir demasiados parámetros (más de lo necesario) para mejorar el ajuste.

En general el modelo que mejor se aproxima a un buen resultado será aquel que entregue la menor desviación estándar para el error de un paso durante el proceso de validación. Un método avanzado, como se ha visto en una sección anterior, es evaluar la similitud por DTW.

Una primera referencia de métrica es el error cuadrático medio.

$$s^2 = \frac{\sum_{t=1}^{T} e_t^2}{T - p} \tag{2.93}$$

En que T son los periodos de datos usados para ajustar un modelo con **p** parámetros, siendo e_t el residual del modelo ajustado. Este valor es solo la varianza de la muestra de los residuales y es un estimador de la varianza del error.

Otro criterio de interés es el estadístico R^2.

$$R^2 = 1 - \frac{\sum_{t=1}^{T} e_t^2}{\sum_{t=1}^{T} (y_t - \bar{y})^2} \tag{2.94}$$

En que el denominador es la suma del los cuadrados de las diferencias de las observaciones, no dependiente del modelo. El numerador es la suma cuadrática de los residuales. Sabiendo además que seleccionar un modelo que maximiza R^2 es equivalente a minimizar la suma cuadrática de los residuales. Un valor alto de este parámetro indica un buen ajuste a los datos históricos. Sin embargo, confiar solo en este valor corre el riesgo de caer en sobre ajustes. Por lo que un mejor criterio es el R^2 **ajustado**.

$$R^2_{adj} = 1 - \frac{s^2}{\sum_{t=1}^{T} (y_t - \bar{y})^2 / (T - 1)} \tag{2.95}$$

Con esto, se tienen métricas de mayor importancia para comparar la efectividad de modelos son **Criterio de Información Akaike (Criterio de Información Akaike (AIC))** y el **Criterio de Información Schwarz Bayesiano (Criterio de Información Schwarz Bayesiano (BIC))**.

Para AIC se tiene:

$$AIC = ln\left(\frac{\sum_{t=1}^{T} e_t^2}{T}\right) + \frac{2p}{T} \tag{2.96}$$

Para BIC se tiene:

$$BIC = ln\left(\frac{\sum_{t=1}^{T} e_t^2}{T}\right) + \frac{pln(T)}{T} \tag{2.97}$$

Estos dos criterios penalizan la suma de los cuadrados residuales al incluir parámetros adicionales en el modelo. Aquellos modelos que tengan valores pequeños de AIC o BIC se

pueden considerar buenos modelos.

Una forma de evaluar un criterio de selección es la **consistencia**. En que un criterio de selección de modelo es consistente si selecciona el verdadero modelo que tenga una mayor probabilidad de acercarse a la unidad si la muestra crece considerablemente. AIC se diferencia de BIC en que este último es más estricto con la adición de parámetros extras.

Además de la consistencia, otro factor de interés que incide en la selección del modelo es que sea **asintóticamente eficiente**, en que dado T el monto de datos disponibles se torna suficientemente largo para realizar cálculos asintóticos. Pudiendo en este punto analizar como la varianza del error de predicción se acerca a un punto de convergencia, el modelo que lo haga más rápido es mejor. En esto AIC es asintóticamente eficiente y BIC no.

La evaluación del impacto de un modelo de regresión y predicción permite comparar con otros modelos entrenados. De esta manera es posible determinar en referencia al resto de los modelos aquellos que presentan un mejor rendimiento, es decir la calidad relativa del modelo.

Los principales textos que definen estas métricas comparativas son:

Fitting Autoregressive Models for Prediction Se público en 1970 (Akaike, 1969), genera una formulación para un modelo, permite ajustar parámetros y seleccionar modelos, además de sistematizar la precisión de la predición.

Regression and time-series model selection in small samples (Hurvich & Tsai, 1989). Realiza un ajuste al modelo Akaike Criterio de Información, su uso se enfoca en simulaciones Monte Carlo y aplicaciones de serie-tiempo. La mejora consiste básicamente en que tiene un mejor comportamiento asintótico.

Stopped Training and Other Remedies for Overfitting Con el fin de prevenir sobreajustes en el texto W.S. Sarle (Sarle, 1995) desarrolla diferentes técnicas enfocadas en redes neuronales, evitando que llegue a aprender del ruido más que de los patrones del fenómeno de estudio. Se vale de regulación del entrenamiento, validación cruzada, técnicas de regularización, y remoción de parámetros de poca influencia.

Estimating the Dimension of a Model El desarrollo de la métrica BIC por G. Schwarz (Schwarz, 1978) permite confrontar la selección de modelos mediante el criterio de información bayesiano, también es compatible con AIC pero, a diferencia de este, penaliza la complejidad del modelo, enfocándose en la robustez frente al sobreentrenamiento, sobretodo con el incremento de la muestra

Entonces, dada la verosimilitud L se tienen las siguientes fórmulas para estas métricas. Las mismas métricas en término de L.

Considerando p la cantidad de parámetros del modelo, n la cantidad de muestras de la serie (extensión).

Criterio de Información Akaike (AIC):

$$\mathrm{AIC} = -2\ln(L) + 2p \tag{2.98}$$

Criterio de Información Akaike corregido (AICc):

$$\mathrm{AICc} = \mathrm{AIC} + \frac{2p(p+1)}{n-p-1} \tag{2.99}$$

Criterio de Información Bayesiana (BIC):

$$\mathrm{BIC} = -2\ln(L) + k\ln(n) \tag{2.100}$$

Una vez que se entrene cada modelo, considerando su arquitectura, parámetros y datos de entrenamiento que se reflejan en el error, podría ser MSE o DTW. Es posible realizar una estimación de estas métricas para comparar modelos.

2.18. Estado del arte: GNSS y Máquinas de Aprendizaje

Esta sección hace una inspección acerca de los trabajos aplicados de *machine learning* sobre el área de geodesía que son de interés destacar.

En Thomas y cols. (2023), se aplica un filtro de ruido a señales GNSS usando un modelo de deep learning sobre tramos estáticos de serie-tiempo, mostrando que efectivamente es una herramienta que aporta mejorar las mediciones, permitiendo una detección sísmica de menor intensidad al aumentar la sensibilidad con la extracción del ruido.

Además de la sismicidad clásica, en este tipo de señales se pueden observar fenómenos sísmicos a largo plazo, llamados 'terremotos lentos' (Ruiz y cols., 2017). Lo que es de interés para la institución CSN poder generar herramientas que también permitan observar la posibilidad de ocurrencia de estos.

Una etapa de vital importancia en este trabajo es la preparación adecuada de los datos con que se entrenará cada modelo, de tal manera que sea posible la gestión de los recursos y la entrega de la información suficiente para cada muestra al modelo. Esto significa realizar un análisis estructural del sistema, tanto a nivel particular, en su vecindad y en general, permitiendo construir una estructura de información lo más enriquecida posible.

De manera análoga a las señales GNSS, las señales por electroencefalograma han sido estudiadas como streams en que se les han aplicado técnicas de Deep Learning, como muestra Jaworski et al. Muestra diferentes metodologías en la implementación de estos modelos que podrían ser referentes para este trabajo.

Considerando el estado de arte modelos de machine learning que procesan secuencias de datos será necesario aplicar y estudiar los modelos adecuados para el conjunto de datos recibidos en stream en tiempo real.

Un modelo a destacar es RNN y lo más actual son modelos con los llamados Transformers (Vaswani y cols., 2023) en su artículo *Attention Is All You Need*, proponen un enfoque novedoso en el campo del procesamiento de señales con aplicaciones en machine learning ya que se caracterizan porque técnicamente permiten una alta paralelización del proceso de entrenamiento y predicción.

Así mismo se refuerzan las herramientas disponibles para el análisis de señales de datos en *streams* y la reducción de ruido, tomando de aplicaciones de *denoising* o cancelación de ruido en tiempo real, como lo describen (Porr B, 2022).

2.19. Procesos Gausianos aplicados a Redes Neuronales

Dado que existe una equivalencia entre una arquitectura de aprendizaje profundo con un proceso gaussiano, Neal, 1996 (Neal, 2012) presenta el marco de trabajo Bayesiano para el aprendizaje conectando el prior de una red neuronal con el prior como conocimiento formal. Esta relación permite utilizar inferencia estadística para comprensión del problema en estudio.

Dada una secuencia S_n para n valores, la probabilidad de obtener un valor x_{n+1} se da por la expresión bayesiana.

$$P(x_{n+1}|S_n) = \frac{P(S_n|x_{n+1})P(X_{n+1})}{P(S_n)} \tag{2.101}$$

La predicción de la cantidad x_{n+1} va dada por la inferencia integrando la distribución posterior.

$$P(x_{n+1}|S_n) = \int P(x_{n+1}|\theta)P(\theta|S_n)\partial\theta \tag{2.102}$$

Para conocer que tan exacto o tan *cierta* es la predicción se utiliza una función de pérdida.

La elección de un modelo simple puede repercutir en un alto *bias*, en cambio un modelo más complejo podría ajustar mejor el error.

En términos estadísticos las redes neuronales se pueden considerar como modelos *no paramétricos*.

2.19.1. Modelado de una red multicapa simple

Los resultados de una red MLP se pueden expresar por

$$f_k(x) = b_k + \sum_j v_{jk} h_j(x) \tag{2.103}$$

En que:

$$h_j(x) = tanh(a_j + \sum_i u_{ij} x_i) \tag{2.104}$$

Las funciones 'h' representan los nodos de la capa oculta. En que, además, a_k y b_k son el Bias para las unidades de ocultas y de salida respectivamente.

Un modelo de regresión con valores reales, la distribución condicional para los objetivos, y_k, dada la entrada,x, puede ser definida por una distribución gaussiana, con media $f_k(x)$ y desviación estándar σ_k, se tiene entonces.

$$P(y|x) = \pi_k \frac{1}{\sqrt{2\pi}\sigma_k} exp(-(f_k(x) - y_k)^2/2\sigma_k^2) \tag{2.105}$$

Los pesos y biases en la red están basados en un conjunto de casos de entrenamiento. Se minimiza la medida del error, siendo equivalente a conseguir la máxima verosimilitud para el modelo de ruido gausiano, dado que el log negativo de la verosimilitud de este modelo es proporcional a la suma de los errores cuadrados. Todo el proceso de entrenamiento considera

la *backpropagación*.

Desde la aproximación Bayesiana, la densidad posterior es proporcional al producto de cualquier prior utilizado.

$$L(\theta|(x,y)^{(1)},\ldots,(x,y)^{(n)}) = \pi_{i=1}^{n} P(y^{(i)}|x^{(i)},\theta) \tag{2.106}$$

Dado un nuevo valor x^{n+1} la predicción se puede calcular mediante la integración de la distribución de probabilidad prior.

$$\hat{y}_k^{(n+1)} = \int f_k(x^{(n+1)},\theta) P(\theta|(x,y)^{(1)},\ldots,(x,y)^{(n)}) \partial\theta \tag{2.107}$$

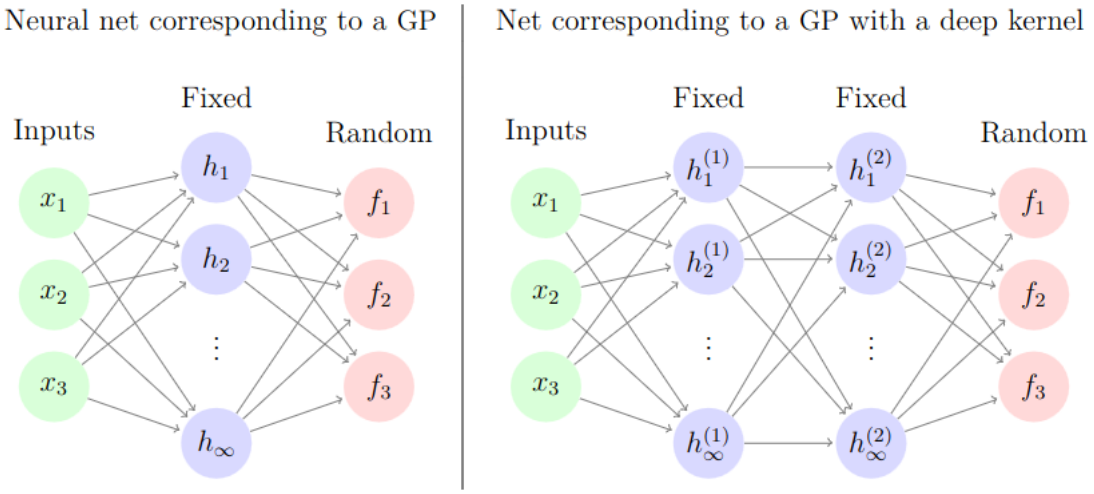

Figura 2.19: A la izquierda GP derivada en una capa oculta, a la derecha multiples capas generan un GP con un kernel profundo, no un deep GP(Duvenaud y cols., 2016)

La gran familia de redes neuronales se entiende como un conjunto de modelos heurísticos que definen una distribución de probabilidad a prior, que establece el sistema como un modelo de proceso gaussiano (figura 2.19). El objetivo de entrenar una red, definiendo los parámetros y bias, será encontrar la distribución posterior.

La predicción de observaciones con ruido, en que la medición asume un ruido aditivo sobre la señal real, ϵ con varianza σ_n^2 , el prior de las observaciones ruidosas es.

$$cov(y_p, y_q) = k(x_p, x_q) + \sigma_n^2 \delta_{pq} \tag{2.108}$$

En que δ_{pq} es delta de Kronecker, definida por 1 si $p = q$, 0 si no.

La predicción, entonces, se podrá expresar como la combinación lineal de una serie de kernels asociados con un parámetro α constante.

$$\overline{f}(x_*) = \sum_{i=1}^{n} \alpha_i k(x_i, x_*) \tag{2.109}$$

En que $\alpha = (K + \sigma_n^2 I)^{-1} y$. Siendo esta una representación de un proceso gaussiano de una red.

En Lam y cols. (2019) se observa que en una red LSTM aplicada a lenguaje natural, en que se modela un proceso gaussiano genera un efecto positivo en el sentido de mejorar el valor de las métricas de perplejidad.

Capítulo 3

Análisis estático y diseño de modelos

3.1. Exploración de los datos

Este capítulo considera el tratamiento de los datos en dos sentidos o planos de complejidad.

Uno individual a cada fuente, en que el dato **crudo** es tal cual lo entrega, que contiene varios campos de información se hace una selección específica de los campos referentes a la posición en el sistema de coordenadas **ECEF**.

El segundo consiste en un tratamiento conjunto de datos generados por grupos de fuentes cercanas. En que es de importancia determinar una correcta asociación bajo criterios de patrones similares o estaciones que tengan una alta correlación entre sí.

3.1.0.1. Fuente generadora de datos

El equipo que genera los datos **GNSS** es, en este caso, de la marca **Trimble**, utilizando la técnica de cálculo geodésico **RTX**, que permite un dato de alta precisión comparado con otras tecnologías de medición de la posición en tiempo real.

La frecuencia de generación de datos es cada un segundo (1 hz). Es decir, cada segundo en cada estación hay un nuevo dato. Además hay un ajuste de sincronización periódico cada 6 segundos aproximadamente, que permite mantener la precisión.

El dato se comunica en **tiempo real** mediante un protocolo de comunicación privativo llamado **General Serial Output Format (GSOF)**.

El equipo dispone de una interfaz gráfica configurable que permite seleccionar diferentes [8] tablas disponibles, además de la información base y la frecuencia de generaciónd el dato.

3.1.0.2. Dato original de fuente

El dato que se recibe en binario, a través de gsof, y se convierte a una estructura de datos de tipo **diccionario** como la que se puede observar en figura 3.1.

En el dato es posible encontrar diversos campos, en la tabla 3.1 se puede encontrar la definición de cada uno.

[8] Tablas General Serial Output Format (GSOF) https://receiverhelp.trimble.com/alloy-gnss/en-us/GSOF messages_Overview.html

```
{
  "DELTA_TIME": 0.935091,
  "DT_GEN": "2023-09-10T07:59:59.019000+00:00",
  "DT_RECV": "2023-09-10T07:59:59.954000+00:00",
  "ECEF": {
      "TIMESTAMP": 1694332799.138791,
      "X_POS": 1718919.5307764746,
      "Y_POS": -5186198.452159204,
      "Z_POS": -3279845.9676185185
  },
  "HEADER": {
      "CHECKED": true,
      "LENGTH": 114,
      "LOW_BATTERY": false,
      "MAX_PAGE_INDEX": 0,
      "PAGE_INDEX": 0,
      "STATUS": 168,
      "STX": 2,
      "TYPE": 64,
      "T_NUM": 126
  },
  "LATLONHEIGHT": {
      "HEIGHT": 46.68877063666977,
      "LATITUDE": -0.5436119956490042,
      "LONGITUDE": -1.2507497113923698,
      "TIMESTAMP": 1694332799.138791
  },
  "POSITION_VCV": {
      "NUM_EPOCHS_VCV": 0,
      "POSITION_RMS_VCV": 8.512575149536133,
      "TIMESTAMP": 1694332799.138791,
      "UNIT_VAR_VCV": 1,
      "VCV_XX": 0.0001965355040738359,
      "VCV_XY": -0.00012476563279051334,
      "VCV_XZ": 4.032628112327075e-06,
      "VCV_YY": 0.0007076954934746027,
      "VCV_YZ": 0.0004379991441965103,
      "VCV_ZZ": 0.0005822520470246673
  },
  "TIME": {
      "FLAG_1": 191,
      "FLAG_2": 35,
      "GPS_TIME": 28817000,
      "GPS_WEEK": 2279,
      "INIT_NUM": 2,
      "SVN_NUM": 13,
      "TIMESTAMP": 1694332799.138791
  },
  "TRACE": [
      "IPT_0",
      "VHIRSW5A",
      "A58U"
  ],
  "UTC": {
      "CT_FLAGS": 3,
      "GPS_MS_OF_WEEK": 28817019,
      "GPS_WEEK": 2279,
      "TIMESTAMP": 1694332799.138791,
      "UTC_OFFSET": 18
  },
  "id": "5e51c3c7-933e-43c3-9da0-254a9bccea3c"
}
```


Figura 3.1: Muestra de dato original convertido a diccionario de una estación GNSS Rtx

Tabla 3.1: Descripción de los campos de datos y su relevancia para la transformación a **ENU**. Los campos relevantes están marcados.

Campo	**Descripción**	**Relevante**
DELTA_TIME	Diferencia de tiempo en segundos desde que se genera hasta que se recibe en el servidor.	
DT_GEN	La marca temporal en tiempo GPS.	✓
DT_RECV	La marca temporal en recepción.	
ECEF	Valores principales de posición en un momento específico.	✓
HEADER	Información para control y chequeo de mensaje.	
LATLONHEIGHT	Información de posicionamiento en LAT, LON y altura.	
POSITION_VCV	Error de varianza correlacional entre los ejes ECEF.	✓
TIME	Valor de tiempo GPS generado.	✓
TRACE	Información de equipo fuente.	
UTC	Tiempo expresado en UTC.	

3.1.0.3. Estructura del dato a trabajar

El dato o unidad de información que se genera en sistema de coordenadas **ENU**, es obtiene a partir de lo siguiente:

- Valor de posición referencial de cada estación.
- Valor de dato original generado en ese momento.

Con esto se obtiene un dato que contiene solo lo necesario para este este estudio y otros relacionados a sismología, geodesia y otras ciencias.


```
{
    "source":"DataWork",
    "station":"INCA",
    "timestamp":1691294865000,
    "dt_gen":"2023-08-06 04:07:45+00:00",
    "data":{
        "N":{"value":0.1710646671312405,"error":0.011626335206857222},
        "E":{"value":-0.025732998963126405,"error":0.011626335206857222},
        "U":{"value":-0.25041155964472067,"error":0.011626335206857222}},
    "time":{"recv":"2023-08-06 04:07:45.688000+00:00","delta":0.688038}}
```


Figura 3.2: Muestra de dato en proyección ENU

El significado de cada campo es:

source Origen del dato calculado, el software que realiza el cálculo.

station Código de estación generadora

timestamp Marca de tiempo UNIX

dt_gen Marca de tiempo expresado

data Diccionario con los valores de la diferencia de desplazamiento entre la referencia y el valor recibido.

time Diccionario con información de tiempo de recepción.

La unidad de medida es en **metro**. Los campos **value** indican la diferencia respecto a la posición de referencia registrada (desplazamiento).

3.1.0.4. Metadata de las estaciones

Cada una de las estaciones contiene información de nombre, código, referencia y otra relativa a la administración de la red.

Se ubican geográficamente a lo largo del país, por lo que es posible visualizarlas en el mapa de la figura 3.3.

Como se observa, hay estaciones ubicadas a lo largo del borde costero, que son aquellas que detectan primero una onda sísmica que proviene desde la juntura de placas.

Las posiciones descritas en las tablas (3.2)son valores ofuscados para evitar dar la ubicación exacta, por lo que se reduce la precisión.

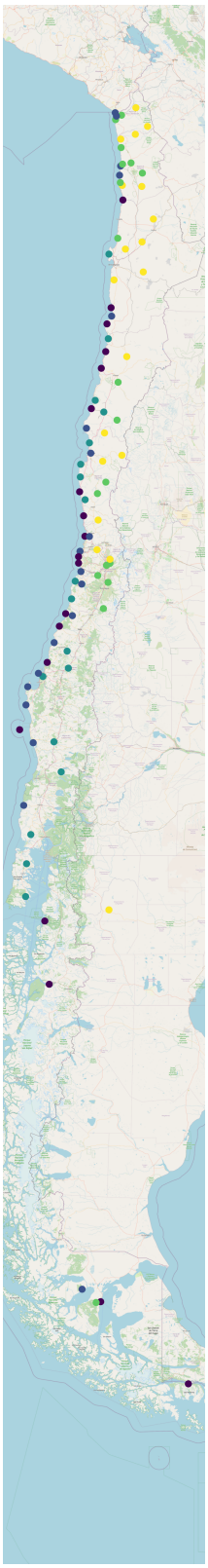

Figura 3.3: Estaciones GNSS instaladas en Chile

3.1.1. Estudio de data Serie-Tiempo

Cada estación provee de un *flujo* (stream) de datos cada segundo, salvo cuando ocurre alguna desconexión de la red.

Estos datos pueden visualizarse mediante un gráfico de serie-tiempo como en la figura 3.4.

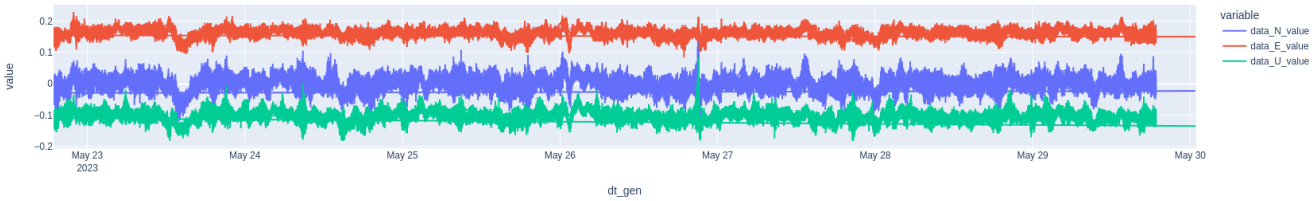

Figura 3.4: Serie tiempo para estación FMCO, 1 semana

El comportamiento de las curvas serie tiempo, en el común de los tiempos es similar al observado:

- Cada curva tiene una tendencia a mantener la diferencia respecto a la referencia. Es decir, mantiene una estacionalidad.
- Se observa ruido en torno al punto medio de la curva
- Se observan perturbaciones o cambios generales que no están asociadas a sismos.

En un acercamiento con mayor detalle se define un tramo de tiempo más corto en la figura 3.5. La visualización de los tres ejes permite observar que se manifiestan patrones de comportamiento a lo largo del tiempo, cómo que para U muestra una curva que parece ser un reflejo de N, lo que nos induce a establecer que podrían haber ciertas correlaciones entre las series de los ejes de la misma estación.

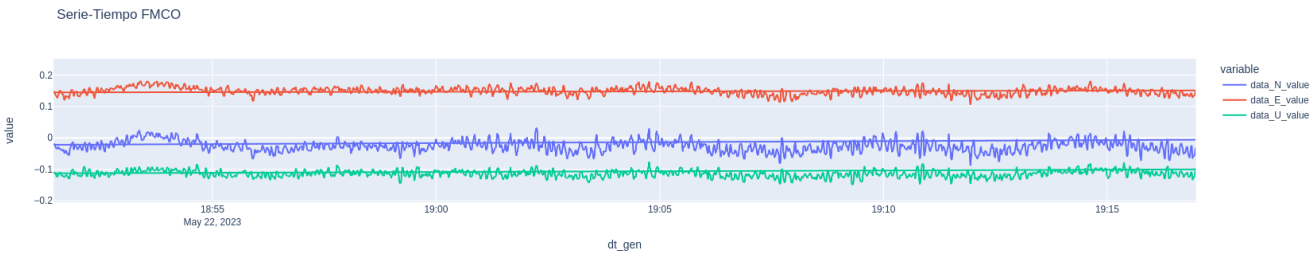

Figura 3.5: Gráfica de serie tiempo FMCO, tiempo corto

Así como también el error en asociado a la señal para cada muestra en figura 3.6:

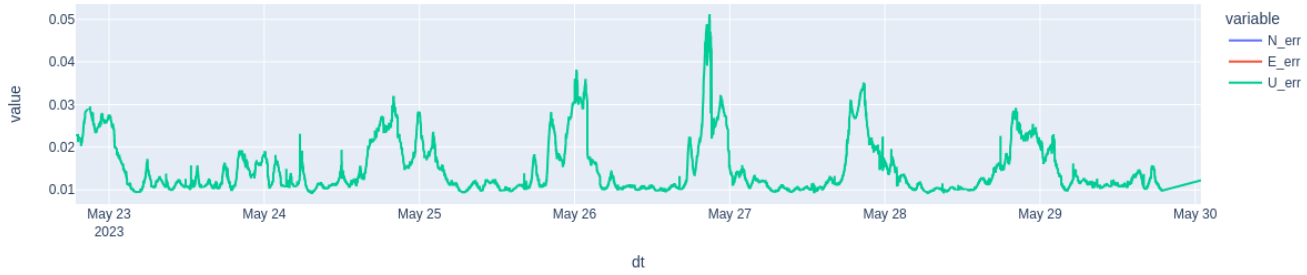

Figura 3.6: Error asociado a la secuencia serie-tiempo (FMCO), se observa un periodicidad del error diariamente de 0.1 a 0.3

Además de estos patrones, también se observa un ruido que perturba la señal constantemente.

También se puede observar, mediante un gráfico de caja en la figura 3.8, que mantiene una estacionalidad en torno al valor 0.0, es decir se mantiene estable en torno al punto de referencia. La variabilidad se podría explicar por algún fenómeno periódico como efecto de las mareas o la periodicidad de los satélites que envían la señal.

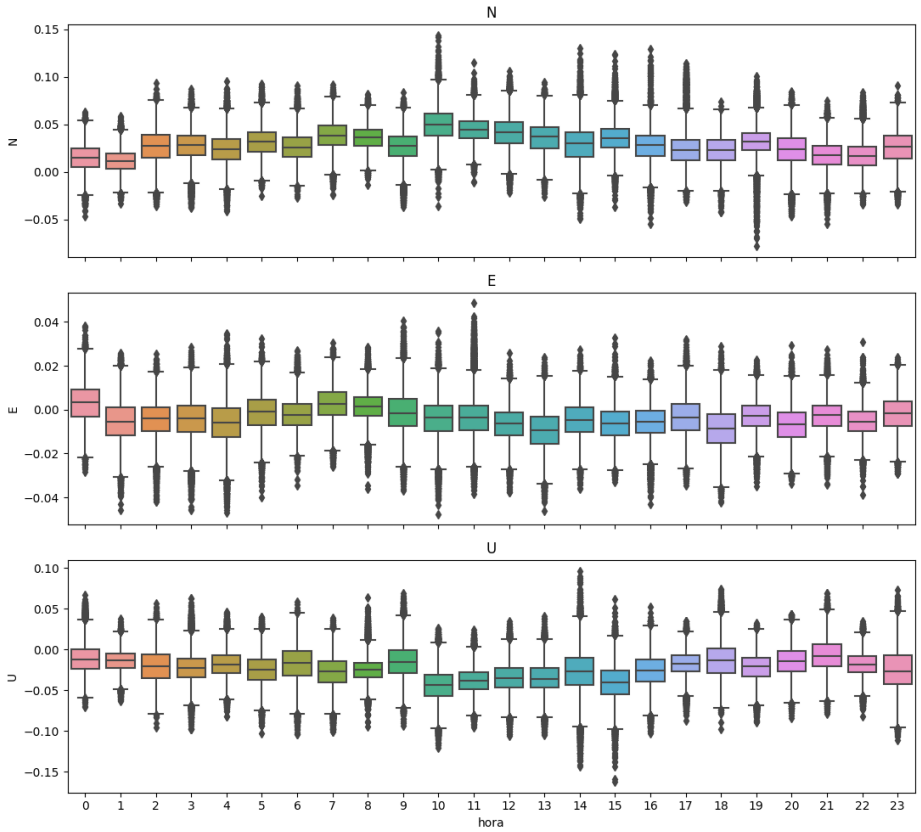

Figura 3.7: Gráfica de cajas de serie tiempo UTAR, tiempo acotado

Respecto a que tanto depende el valor actual de los anteriores, comprendiendo que cada parámetro es una variable aleatoria definida dependiente de sus valores anteriores. Se tiene, para un tiempo **t**, por cada serie.

$$P(x_t) = P(x_t|x_{t-1}, x_{t-2}, ..., x_k) \tag{3.1}$$

Para encontrar un **k** adecuado, dado que no se dispone de la información inicial real, sino desde el momento en que se realiza la medición, es posible hacer un estudio de la **autocorrelación**, que determina una curva descendiente, en que para tener un valor superior a **0.6**, que podría considerarse adecuado, serían necesarios 300 puntos o segundos de información en lo ejes N y U, pero en E basta con 40, que es el eje de mayor interés para la sismología en Chile.

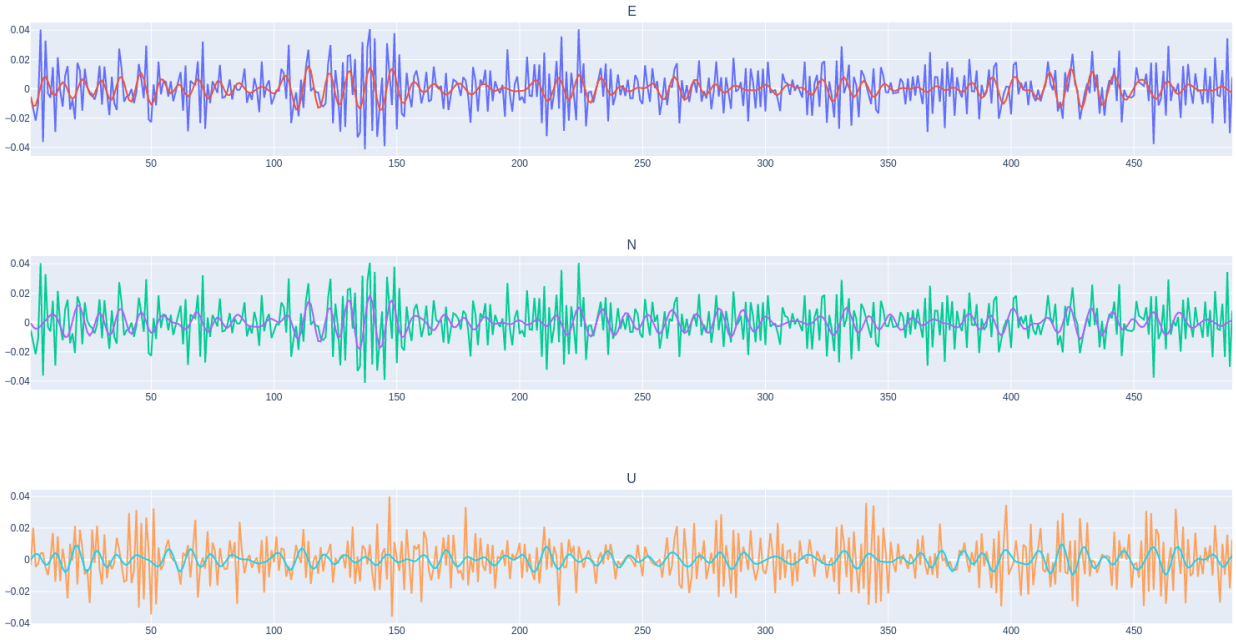

Figura 3.8: Gráfica de autocorrelación, para toda la serie FMCO

Este es un valor importante, ya que permite afinar el tamaño de **buffer** de cada serie para el presente estudio. Además, se observa que se tienen periodicidades cada unas seis posiciones.

3.2. Descomposición Serie-Tiempo

Para encontrar una mejor expresión de las secuencias serie-tiempo estudiadas, se hace un análisis que nos permitirá describir la composición de estas:

- tendencia - periodicidad - estacionalidad - ruido

Encontrando una descomposición adecuada se hace más sencillo generar un modelo predictor y que además sea lo más general posible.

Aplicado el operador **B** (backshifted) para ir descomponiendo de manera estándar la secuencia serie-tiempo en sus características principales.

$$w_t = \hat{y}_t = (1 - B)^2 y_t \tag{3.2}$$

Al extraer la tendencia las secuencias se ajustan a un valor central en 0 (figura 3.9), lo que es conveniente, ya que esto aporta a estandarizar las secuencias a nivel general y poder omitir la constante de ajuste de la posición de referencia.

Figura 3.9: Secuencia ST FMCO sin la tendencia, centrada en 0

La tendencia muestra claramente el desplazamiento de la referencia para cada eje que es diferente (figura 3.10). Este desplazamiento se debe a imprecisiones en el registro de la posición de referencia y que, con el tiempo, ocurren desplazamientos mínimos que requieren una actualización de este valor.

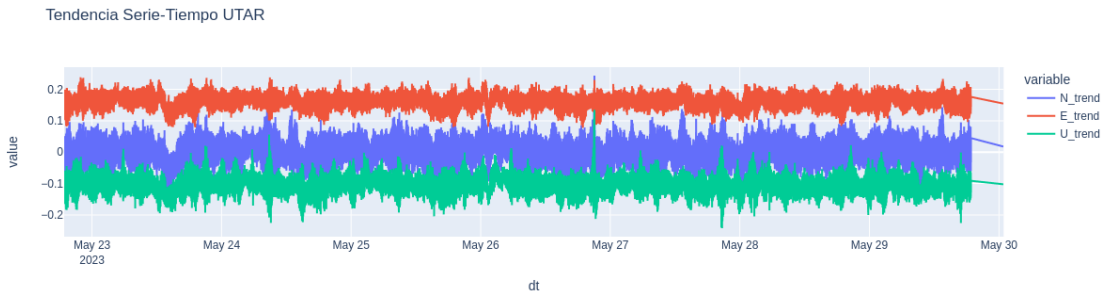

Figura 3.10: Secuencia de tendencia ST FMCO

Nuevamente, si se observa la serie-tiempo sin la *tendencia* será posible destacar la evidencia de una periodicidad (que también aparece en cada estación) y se puede asociar a un efecto instrumental para el ajuste del cálculo.

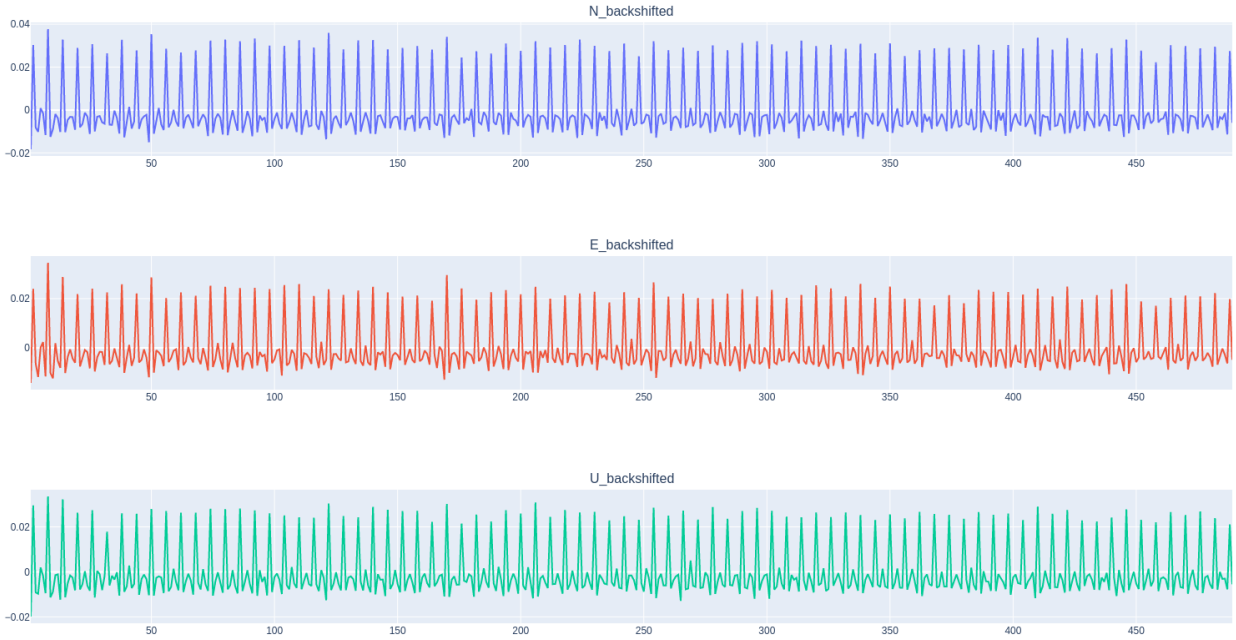

Figura 3.11: Autocorrelación serie-tiempo sin la tendencia

Con esto se postula la aplicación de una operación **backshifted** que permita extraer las periodicidades de la secuencia.

$$\hat{w}_t = (1 - B^6)w_t \tag{3.3}$$

Evidentemente, se genera una extracción del componente periódico de mayor importancia, generando una mayor independencia entre cada punto de la secuencia serie-tiempo (figura 3.12).

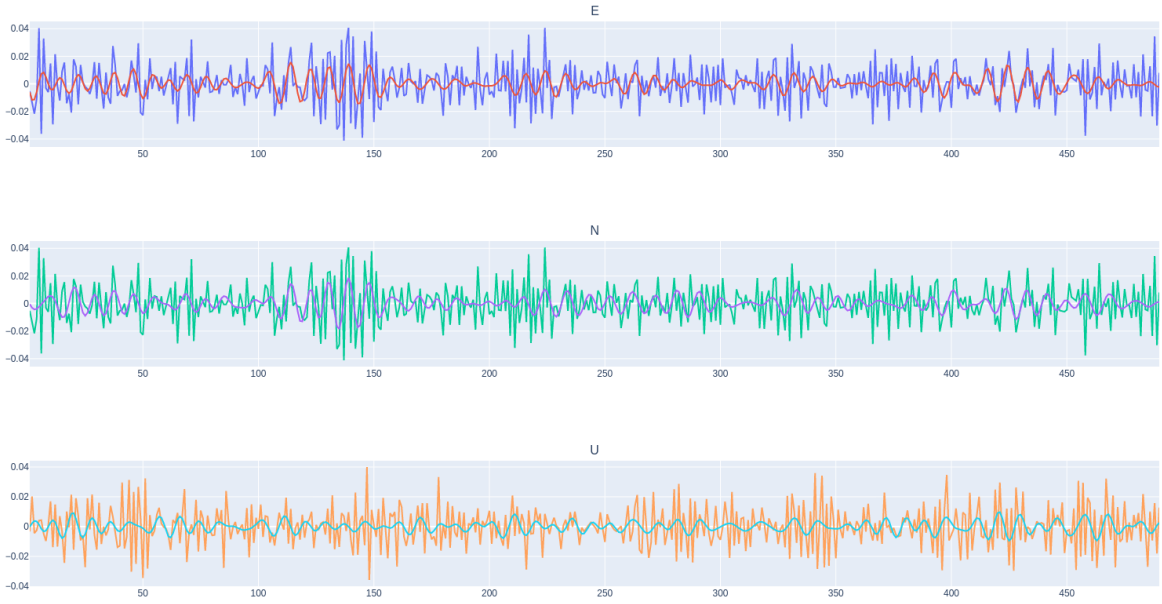

Figura 3.12: Autocorrelación serie-tiempo sin la tendencia y sin periodicidad de 6s

Con esto, corresponde también realizar un análisis espectral de la frecuencia, considerando que el stream de datos es un proceso estocástico y puede escribirse como la suma de senos y cosenos.

$$X_t = \eta + \sum_{n=1}^{N} [a_n cos(w_n t) + b_n sin(w_n t)] \tag{3.4}$$

En que:

1. η :: es la media de la serie.
2. $\omega_n = \frac{2\pi n}{T}$ :: frecuencias naturales, si T par, $N = T/2$, y si T es impoar $N = \frac{T-1}{2}$

Los coeficientes a_i y b_j son la amplitud asociada a cada función, y te un índice que va de 1 a N.

Estos coeficientes obedecen a la siguiente fórmula para cada parámetro a_n.

$$a_n = \frac{2}{T} \sum_{t=1}^{T} (X_y - \eta) cos(w_n t) \tag{3.5}$$

Y la siguiente para los parámetros b_n.

$$b_n = \frac{2}{T} \sum_{t=1}^{T} (X_y - \eta) cos(w_n t) \tag{3.6}$$

Considerando que la mayor frecuencia observable de la serie corresponde a medio ciclo por unidad de tiempo (frecuencia Nyquist).

En un análisis armónico, a_n y b_n son variables aleatorias con:

$$\begin{aligned} E(a_n) &= E(b_n) = 0 \\ E(a_m a_n) &= E(b_m b_n) = \sigma^2 (m = n) \\ E(a_m a_n) &= E(b_m b_n) = \sigma^2 (m! = n) \\ E(a_n, b_m) &= 0 \end{aligned} \tag{3.7}$$

Le generación de la secuencia filtrada con FFT permitirá obtener una serie-tiempo **target** que, además de suavizar cada secuencia, extrae aquellos factores de ruido que no son de interés para este estudio. Se puede observar en la figura 3.13 que mantiene la coherencia de la señal original pero sin exponerse a los efectos del ruido de alta frecuencia.

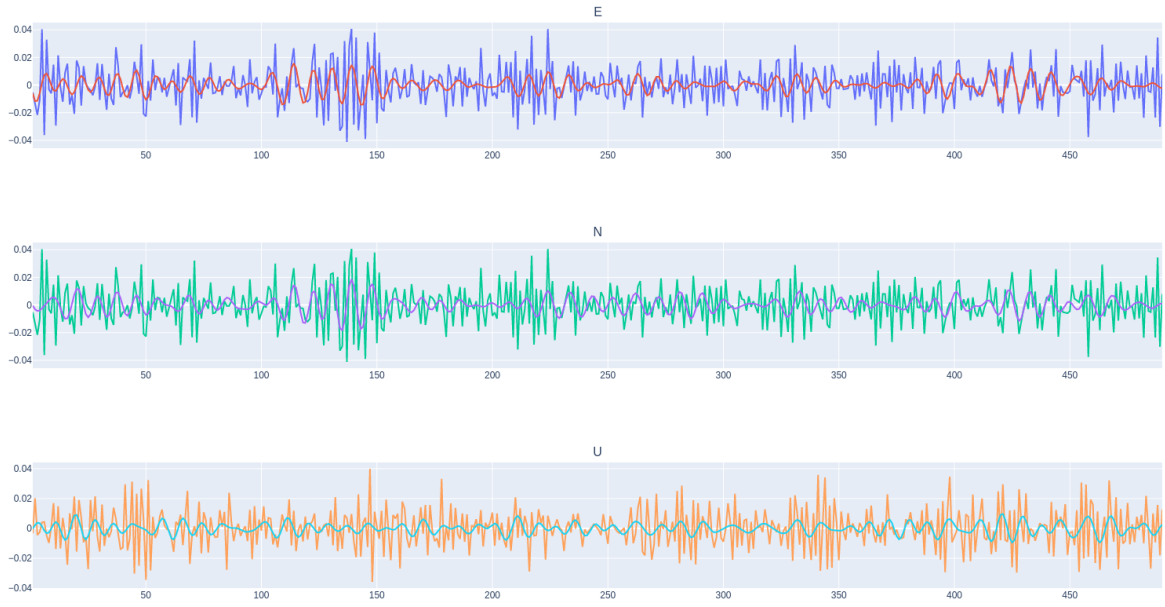

Figura 3.13: Secuencia Serie-Tiempo Backshifted + Sin-periodicidad, con secuencia FFT filtro de ruido

En resumen, cada secuencia serie-tiempo de cada estación se puede desglosar en sus componentes principales (tendencia, periodicidad, residuo).

$$y_t = T_t + P_t + R_t \tag{3.8}$$

Tabla 3.2: Localización (ofuscada) de estaciones GNSS

Código	Longitud	Latitud
PTRE	-69.5743	-18.1943
CHYT	-70.3421	-18.3708
PCCL	-70.1067	-18.4577
IACR	-70.3329	-18.4797
UTAR	-70.2965	-18.4906
CMRC	-70.3282	-18.6142
SURI	-69.1434	-18.8538
MNMI	-69.5956	-19.1314
ATJN	-70.1368	-19.3008
PSGA	-70.1230	-19.5974
FBAQ	-69.7554	-20.1346
CGTC	-70.0690	-20.1771
UAPE	-70.1413	-20.2430
PICC	-69.3346	-20.4898
GTA1	-70.1807	-20.5657
PATH	-70.1527	-20.8208
CRSC	-70.0798	-20.9177
CLLA	-69.3566	-20.9545
ELOA	-70.0568	-21.4249
RADO	-68.9268	-22.0829
MCLA	-70.2475	-22.7458
SRGD	-69.3478	-22.8711
VLZL	-69.9647	-23.1172
JRGN	-70.5749	-23.2889
PB19	-69.3023	-23.8967
CRIS	-70.3868	-24.1679
PAPO	-70.4953	-25.1083
TLT2	-70.4751	-25.3913
CIFU	-70.6470	-25.6526
PAZU	-70.5987	-26.1479
FMCO	-70.6943	-26.5790
INCA	-69.9128	-26.7459
BING	-70.8575	-27.1326
TAMR	-70.2346	-27.5938
LLCH	-71.0815	-28.1903
HSCO	-71.2266	-28.4615
VALL	-70.7644	-28.5720
TRST	-70.2738	-28.8363
CRZL	-71.4097	-29.1016
SILL	-70.7387	-29.2549
LHOR	-71.2975	-29.5759
LSCH	-71.2460	-29.9082
JUNT	-70.0936	-29.9766
TOLO	-70.8061	-30.1699
PVCA	-71.6232	-30.2675
PFRJ	-71.6354	-30.6747
PEDR	-70.6891	-30.8390
EMAT	-71.6627	-31.1467
CMB2	-71.0006	-31.2036
CNBA	-71.4580	-31.3982
LVIL	-71.5138	-31.9092
SLM1	-70.9745	-32.0434
ZAPA	-71.4656	-32.5528
CTPC	-71.2976	-32.5611
ROB1	-71.0158	-32.9758
TRPD	-71.6475	-33.0228
QTAY	-71.7022	-33.1927
SMAR	-70.5372	-33.2727
QSCO	-71.7019	-33.3919
CALN	-70.5372	-33.3980
DGF1	-70.6617	-33.4573
CCSN	-70.6640	-33.4632
RCSD	-71.6135	-33.6544
TLGT	-70.9888	-33.7763
NAVI	-71.8246	-33.9527
CHDA	-70.6090	-33.9890
RAPL	-71.5799	-34.0405
CRAP	-71.5888	-34.0416
PCMU	-71.9612	-34.4962
SBLL	-70.7807	-34.7921
ILOC	-72.1790	-34.9488
HLN2	-71.9302	-35.0099
CONS	-72.4122	-35.3310
PELL	-72.6056	-35.8279
QLAP	-72.1255	-36.0845
VITA	-72.8647	-36.4236
CLL1	-72.0800	-36.5951
HLPN	-73.1902	-36.7478
CONZ	-73.0255	-36.8438
PLVP	-73.5850	-37.1488
PECL	-73.6506	-37.6870
IMCH	-73.8853	-38.4120
TMCO	-72.6137	-38.7648
SAAV	-73.3834	-38.7916
PGLL	-72.3455	-39.6322
PUCA	-73.7350	-40.5864
MUER	-73.4713	-41.4052
LNCM	-73.6282	-42.2151
QLLN	-73.6642	-43.1142
UDAT	-70.5749	-43.4738
RMBA	-72.9533	-43.7751
CHAC	-72.7894	-45.4504
IRSC	-71.5698	-52.8574
PARC	-70.8799	-53.1370
PTAR	-71.0496	-53.1552
PWIL	-67.6304	-54.9322

Capítulo 4

Diseño e Implementación de los modelos

En este capítulo se describe el diseño base y la metodología llevada a cabo para entrenar los modelos, los hiperparámetros seleccionados, función de optimización y razonamientos realizados en el proceso.

En primero lugar, es necesario recalcar que se intenta encontrar un modelo que pueda ser aplicado de manera general a toda la red. Sin embargo, se realizará una aproximación progresiva a la generalización.

La estrategia a seguir será entonces realizar entrenamientos localizados en estaciones específicas e ir evaluando con esto la posible generalización del modelo.

Criterios a analizar:

1. Entrenamiento focalizado
2. Entrenamiento sobre un grupo aleatorio
3. Entrenamiento general

Los factores a considerar serán:

1. Tiempo que demora del entrenamiento
2. Tiempo que demora en entregar una predicción
3. Hiperparámetros y ajustes

Dado que se está tratando de realizar una regresión y predicción de serie-tiempo será de utilidad considerar las siguientes métricas:

1. Mínimos cuadrados (MSE)
2. Dynamic Time Warping
3. Akaike Information Criterion corrected (AICc)
4. Bayesian Information Criterion (BIC)

El análisis de cobertura consistirá en evaluar el impacto predictivo de cada modelo según la distancia. A partir de un nodo (una estación) y su modelo asociado se evaluará cómo se comporta el modelo con estaciones ubicadas a diferentes distancias. Así será posible definir un área de cobertura para el modelo en que sus resultados predictivos sean aceptables.

Con esto, se podrá definir un grupo específico de estaciones en que el modelo tenga un comportamiento predictivo similar, haciendo posible entrenar un modelo ya no específico a la estación sino un modelo de cobertura asociado a la zona que cubre el grupo de estaciones.

El procedimiento para entrenar modelo es:

- Selección de estación: por ejemplo UTAR.
- Selección de grupo de estaciones.
- Entrenamiento general.

Se definen los conjuntos de entrenamiento, previamiente separados cada dataset en 15 secciones (cada 12 horas).

- entrenamiento : de 0 a 11
- test : 11 y 12
- validación : 12 y 14

Se considera además la inclusión de $dropout = 10\,\%$ y funciones de activación *tanh* y *relu* en la etapa previa a la salida.

Las funciones de normalización para las características serán las siguientes.

Normalización del tiempo, segundos del día:

$$B = A/24 * 60 * 60 \tag{4.1}$$

Normalización de los valores en la secuencia serie-tiempo.

$$\begin{aligned} top &= 0.15 \\ B &= A/top \end{aligned} \tag{4.2}$$

Normalización de la distancia en la tierra. Considerando 100 mil metros como un radio medio de referencia.

$$\begin{aligned} R &= 100\_000 \\ B &= A/R \end{aligned} \tag{4.3}$$

Normalización de la posición (lat, lon, altura), expresandas en (deg, deg, m).

$$\begin{aligned} LAT &= LAT/180 \\ LON &= LON/180 \\ H &= H/9000 \end{aligned} \tag{4.4}$$

Los hiperparámetros de entrenamiento inicial son, ajustables para cada experimento, en tabla 4.1.

Tabla 4.1: Hiperparámetros y configuraciones del modelo

Parámetro	**Valor**
Learning rate (λ)	0.001
Momentum (μ)	2.2
Batch size train	1024/256
Batch size test	512/128
Epochs	1
Window	120
Future	60

4.1. Componentes de modelo

En esta sección se definen e implementan modelos ya estudiados en la literatura y se desarrolla una propuesta de componentes para cada modelo en que se trabajará en esta tesis.

Es importante destacar la estrategia de ordenamiento del dataset que permita la detección de los diversos patrones de interés mediante la definición de una estructura de datos adecuada e interpretable.

4.1.1. Repositorio de software implementado

El software implementado para este trabajo es accesible libremente mediante la siguiente dirección al repositorio de la plataforma **gitlab**.

Software GNSS DL https://gitlab.com/pineiden/gnss-ml-v2.git

Considera una estructura ordenada de directorios en que se definen clases, funciones y *scripts* de operación para los distintos experimentos realizados.

- cobertura
- dataload
- datasets
- dataslice
- evaluate
- funciones
- managers
- models

- notebooks
- scripts
- tests
- train

Los requisitos básicos para operar son:

1. Python 3.10
2. Tarjeta Gráfica NVidia (GPU)

Para realizar la instalación y hacer pruebas será necesario leer **README.md** que es la documentación básica para utilizar el software.

Contiene las clases que definen los modelos, iteradores, dataloaders y funciones para entrenar y evaluar la evolución del modelo.

En el **anexo I** es posible encontrar la definición de estos modelos y la arquitectura desarrollada.

4.1.2. Modelo de Regresión y predicción

Se definen distintos modelos para realizar la tarea de **Regresión**, es decir interpretar un tramo de la serie tiempo de cada eje y producir otra en que el ruido haya sido filtrado, además la tarea de **predicción** (forecasting) que estima el comportamiento de la serie tiempo en un futuro cercano, dado los valores de la serie entregada.

Dada una serie-tiempo de datos de una estación, en conjunto con sus vecinos, es posible aplicar un filtro de ruido que permita mejorar la detección de comportamientos que podrían ser no detectables a simple vista.

Consiste en la aplicación de diversas etapas que extraigan las características principales de la señal, como por ejemplo la convolución sobre el conjunto de estaciones cercanas, luego aplicación de una red neuronal que permita entregar la señal decodificada con reducción del ruido.

4.1.3. Modelo de grafos por vecindad.

Se tiene una red de sensores que se puede representar como un modelo de grafos, cada sensor es un nodo y los ejes serán definidos por su vecindad.

La ubicación geográfica de cada estación posee coordenadas específicas, lo que permite evaluar la *distancia* entre cada una de ellas.

Se observa en los datos una correlación entre estaciones ubicadas en proximidades, por lo que se define que únicamente las primeras M estaciones más cercanas serán consideradas para la formación de un conjunto de series temporales con propósito de entrenamiento.

Se establece un *modelo de grafos* sobre la red y la definición de una distancia, en este caso se utilizará una *distancia euclidiana* que permite tener una medida simple y con esto un ranking de cercanía.

El objetivo de definir estos conjuntos de cercanía a cada estación es extraer tramos equivalentes de tiempo del conjunto y aplicar una extracción de características como la convolución sobre todas las estaciones y sus ejes (N,E,U).

Así será posible definir un conjunto de sensores vecinos a uno central. Se define así el **sensor-foco** ya que la medición que cubren en conjunto corresponderá a un área focal, cuya medición general estará referida al sensor central.

Se establecen tres diferentes conjuntos por vecindad. Este procedimiento se realiza en el *notebook* **distance_matrix_grafos**. Tomando la geolocalización de cada estación, se establece una clasificación en base a la distancia,

first primeras cuatro estaciones

second segundas cuatro estaciones

random selección aleatoria entres las primeras diez

Esta estrategia permite definir una matriz de datos de **MxT** en que **T** es la cantidad de segundos seleccionados por tramo. Además de que en cada conjunto seleccionado hay un desplazamiento de **K** segundos para cada nuevo dataset, para dar una idea de esta estructura revisar figura 4.1. Permitiendo de esta manera crear un conjunto de datasets que podría considerarse también como **data augmentation**, pero llamaremos **N-composición** a esta modalidad.

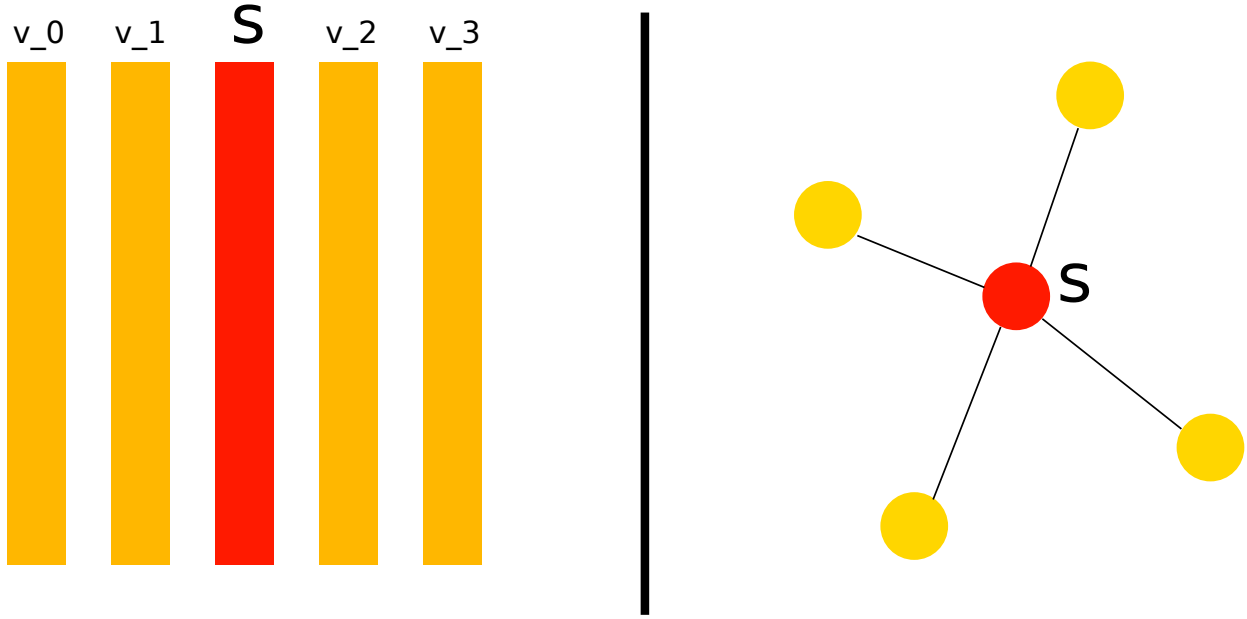

Figura 4.1: Composición de secuencias serie-tiempo sensor-foco, representación matricial a la izquierda, representación de grafo a la derecha

Además como **data augmentation** se debe considerar la estrategia de tomar segundos niveles de distancia, para modelar casos en que no se tengan datos de estaciones, se agreguen estaciones, o evaluar refinamientos.

Con esto se define la salida como el conjunto de datos específico de la estación a estudiar aplicando un **filtro** de ruido por transformada de Fourier (notebook *analisis_fourier*) [9]

Un criterio de diseño a considerar es que el tamaño de ventana del kernel de convolución debe ser una fracción más pequeño que la ventana aplicada a la serie de datos $\mathbf{Y}$. De esta manera el ajuste de sensibilidad sobre los parámetros de entrenamientos permitirá realizar una tarea de predicción adecuada.

Preparados los datos de esta manera será posible entrenar un conjunto de máquinas que tengan dos características de interés:

- Filtro de señal
- Predicción al futuro

Se evaluarán las capacidades de los modelos dado el dataset disponible.

4.1.4. Algoritmo para la gestión de memoria.

La magnitud del dataset de entrenamiento para los modelos es significativa, llegando a los 10GB para contener datos de una semana para el conjunto completo de estaciones.

En este sentido las primeras pruebas mostraron la necesidad de crear un método que permitiera cargar los datos de manera regulada a medida que se necesiten.

Esto se hace posible ya que se parte con el modelo de grafos por vecindad y se realiza una lectura de los archivos de cada estación en orden norte a sur.

El algoritmo implementado permite ir moviendo un foco de uso del conjunto de estaciones cuyos datasets deben permanecer en memoria, liberandola a medida que deje de necesitarse una estación en particular.

Para eso se tiene un mapa diccionario de *modos de agrupación* asociado a un mapa diccionario de *grupos de estaciones vecinas*.

Require: Mapping $G[k, g_k]$
Require: Mapping counter $C[k, l_k]$

for *code, group* $\leftarrow$*G.items()* **do**
    `dataset` $\leftarrow$ `read(`*group*`)`;
    yield: `batched(`*dataset*`)`
    `discount(`*C, group*`)`;
end

Algoritmo 1: Algoritmo de carga de datos `algoritmo_pocoapoco.py`

Con esto en cuenta, se realiza un mapa diccionario de la *cantidad de veces que se ocupa cada estación*.

Una vez procesando los datos y entrenando los modelos, se va descartando del contador cada estación (código) correspondiente. Así, el dataset cargado a memoria en un mapa diccionario *buffer* teniendo cargado estos datos, se descartan cuando el contador llega a cero (0).

[9] Implementación de filtro FFT para extracción de ruido https://www.kaggle.com/code/theoviel/fast-fourier-transform-denoising

Esta solución resulta ser efectiva para el objetivo de entrenar máquinas de aprendizaje, ya que se integra con facilidad en el flujo de trabajo de este proyecto.

4.1.5. Etapa de convolución sobre Serie-Tiempo

En conjunto, la estación y sus vecinas permiten definir mediante convolución con un **kernel** una serie de características serializadas que pueden entrar a una **red neuronal**.

El dataset se debe preparar de tal manera que en la centralidad de cada conjunto este la estación a estudiar y en las posiciones laterales las serie tiempo de las estaciones vecinas.

La aplicación del **kernel** opera de manera similar a aplicarlo sobre una imagen, en este caso se tendrán la cantidad de canales de profundidad dependiendo de la **N-composición** y **M** columnas, con la estación de estudio en el centro. En el caso de este trabajo, solo se aplicará sobre (N=1) capa y series tiempo con valores de distancia.

Además, teniendo en consideración que contamos con tres ejes (N-E-U), se deberá realizar el mismo procedimiento de entrenar un modelo para cada uno de estos.

Con esto, al aplicarse el **kernel** se tendrá un vector de una columna por cada eje (N-E-U).

En el estudio de los modelos se definirá la cantidad de capas en convolución que serán aplicadas.

Se debe considerar además un factor de **dropout** que apague de manera aleatoria algunos parámetros, de manera tal que simule momentos en que no se han obtenido datos (por desconexión u otra causa).

4.1.6. Dataloaders

La creación de la clase Dataloader permitirá gestionar la lectura de los archivos con datos de cada estación de manera adecuada y acotada a lo que se necesitará en el entrenamiento del modelo.

Debe considerar que la lectura debe ser eficiente para evitar colapsos en el uso de la memoria computacional, por lo que cada archivo leído debe permanecer solamente para los conjuntos en que usará.

Esta clase debe definir una **ventana** de tiempo sobre la cúal se hace la observación en el modelo, es decir un tramo de tiempo suficientemente grande para extraer las características de mayor frecuencia pero suficientemente pequeño para ser gestionable en el modelo y la capacidad computacional.

Además, debe ser capaz de gestionar el modelo de grafos por vecindad. En que primero se construyen los conjuntos de vecidades bajo diferentes categorías y el acceso a los archivos se hace en base al criterio obtenido por estos conjuntos.

Para eso se define el algoritmo anteriormente descrito de lectura que pueda gestionar estos recursos, manteniendo en memoria la data de cada estación solamente mientras sea necesaria.

Esta clase contiene el arreglo de grupos de estaciones asociadas mediante la definición de grafo y sus vecinos. Haciendo referencia a los archivos ".data".

4.1.7. Etapa de red neuronal

Las redes neuronales a considerar para el entrenamiento de los datos **gnss** se consideran de la siguiente manera:

CNN + Fully Connected Lineal Test una red simple con un dataset recortado para probar que el algoritmo opere correctamente.

CNN + Fully Connected Lineal una red simple que no considerar la correlación temporal hacia el pasado, tomando cada valor como independiente. Se considerará este modelo como la *linea base*.

RNN/LSTM Es una red que aplica una mayor inteligencia a la dependencia con valores anteriores. Siendo una red de tipo recurrente, permite obtener resultados satisfactorios pero limitados por la recursividad.

Transformers Es un tipo de red moderno, en estado del arte, que permite establecer las relaciones de dependencia con los valores anteriores y además es capaz de utilizar recursos de GPU en paralelo.

4.2. Definición de etapa convolucional CNN.

Una convolución es una operación entre un bloque de entradas, en este caso el conjunto de señales de la estación y sus vecinas, con uno o más *kernels* que operan como filtros.

Esta operación, en conjunto con las siguientes etapas de la red neuronal, permiten realizar una operación de extracción de características a la entrada. En términos matemáticos, permite pasar desde una matriz $N * M$ a un vector Wx1, en que necesariamente $W >= N * M$,

Cada uno de los *kernel* consiste en una matriz cuyos valores se van ajustando con el entrenamiento del modelo, la convolución sobre cada punto (para no llamarlo pixel) de la matriz entregará un valor compuesto que caracteriza la localidad. En este caso se compone de valores en el tiempo de si misma y de las estaciones vecinas en el mismo tramo de tiempo.

La etapa de convolución produce un *mapa de activación*, y este puede ser el resultado de una o más etapas de convolución en cascada. Se espera que estas características tengan relación con la periodicidad, comportamientos locales y otros patrones que podrían existir. El resultado final de esta operación debe ser un vector de valores de entrada para la siguiente etapa.

4.2.1. Optimizadores

Se evalúan dos optimizadores conocidos como SGD y ADAM

1. Adam Ayuda a converger más rápido.
2. SGD Generaliza mejor la solución encontrada

Una comparativa de ambos optimizadores se observa en el artículo 'A guide to improving CNNs optimizer' [10]

[10] Comparativa Adam vs SGD https://medium.com/geekculture/a-2021-guide-to-improving-cnns-optimizers-adam-vs-sgd-495848ac6008

4.2.1.1. Criterio de pérdida

Para problemas de regresión en series temporales, donde el objetivo es predecir valores numéricos continuos en lugar de clases discretas, se dispone principalmente de MSE y DTW.

En su lugar, para problemas de regresión en series temporales, algunas opciones comunes de funciones de pérdida son:

Mean Squared Error (MSE) Es la forma de calcular la pérdida más común en problemas de regresión. Calcula el promedio de las diferencias al cuadrado entre las predicciones y los valores reales.

Classic Dynamic Time Warping Mejora el ajuste del modelo comparado con MSE al comparar la forma de la onda y entregar una métrica de similaridad entre las ondas comparadas.

4.3. Modelo Base

A manera general se propone un modelo esquemático, que considera la inclusión de los datos, que contiene un vector de tiempo (ordenado según en segundos del día), una matriz de datos del grupo asociado, distancias a estación, posición y valor representativo del código de estación (onehot).

La matriz de datos serie-tiempo se inserta en la capa CNN para extraer sus características y su salida se concatena con el vector de tiempo normalizado, posición, distancia y onehot. Esto entra en la etapa de red para ser procesada como una red *fully connected*.

La composición esperada de esta arquitectura permite procesar la matriz de serie-tiempo asociada bajo una convolución que termina produciendo un vector de características que puede ser acoplado junto al resto de la información de interés. Se puede considerar esta etapa como una etapa de preparación que está integrada a la arquitectura.

Con esta arquitectura se planifica la realicación de diferentes pruebas y ajustes de hiperparámetros con el fin de encontrar los mejores resultados con este modelo.

En la figura 4.2 se puede observar la arquitectura que el modelo base (en abstracto) contiene:

1. capa CNN (convolucional).
2. vector de tiempo: 120 s.
3. vector de posición de cada estación: 3x5.
4. vector de distancias de cada estación: 1x5.
5. one hot código estación: 8.
6. etapa MLP.
7. salida regresión 120s.
8. forecasting de 60s.

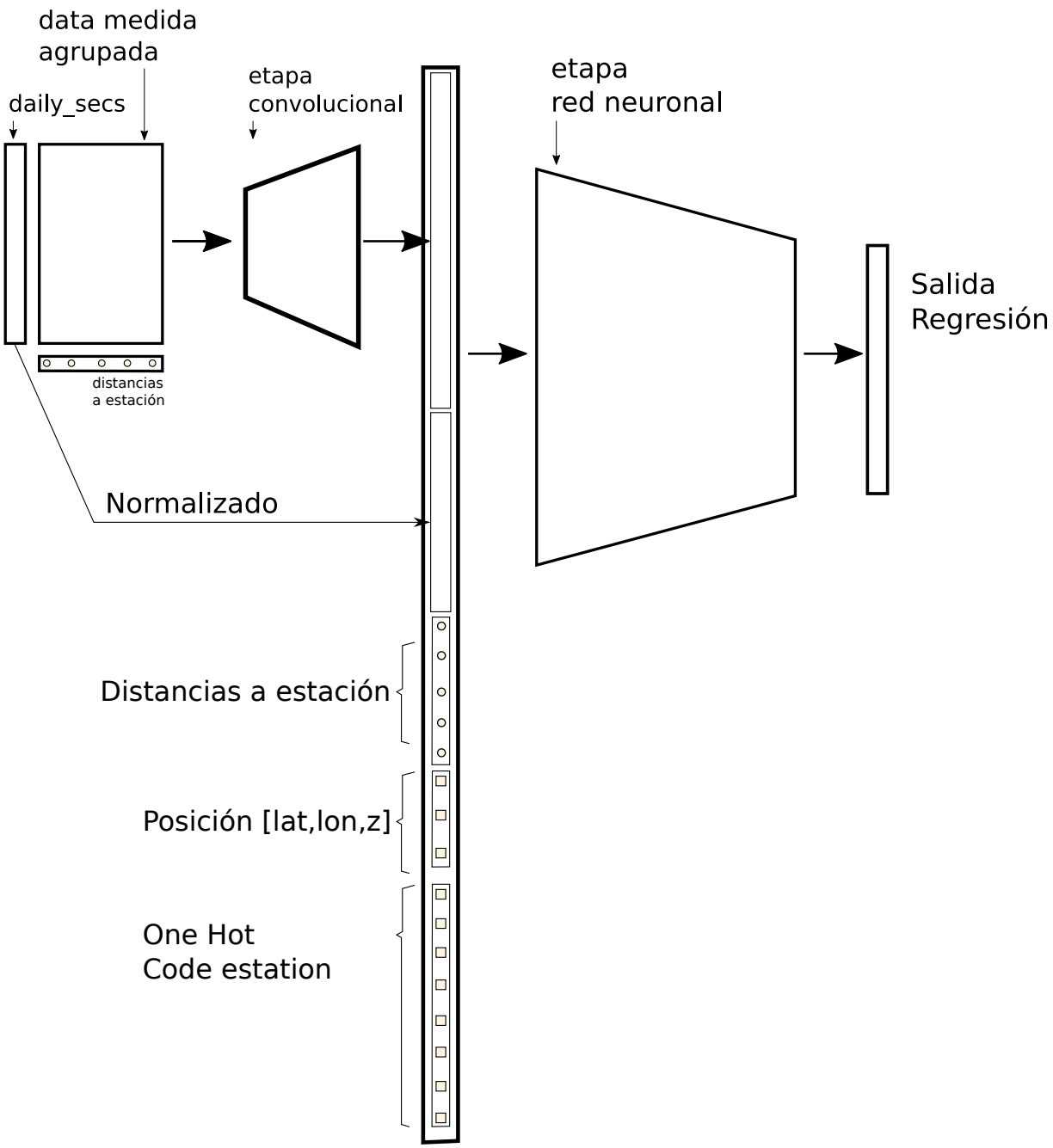

Figura 4.2: Modelo base CNN con Deep Learning

Este modelo se define con el fin de establecer una línea base de parámetros para comparar los otros modelos estudiados. Con esta misma arquitectura se estudian tres variantes.

1. Solo valores de medición del grupo de una estación.
2. Selección aleatoria de estaciones de la red.
3. Entrenamiento sobre la cobertura total.

Cada variante tiene el objetivo de encontrar alguna forma de evaluar la fortaleza del modelo frente a datasets que son producidos por diferentes fuentes, pero que mantienen la misma estructura y están ubicados en diferentes posiciones. El producto ideal es que el mismo modelo entrenado sea tan flexible que pueda ser aplicado en todo el territorio.

4.4. Modelo Baseline: CNN + Multicapa

La definición de este modelo se entrenará con los siguientes hiperparámetros.

Segundos a considerar $T = 120$, generando una predicción a futuro de $T_f = 60$, por lo que el modelo será entrenado para entregar una secuencia de $T + T_f = 180$.

La capa de convolución entregará un vector de $Z = 9796$, que se concatena con los vectores de tiempo, posición y distancia que se añaden a la salida de la convolución. A partir de aquí se procede a operar una secuencia de una etapa de red fully connected hasta la salida de predicción.

Ahora bien, para estimar la cantidad de parámetros que contiene la arquitectura de este modelo, de manera desglosada se tendrá:

En la etapa CNN, considerando los parámetros.

- K_h es la altura del kernel.
- K_w es el ancho del kernel.
- C_{in} es el número de canales de entrada.
- C_{out} es el número de canales de salida.
- El término +1 corresponde al sesgo (bias) en cada filtro.

En las capas lineales fully connected.

$$P = (K_h \times K_w \times C_{\text{in}} + 1) \times C_{\text{out}} \tag{4.5}$$

Teniendo tres capas que procesan la matríx de 120×5. Stride (1×1) y padding 0.

Capa 1 Conv2d El kernel (3×3).

$$P_{conv}(1) = (3 * 3 * 1 + 1) * 8 = 80 \tag{4.6}$$

Capa 2 Conv2d El kernel (1×1).

$$P_{conv}(2) = (1 * 1 * 8 + 1) * 48 = 432 \tag{4.7}$$

Capa 3 Conv2d El kernel (1×1).

$$P_{conv}(3) = (1 * 1 * 48 + 1) * 164 = 8036 \tag{4.8}$$

En resumen, el tamaño de la etapa convolucional considera:

$$P_{cnn} = 8548 \tag{4.9}$$

Luego, para la etapa MLP fully connected se tiene.

fc1

$$P_{fc}(1) = (9824 \times 7200) + 7200 = 70740000 \tag{4.10}$$

fc2

$$P_{fc}(1) = (7200 \times 2160) + 2160 = 15554160 \tag{4.11}$$

fc3

$$P_{fc}(1) = (2160 \times 180) + 180 = 388980 \tag{4.12}$$

El total de parámetros que determinan esta etapa son:

$$P_{mlp} = 86683140 \tag{4.13}$$

Por lo tanto, el modelo que presenta una arquitectura combinada CNN+MLP tendrá:

$$P_{total} = P_{cnn} + P_{mlp} = 86691688 \tag{4.14}$$

4.5. Modelo RNN/LSTM

Dada una secuencia de entrada, determinada por una secuencia serie-tiempo con la información del dataset ordenada, se implementa una red recurrente que tiene memoria a corto y a largo plazo de manera selectiva (LSTM: *long short-term memory*).

Se implementa una arquitectura que tenga como entrada una matriz con las serie-tiempo del grupo y sus valores característicos como distancia, tiempo y posición. Se definen en cascada varias etapas de bloques LSTM bidireccionales que operan como extractor de características.

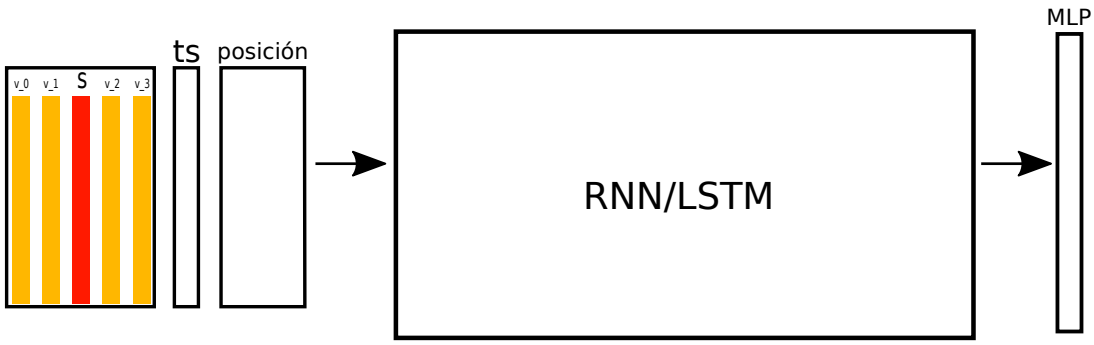

Figura 4.4: Propuesta arquitectura LSTM

A la salida de la última LSTM se conecta a una etapa de red neuronal fully connected para entegar la serie predicha 4.4.

Entre los atributos de esta arquitectura se cuenta:

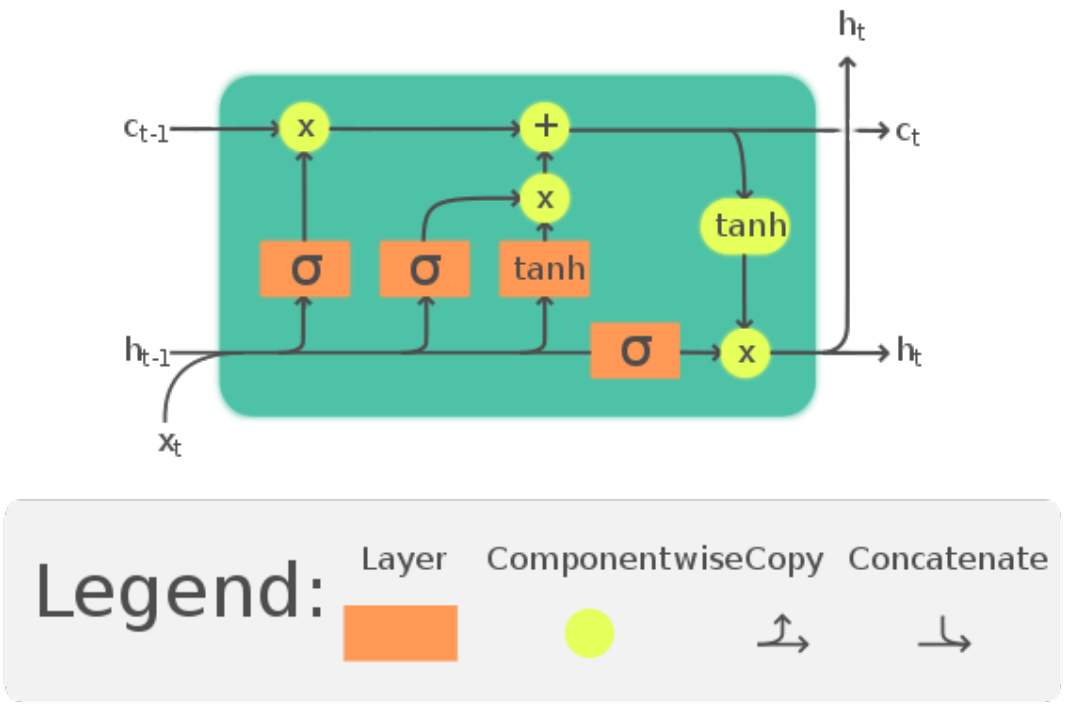

Figura 4.3: CNN/RNN-LSTM

1. Entrada de secuencia 120s
2. Es bidireccional.
3. LSTM de tres (3) capas.
4. Regresión de 120s.
5. forecasting de 60s.

La configuración de este modelo se diferencia en que no tiene capa convolucional, en reemplazo se definen tres (3) capas LSTM concatenadas en cascada. Estas tres capas realizarán el trabajo de extraer las distintas características de la serie-tiempo, considerando las relaciones de contexto que ocurren cuando existe una correlación secuencial.

En otros parámetros de ajuste se tiene.

1. dropout ajustado al 15 %.
2. capa LSTM de 3 niveles.
3. bidireccional.
4. con puerta de memoria para olvidar.
5. capa MLP *fully connected* de salida.

Para estimar la cantidad de parámetros del modelo es necesario conocer cuantos parámetros contiene cada capa. Considerando la arquitectura del modelo se tendrá:

Para cada capa LSTM bidireccional:

$$P_{LSTM}(n) = 4 \times ((input\_dim + hidden\_dim) \times hidden\_dim + hidden\_dim) \tag{4.15}$$

Para las capas fully connected:

$$P_{FC}(n) = input\_dim \text{»} output\_dim + output\_dim \tag{4.16}$$

En resumen, cada capa tendrá la siguiente cantidad de parámetros:

Capa 1 LSTM

$$P_{LSTM}(1) = 4 * ((120 + 180) * 180 + 180) = 216720 \tag{4.17}$$

Capa 2 y 3 LSTM

$$P_{LSTM}(2) = 4 * ((360 + 180) * 180 + 180) = 389520 \tag{4.18}$$

Capa Fully Connected 1

$$P_{LSTM}(1) = 180 * 1080 + 1080 = 195480 \tag{4.19}$$

Capa Fully Connected 1

$$P_{LSTM}(2) = 1080 * 180 + 180 = 194580 \tag{4.20}$$

Por lo tanto, el total de parámetros que contiene este modelo es de: 1385820. Este valor será de utilidad para calcular las métricas AIC, AICc y BIC.

4.6. Modelo Transformer

Dada una secuencia de entrada, determinada por una secuencia serie-tiempo con la información del dataset ordenada, se implementa una red transformer.

Esta arquitectura se focaliza en un **Encoder Posicional** especializado que otorga un mayor peso a las mediciones de los últimos segundos, considerando además el ajuste por entrenamiento de los parámetros asociados a cada posición.

En el contexto de la arquitectura Transformer, los elementos clave incluyen la codificación posicional y las estructuras fundamentales del codificador y decodificador. A continuación, se describe detalladamente cada componente y se proporciona un cálculo exhaustivo de los parámetros totales de la arquitectura, siguiendo convenciones académicas.

4.6.1. Codificación Posicional

Los Transformers, por su diseño intrínseco, carecen de capacidad para procesar secuencias con dependencias de orden temporal o espacial debido a la naturaleza de las capas de auto-

atención. Para remediar esta limitación, se introduce la codificación posicional, que incorpora información de la posición relativa de los tokens dentro de la secuencia.

4.6.1.1. Implementación y Variaciones

La implementación de una arquitectura de transformers considera:

Codificación Posicional Fija Utiliza funciones trigonométricas alternas para codificar las posiciones, aplicando senos y cosenos con frecuencias que disminuyen exponencialmente a lo largo de las dimensiones del espacio de embebido. Además se incluye un codificador exponencial que otorga mayor importancia a los últimos valores. Esta metodología está diseñada para permitir que el modelo capture relaciones de largo alcance efectivamente.

Codificación Posicional Aprendible En lugar de utilizar valores fijos, esta variante emplea embeddings posicionales que son optimizados durante el proceso de entrenamiento, ofreciendo una flexibilidad adicional que puede ser ventajosa dependiendo de las especificidades de la tarea.

4.6.1.2. Fundamentos de la Inclusión del Término Exponencial

Diversificación de Patrones de Codificación La codificación posicional clásica, que alterna entre funciones seno y coseno con frecuencias que varían logarítmicamente, proporciona una base efectiva para modelos que necesitan interpretar dependencias a diferentes escalas temporales. Sin embargo, esta codificación puede limitarse a capturar únicamente ciclos y periodicidades. La adición de un componente exponencial introduce un nuevo patrón de variación que puede ser particularmente útil para modelar fenómenos que no son cíclicos sino que crecen o decaen de manera exponencial a lo largo del tiempo.

Mejora de la Capacidad de Generalización Al incorporar un término exponencial en la codificación posicional, se ofrece al modelo una herramienta adicional para aprender y diferenciar posiciones dentro de secuencias largas de manera más efectiva. Esto es particularmente relevante en tareas donde la importancia de los elementos puede aumentar o disminuir exponencialmente con respecto a su posición en la secuencia.

Facilitación de la Aprendizaje de Dependencias a Largo Plazo Los términos exponenciales pueden ayudar a los modelos a ponderar de manera diferencial la relevancia de las posiciones según su distancia, lo cual es crítico en escenarios donde las dependencias no son uniformes a lo largo de la secuencia.

4.6.2. Arquitectura del Transformer

El Transformer se compone de un codificador y un decodificador, cada uno con múltiples capas que procesan la entrada de manera secuencial.

4.6.2.1. Codificador

Capas de Auto-atención Cada token en la entrada puede influir en la representación de cualquier otro token, independientemente de la distancia secuencial entre ellos, gracias a la mecanización de la atención.

Red Feed-Forward Puntual Después de la atención, cada posición pasa a través de una red completamente conectada, que es idéntica para todas las posiciones, asegurando la uniformidad del tratamiento a lo largo de la secuencia.

4.6.2.2. Decodificador

Auto-atención con Máscara Previene la fuga de información futuro a las predicciones actuales o pasadas durante el entrenamiento, esencial para tareas predictivas.

Atención del Codificador al Decodificador Facilita que cada posición en el decodificador pondere la importancia de todas las posiciones en las serie de entrada, enriqueciendo la capacidad del modelo para contextualizar y sintetizar la información recibida del codificador.

4.6.3. Cálculo de Parámetros

Para una estimación rigurosa de los parámetros en una arquitectura Transformer típica, consideremos:

Dimensiones de Entrada y Salida (*input_dim* y *output_dim*) La entrada es de 120 y la salida de 180 ya que considera el filtro de ese tramo de tiempo y la predicción a futuro.

Dimensión Interna del Feed-Forward (*dim_feedforward*) Representa la expansión interna dentro de las capas feed-forward.

Número de Cabezas de Atención (nhead) y Número de Capas Configuraciones que multiplican la complejidad y la capacidad del modelo. Número de capas: (*num_encoder_layers*, *num_decoder_layers*))

4.6.4. Fórmula Detallada para una Capa

Cada capa contiene proyecciones para consultas, claves y valores, además de una capa lineal para proyecciones finales en la atención y dos capas lineales para la red feed-forward. Sumando estos componentes se obtiene:

Parámetros por

$$Capa = 5 \times (input\_dim^2) + 2 \times (input\_dim \times dim\_feed\_forward) \tag{4.21}$$

4.6.5. Totalización de Parámetros en el Modelo

Sumando los parámetros de todas las capas del codificador y decodificador, se calcula el total como:

$$Total\_Parmetros = (Params \times num\_encoderlayers) + (Params \times num\_decoderlayers) \tag{4.22}$$

Este enfoque proporciona una estimación completa y precisa de los parámetros involucrados en una arquitectura Transformer, esencial para la evaluación de su complejidad y requisitos computacionales en investigaciones académicas y aplicaciones prácticas.

4.6.6. Parámetros de las Capas del Transformer

4.6.6.1. Parámetros de la Multi-Head Attention

La atención multi-cabeza es responsable de un conjunto significativo de parámetros, dado que involucra múltiples proyecciones lineales para las consultas, las claves, y los valores, además de una proyección para combinar las salidas de las cabezas de atención:

$$\text{Parámetros por atención} = 3 \times \left(input\_dim \times \frac{input\_dim}{num\_heads} \right) \times num\_heads + input\_dim \times input\_dim$$

$$= 4 \times input\_dim^2$$

4.6.6.2. Parámetros de la Red Feed-Forward (FFN)

Cada red feed-forward dentro de las capas del codificador y del decodificador contiene dos transformaciones lineales principales:

$$\text{Parámetros por FFN} = (input\_dim \times dim\_feedforward) + (dim\_feedforward \times input\_dim)$$

$$= 2 \times input\_dim \times dim\_feedforward$$

4.6.6.3. Parámetros Totales por Capa

La suma de los parámetros de la atención multi-cabeza y la red feed-forward da los parámetros totales por capa:

$$\text{Parámetros por capa} = \text{Parámetros de Multi-Head Attention} + \text{Parámetros de FFN}$$

$$= 4 \times input\_dim^2 + 2 \times input\_dim \times dim\_feedforward$$

4.6.6.4. Parámetros Totales de Todas las Capas

Finalmente, sumando los parámetros de todas las capas del codificador y del decodificador, obtenemos el total de parámetros en la arquitectura del Transformer:

$$\text{Total Encoder y Decoder} = (\text{Params} \times num\_encoder\_layers) + (\text{Params} \times num\_decoder\_layers)$$

$$= 2 \times (\text{Parámetros por capa} \times num\_encoder\_layers)$$

Aquí se multiplica por 2 porque se asume que el número de capas en el codificador y el decodificador es el mismo, y se desea sumar el total de ambos.

Estas ecuaciones describen de manera completa y formal el cálculo de los parámetros en una arquitectura Transformer basada en las configuraciones de capas, dimensiones y arquitectura específicas.

La cantidad todal de parámetros para la configuración entrenada es de: 1831680.

Capítulo 5

Resultados del entrenamiento de los modelos

En este capítulo se analizan y comparan los resultados obtenidos con cada arquitectura propuesta. El impacto que podrían tener como modelo general a nivel geográfico y un análisis comparativo con métricas conocidas y de performance, tanto en el entrenamiento como en operación predictiva.

5.1. Entrenamiento de modelo por estación: CNN/MLP

La duración aproximada para entrenar un modelo para una estación con solamente el dataset de la misma es de 15 minutos. La evaluación para el total de las estaciones por cada modelo toma cerca de una hora. Por lo que este análisis completo requiere cerca de cinco días para unas 72 estaciones, que es la cantidad disponible para este estudio.

El entrenamiento evalua la métrica DTW y va ajustando los parámetros, en la figura 5.1 se observa que la predicción mantiene la amplitud ajustada por la señal **target** pero la forma de onda se parece más al **input** original.

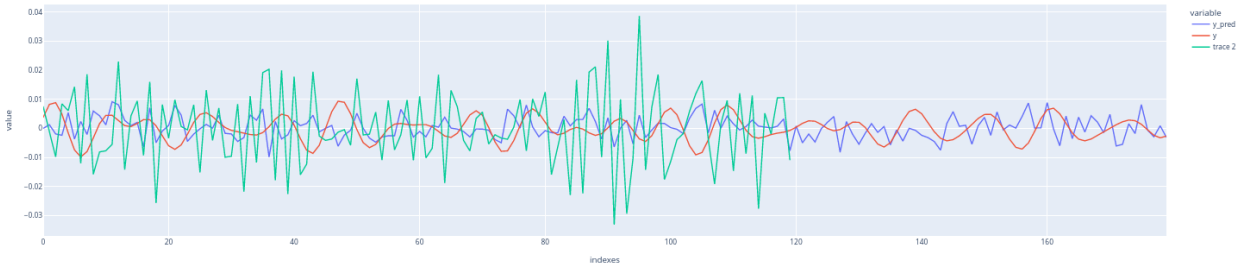

Figura 5.1: Predicción muestra UTAR

Para una muestra, en el conjunto de validación, se tiene lo siguiente:

El valor de DTW=0.0080 es reducido, lo que se puede interpretar como un buen acercamiento a la predicción pedida, además se calcula el error cuadrático medio=4.4713e-05, que permite deducir que la regresión es suficientemente correcta.

Para evaluar el efecto del modelo entrenado sobre una estación respecto el resto de la red, se toma el conjunto de evaluación del dataset y se recuperan las métricas MSE y DTW para calcular el promedio por cada estación. Esto dará un índice que que tan bueno es el modelo prediciendo otros conjuntos distintos al de entrenamiento.

Ahora bien, es interesante observar cómo se comporta el modelo entrenado para la estación MCLA comparado con el resto de las estaciones, si observamos el gráfico de barras de la figura 5.2 se podrá observar que mantiene una estabilidad en valores MSE y DTW aceptables.

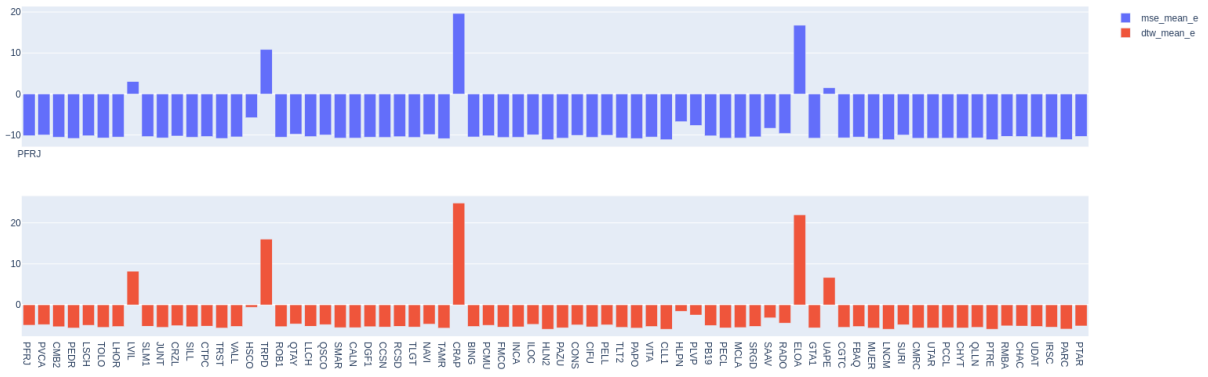

Figura 5.2: Efecto del modelo con la distancia según log(MSE) y log(DTW)

Pero también hay un conjunto de estaciones que exceden valores aceptables para estas métricas, se pueden visualizar en el gráfico de log-barra de la figura 5.3. Esto se puede interpretar directamente como un error que permite descartar el tramo de datos en particular.

Figura 5.3: Estaciones no asociables al modelo debido a que exceden el error en la predicción

Esto lleva a hacer un análisis más detallado de este conjuntos de estaciones y tratar de encontrar que podría estar sucediendo. Ubicando cada estación sobre el mapa, comparando el valor MSE/DTW se podrá ver en la figura 5.4 que la predicción realizada por el modelo si es

generalizable en la extensión territorial, pero hay casos especiales del conjunto de estaciones que están ubicadas en diferentes puntos a los que no responde de manera adecuada.

También se puede notar que hay una relación directa entre las métricas DTW y MSE. Sabiendo que el modelo se entrena con DTW y los ajustes de parámetros se realizan en base a este optimizador, MSE le hace seguimiento para tener una interpretación comparable on otros modelos que si usan MSE.

Otra manera de observar el efecto de cada modelo respecto a la predicción que realiza sobre el resto de las estaciones, será posible observar a simple visa aquellas estaciones en las que se presenta algún problema, en la figura 5.5

Aquellas estaciones que generan una predicción con un error grande tienen una tendencia al rojo, y en este caso el comportamiento desde las diferentes perspectivas (estaciones) presenta el mismo comportamiento. Incluso para aquellos modelos entrenados en las mismas estaciones (TRPD, CRAP y ELOA en la figura). Para generar una mejor diferenciación se aplica *log* al valor del error.

Corresponde preguntarse con esto, ¿qué tendrán de especial estas estaciones? ¿Qué ha sucedido que el modelo no logra hacer una predicción adecuada?

Si se realiza una revisión de la secuencia serie-tiempo original, por ejemplo para las estaciones ELOA y TRPD, es visible que presentan un comportamiento diferenciado con respecto al general de las estaciones, y parece ser que se ha configurado de manera incorrecta esa estación en particular o bien han habido errores en la red, figura 5.6.

Este resultado concreto nos muestra la robustes del modelo como concepto general, ya que presenta un comportamiento correcto frente a casos en que las estaciones se presentan activas y con señal. En caso de fallas, errores de configuración y desconexiones se puede observar que es seguro que generará un error alto, lo que puede gatillar alguna alerta que permita poner atención a estas fallas.

Dada la magnitud del dataset y la extensión geográfica del campo estudiado, se hace necesario también evaluar el efecto de los hiperparámetros de entrenamiento. Entre los principales se encuentran el factor λ, μ momentum para regular el sobreajuste local, y la cantidad de **épocas**

En los experimentos realizados se observa que un λ demasiado pequeño $<= .001$ tiene una tendencia a reducir la amplitud de la onda. Así como un factor momentum μ debe ser > 1 para aportar a regular la estabilización en mínimos locales, lo que no es deseable. El entrenamiento sobre más de una época tiene tendencia a generar un sobreajuste en el modelo, produciendo insensibilidad frente a patrones deseados. Con el ajuste de hiperparámetros $(.001, 2.3, 1)$ se hicieron pruebas satisfactorias para entrenar subconjuntos del grupo total de estaciones, lo que parece ser una recomendación aceptable (notebook:evaluate_models.ipynb).

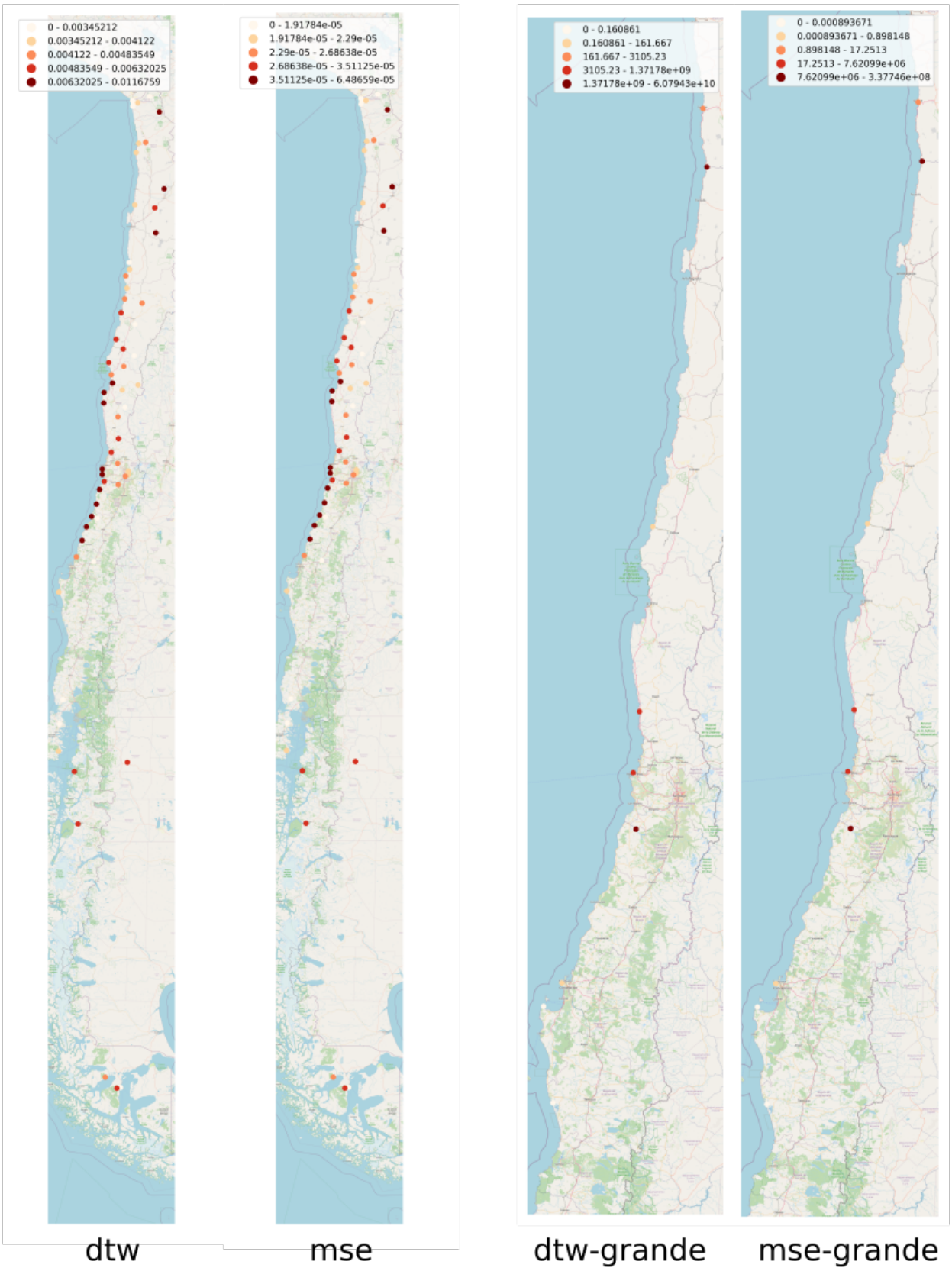

Figura 5.4: Geolocalización de cada estación, a la izquierda marcadores aceptables para el modelo, a la derecha estaciones descartadas

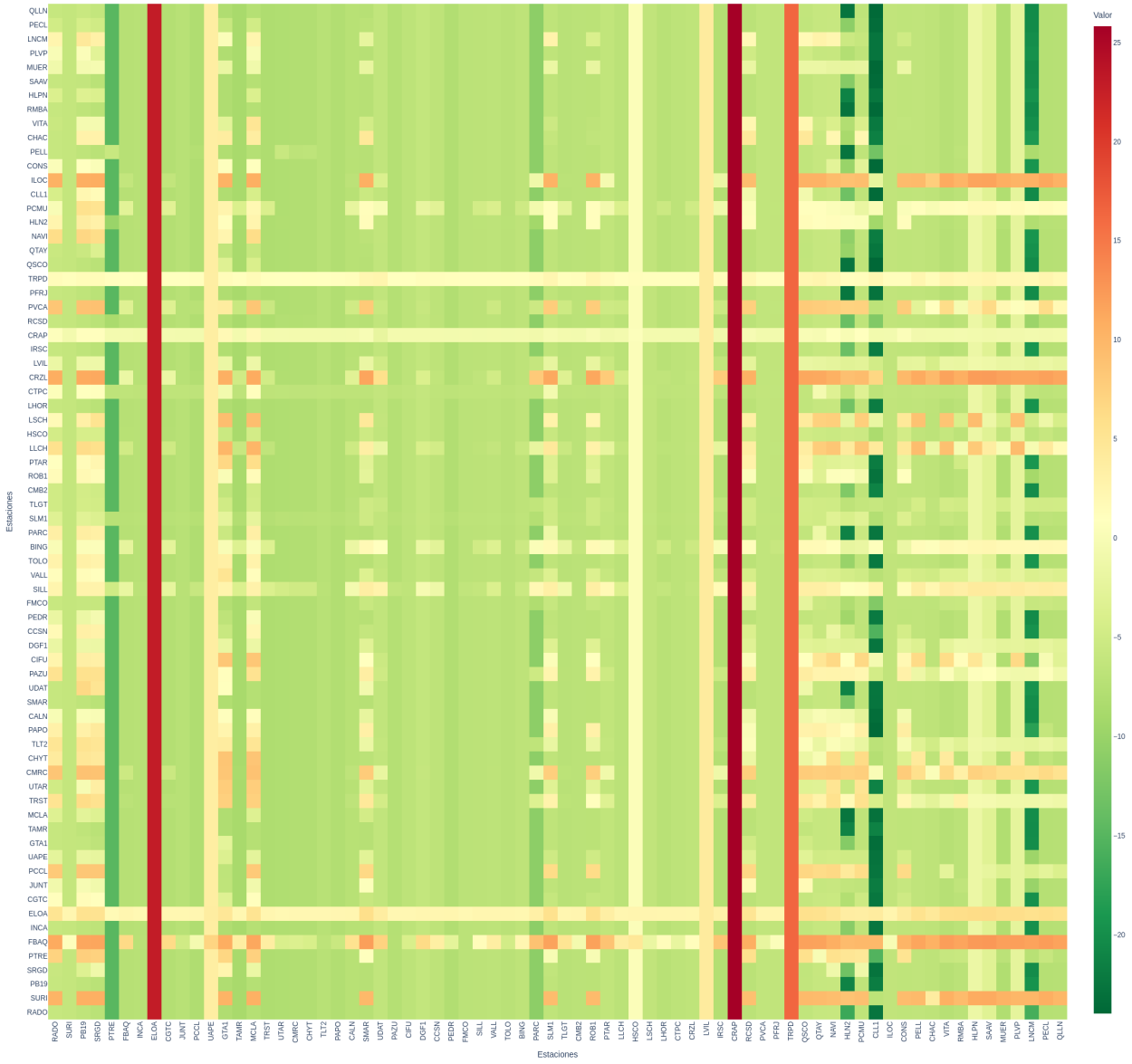

Figura 5.5: Mapa de calor modelo vs estaciones - log(mse)

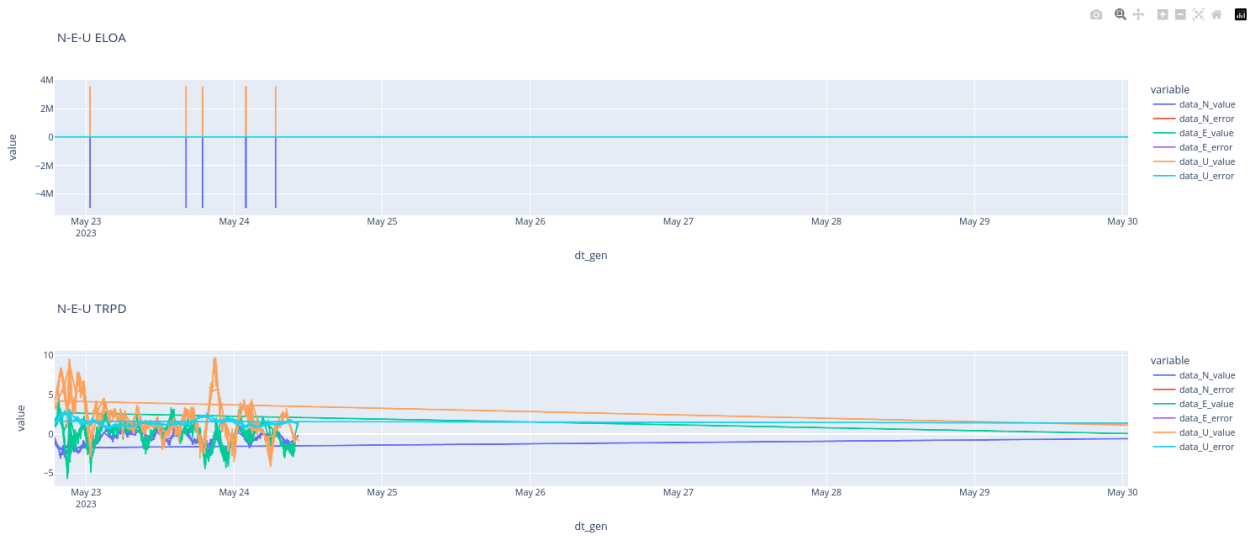

Figura 5.6: Serie-Tiempo para ELOA y TRPD último día de la selección

5.2. Entrenamiento de modelo general: CNN/MLP

Una vez determinadas las diferencias que se pueden observar en cada modelo asociado a cada estación respecto al resto de las estaciones, se abre la posibilidad de buscar un ajuste de modelo que permita realizar predicciones a nivel transversal a todo el territorio.

Dada la robustez del modelo será posible la inclusión de nuevas estaciones en la red sin perturbar o modificar de manera significativa los resultados de lo ya existente. Bastará hacer los cálculos de distancia y modificar el modelo de grafos para incluir los nuevos elementos.

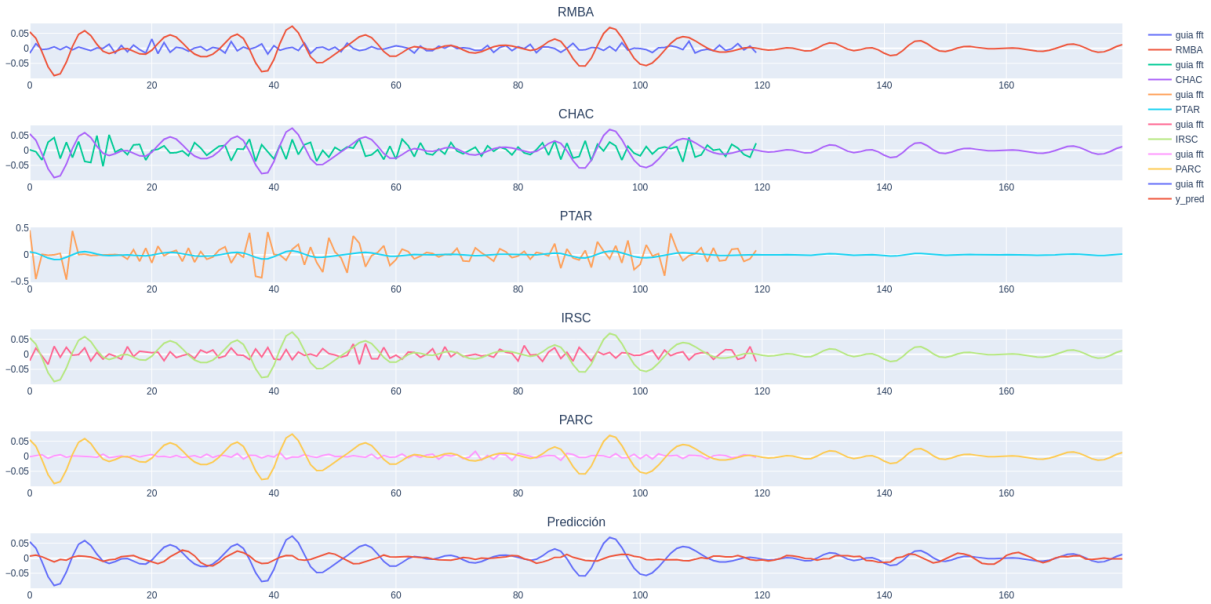

Figura 5.7: Modelo Baseline CNN/MLP|Entrada vecinos a PTAR y predicción

Para un mayor entendimiento de cómo opera el modelo se puede tomar una muestra y analizar comparativamente la entrada y la predicción. Si consideramos la señal filtrada por *fast fourier* como y_{fft} y la comprendemos como una guía más que como el valor exacto al que debe llegar la predicción. Es posible observar que el modelo ajusta la predicción ayudándose de entrada grupal y de la guía. Permitiendo así ajustar tanto la escala de la estación central (en este caso PTAR) como una predicción a futuro cercano. En la estación central, para este caso específico, se observa que los rangos de la señal cruda presenta algunos peaks de hasta un valor .5 m, lo que que no tiene sentido sismológico, el modelo ajusta de manera grupal este comportamiento y reduce la escala a rangos de 0.05.

Para evaluar el impacto de este modelo general, dado que en particular se ha tenido muy buen resultado para cada modelo respecto al resto de las estaciones, se calculan las métricas AICc y BIC para poder comparar con otras arquitecturas, como la RNN/LSTM que es lo que se realiza a continuación.

La tabla de las métricas usando DTW (tabla 5.1):

Tabla 5.1: Valores de AIC, AICc y BIC para modelo CNN/MLP (N,E,U), usando DTW

Modelo	**AIC**	**AICC**	**BIC**
U	173385063.91583547	1447.9155060350895	415037428.3836729
E	173385498.9966554	1882.996325969696	415037863.4644928
N	173384522.56321213	906.5628826916218	415036887.03104955

La tabla de las métricas usando MSE (tabla 5.2):

Tabla 5.2: Valores de AIC, AICc y BIC para modelo CNN/MLP (N,E,U), usando MSE

Modelo	**AIC**	**AICC**	**BIC**
U	173384440.7610158	824.7606863677502	415036805.22885317
E	173384875.84183967	1259.8415102362633	415037240.30967706
N	173383899.4083927	283.40806326270103	415036263.87623006

5.3. Entrenamiento de modelo: RNN/LSTM

Dado que el modelo RNN/LSTM pertenece a la familia de las redes neuronales podemos dar por sentado que el estudio de cobertura realizado con la red baseline CNN/MLP permite inducir que también se puede aplicar de manera general a todo el territorio.

Efectivamente, se puede observar en la figura 5.8 que al aplicar el modelo a toda la red y luego estudiar la cobertura frente a cada estación se tiene un valor similar del error entre todas las estaciones excepto aquellas que ya se saben hay que considerar que presentan algún problema en su señal.

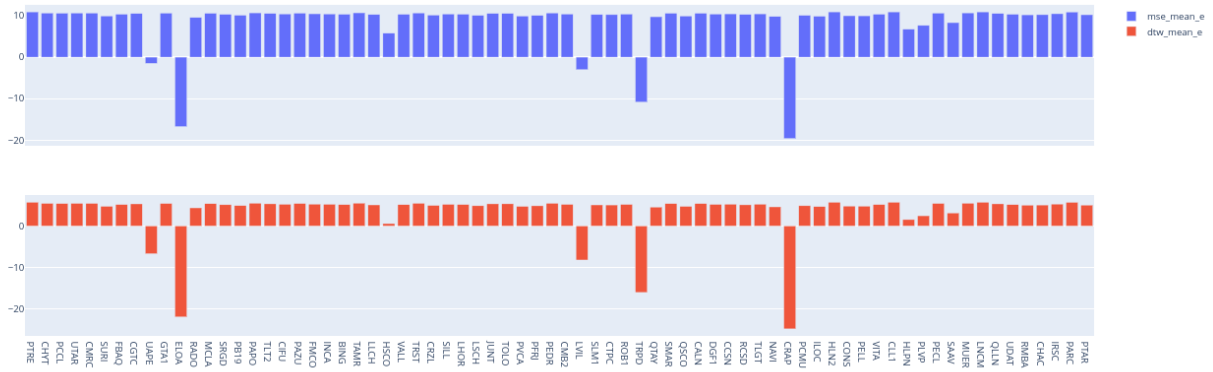

Figura 5.8: Comparativa del similitud DTW y MSE para una red LSTM aplicada a toda la red

De manera análoga al modelo general línea base (baseline), para este modelo general LSTM se realizó un experimento para un día completo en que se obtuvieron muestras de la predicción. En la figura 5.9 se puede observar que presenta un ajuste deseado y se adapta mejor a la suavidad de la onda $y_f ft$. En este sentido se puede decir que la red $LSTM$ presenta mejoras. Sin embargo, para el uso que se le podría dar a esta herramienta habría que hacer más experimentos que afinen la comparación. Al igual que con modelo baseline, el uso de la guía $y_f ft$ permite controlar señales crudas que entregan valores sin sentido sismológico.

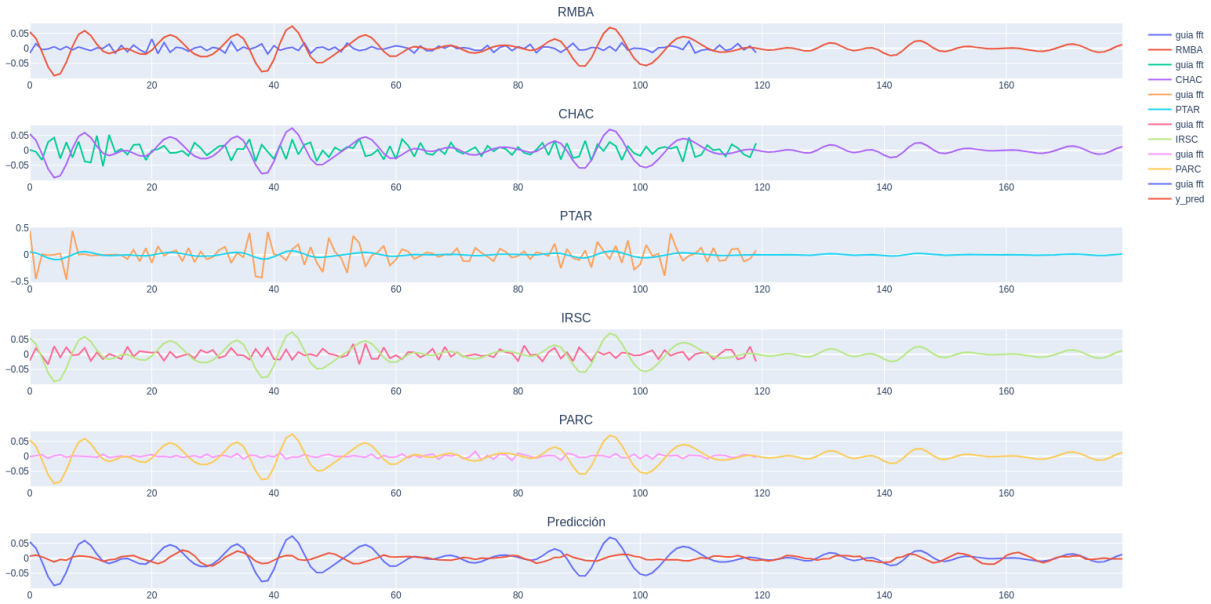

Figura 5.9: Modelo RNN/LSTM|Entrada vecinos a PTAR y predicción

En cuanto al cálculo de las métricas de información (AIC,AICc,BIC) se aplican para MSE y DTW ya que son equivalentes. Aplicando el modelo general sobre cada estación y filtrando aquellas que estén por fuera de la normal (outliers).

La tabla de las métricas usando DTW, tabla 5.3:

Tabla 5.3: Valores de AIC, AICc y BIC para modelo LSTM (N,E,U), usando DTW

Modelo	**AIC**	**AICc**	**BIC**
U	2773327.92	1447.90	6636289.72
E	2773763.00	1882.98	6636724.80
N	2772784.50	904.47	6635746.30

La tabla de las métricas usando MSE, tabla 5.4:

Tabla 5.4: Valores de AIC, AICc y BIC para modelo LSTM (N,E,U), usando MSE

Modelo	**AIC**	**AICc**	**BIC**
U	2772704.76	824.74	6635666.57
E	2773139.84	1259.82	6636101.65
N	2772161.34	281.32	6635123.15

Es interesante observar que entre ambas tablas se conserva la escala entre los cruces de ejes y métrica. Dado que estas métricas consisten en índices de información, indicando la complejidad del modelo, para AIC y BIC el modelo presenta equivalencias a través de los ejes. Sin embargo para AICc, el caso de estudio indica que acorde ya que el número de muestras es bastante mejor a la cantidad total de parámetros, es más claro ver que sobre el eje N el modelo tiene un mejor balance entre la complejidad y la cantidad de muestras. Siendo el eje de mayor interés E, ya que es por dónde genera mayor tendencia. Para esto será necesario comparar con CNN/MLP.

5.4. Entrenamiento de modelo: Terraform

Esta arquitectura de redes profunda presenta fuertes componentes que permiten relacionar los contextos dentro de la secuencia serie-tiempo y se adapta de manera adecuada al fenómeno que se estudia. De la misma manera que con LSTM y Baseline CNN/MLP, permite aplicar el modelo en la extensión de todo el territorio.

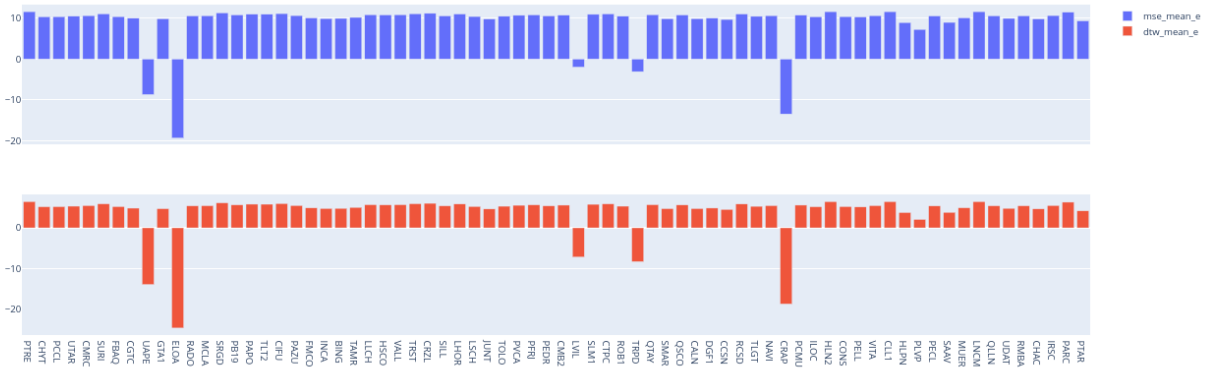

Figura 5.10: Estudio de cobertura general para Transformer

En la figura 5.10 se puede observar que se confirma la hipótesis relativa a que la familia de arquitecturas de aprendizaje profundo (deep-learning) presentan comportamientos generales similares. El modelo **tranformer** si puede aplicarse a todo el territorio y detecta anomalías o ausencias de datos.

Ahora bien, realizando el ejercicio de generar predicciones para un tramo de tiempo en todo el conjunto de estaciones esta arquitectura presenta una peor performance que LSTM ya que no ajusta la predicción a la onda guía, como se observa en la fig. **??**. Esto puede deber a un sobreajuste o bien a que podría ser necesario el uso de una arquitectura más compleja de transformers que aprenda mejor el fenómeno.

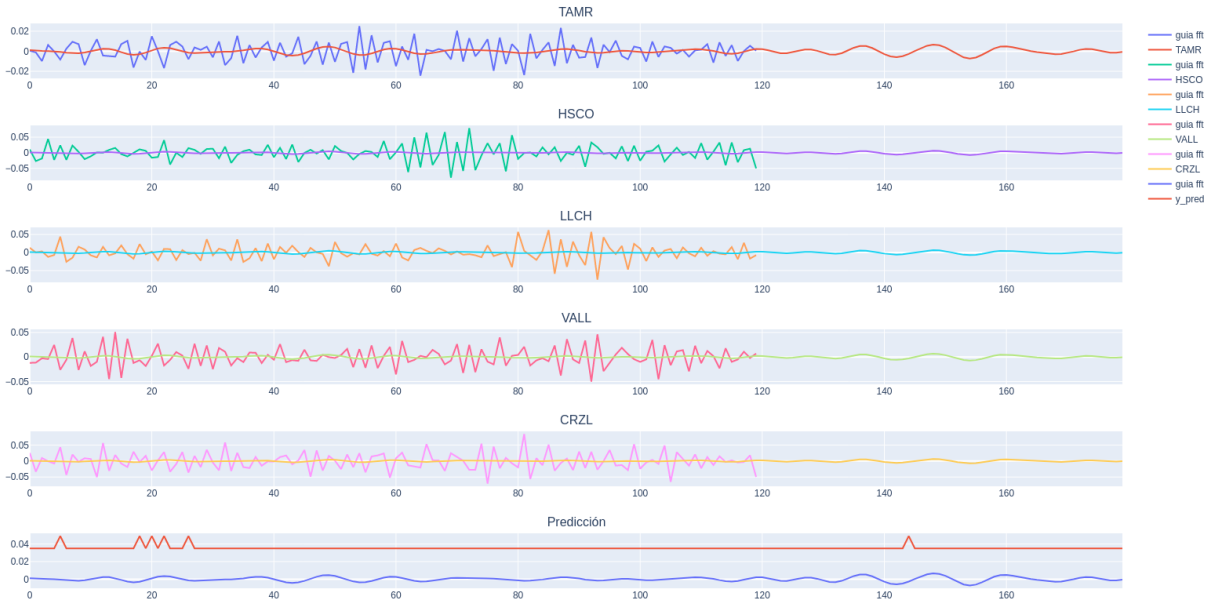

Figura 5.11: Muestra serie-tiempo predicción Transformer para NAVI

En vista y considerando que este modelo no entrega lo que se necesita no se realiza un análisis BIC/AIC para estimar la performance del modelo.

5.5. Diseño de propuesta para tiempo real

Considerando P_{ti} la predicción realizada para el tiempo i, teniendo dos componentes:

$$P_{ti} = P_{ta_i}(120s) + P_{tf_i}(60s) \tag{5.1}$$

La composición entre la predicción del residuo de la señal ahora y la futura.
El sistema para tiempo real debe considerar los siguientes pasos.

1. Convertir datos crudos en diferencias NEU
2. Extraer el residuo de la señal (sin tendencia ni periodicidad)
3. Crear la matriz de vecindad serie-tiempo por cada estación
4. Cada T periodos generar la predicción
5. Evaluar DTW entre la predicción futura en P_{t0} con predicción en P_{t1}

Definiendo un valor $\delta = 5s$ para la actualización de la predicción. Se tendrán para cada predicción actual 20 predicciones pasadas con las que hacer una evaluacíon DTW, si este factor cambia drásticamente o crece el sistema se puede elaborar una hipótess inicial de que está observando un sismo.

Considerando además que debe establecerse un proceso que mantenga cada grupo de vecindades activo, en casos de agregar nuevas estaciones o alguna se desconecte.

Técnicamente se debe desplegar un sistema que tenga la facultad de leer a la base de datos por cada estación o acceder a una cola, mantener una matriz de valores de serie-tiempo que se le vaya extrayendo los últimos δ valores, que son los valores más antiguos, y reemplazarlos por los nuevos δ valores de los últimos segundos.

Las capacidades técnicas del framework **torch** permiten hacer paquetes en *batches* de todo el conjunto de matrices y operarlas simultáneamente. El tiempo de respuesta de la predicción es menor a $1s$.

Con este flujo general de los datos se puede determinar un sistema con las siguientes características.

1. Mantener N conexiones para rescatar M estaciones por conexión.
2. Enviar los nuevos datos por cola a cada sensor-foco, es decir datos de la estación central y sus vecinos.
3. Activar el desplazamiento de los datos haciendo recortes en cada matriz de manera simultánea.
4. Acumular las matrices en batches.
5. Evaluar la predicción con el modelo.

En esto la herramienta clave para el monitoreo de movimiento sísmico consiste en un *trigger* que debe ser definido bajo un criterio y una interfaz de visualización del **desplazamiento DTW** sobre la ventada de tiempo de predicción.

Capítulo 6

Conclusiones

El presente trabajo se propuso investigar la posibilidad de diseñar un filtro predictivo de ruido y modelos de predicción basados en señales de posicionamiento geodésico (GNSS). A través del desarrollo y evaluación de diversas arquitecturas de aprendizaje profundo, se validaron aspectos clave de la hipótesis planteada y se lograron avances significativos en los objetivos establecidos.

6.1. Respuesta a la hipótesis y objetivos

6.1.1. Filtro predictivo de ruido

Se logró implementar modelos capaces de reducir significativamente el ruido en las señales GNSS. Esto se alcanzó mediante una estrategia basada en la descomposición de las señales, que permitió focalizar los esfuerzos en componentes menos dependientes del pasado, facilitando la identificación de fenómenos aleatorios y sensibles, como movimientos sísmicos.

6.1.2. Predicción en tiempo real

Los modelos RNN/LSTM demostraron ser efectivos para generar predicciones basadas en datos históricos, mostrando una superioridad en métricas como el AICc en comparación con modelos como CNN/MLP. Aunque el modelo Transformer no obtuvo los resultados esperados, su desempeño sugiere la necesidad de ajustar su arquitectura para capturar patrones más complejos.

6.1.3. Generalización y sensibilidad del sistema

Ambos modelos principales (RNN/LSTM y CNN/MLP) lograron adaptarse a la geografía estudiada, considerando las estaciones como centros locales, lo que los hace potencialmente útiles para aplicaciones prácticas como detección temprana de sismos o evaluación de sismicidad de baja magnitud. Sin embargo, los resultados también resaltan la importancia de desarrollar estrategias para descartar o ajustar estaciones con menor compatibilidad con el resto del sistema.

6.2. Principales hallazgos y aportes

- **Mejor modelo:** El modelo RNN/LSTM sobresale por su capacidad para gestionar secuencias temporales, proporcionando un mejor ajuste a las guías de onda y conteniendo de manera más eficiente el ruido de las señales.
- **Limitaciones:** Se identificaron desafíos asociados con la sincronización de datos, el manejo de estaciones que dejan de emitir, y restricciones de hardware, como la falta de memoria GPU suficiente para arquitecturas más complejas.
- **Potencial de expansión:** Los métodos de ordenamiento y acceso desarrollados permiten la incorporación futura de vectores adicionales, como velocidad o aceleración, que podrían mejorar la sensibilidad a sismos de baja intensidad.

6.3. Impacto y aplicaciones futuras

El sistema desarrollado puede servir como base para la implementación de mejoras en sistemas de monitoreo sísmico en tiempo real y como herramienta de investigación para otros sistemas que involucran la medición de ondas mediante sensores. Ejemplos claros de su aplicación incluyen la monitorización de ondas en cuerpos humanos mediante electrodos o en sistemas industriales como sensores en minas.

Finalmente, este trabajo no solo contribuye al campo de la geodesia y la sismología, sino que también abre la puerta a nuevas ideas en el análisis de señales multivariadas y sistemas distribuidos en el espacio y el tiempo.

Bibliografía

Akaike, H. (1969). Fitting Autoregressive Models for Prediction. Annals of the Institute of Statistical Mathematics **21**(), 243–247. https://www.ism.ac.jp/editsec/aism/pdf/021_2_0243.pdf

Báez, J. C., de Freitas, S. R. C., Drewes, H., Dalazoana, R., & Luz, R. T. (2007). Deformations Control for the Chilean Part of the SIRGAS 2000 Frame.. https://api.semanticscholar.org/CorpusID:127737451

Blondel, M., Mensch, A., & Vert, J.-P. (2021). Differentiable Divergences Between Time Series.

Boshnakov, G. (2016, 08). Introduction to Time Series Analysis and Forecasting, 2nd Edition. Journal of Time Series Analysis **37**(), . 10.1111/jtsa.12203

Diestel, R. 2017. Graph Theory (5th ed.). : Springer Publishing Company, Incorporated.

Duvenaud, D., Rippel, O., Adams, R. P., & Ghahramani, Z. (2016). Avoiding pathologies in very deep networks.

Elman, J. L. (1990). Finding Structure in Time. Cognitive Science **14**(2), 179-211. https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1 https://doi.org/10.1207/s15516709cog1402_1

Esmael, B., Arnaout, A., Fruhwirth, R., & Thonhauser, G. (2012, 12). Improving Time Series Classification Using Hidden Markov Models.. 10.1109/HIS.2012.6421385

Geler, Z., Kurbalija, V., Ivanović, M., Radovanović, M., & Dai, W. (2019). Dynamic Time Warping: Itakura vs Sakoe-Chiba. 2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA) (), 1-6. https://api.semanticscholar.org/CorpusID:199058384

Gelman, A., Carlin, J., Stern, H., & Rubin, D. 2003. Bayesian Data Analysis (Vol. 45). 10.2307/2988417

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. Neural Computation **12**(10), 2451–2471.

Goldberg, Y. 2017. Neural Network Methods for Natural Language Processing (Vol. 37). San Rafael, CA: Morgan & Claypool. 10.2200/S00762ED1V01Y201703HLT037

Goodman, N., Ullman, T., & Tenenbaum, J. (2011, 01). Learning a Theory of Causality. Psychological review **118**(), 110-9. 10.1037/a0021336

Grigsby, J., Wang, Z., & Qi, Y. (2021). Long-Range Transformers for Dynamic Spatiotemporal Forecasting. CoRR **abs/2109.12218**(), . https://arxiv.org/abs/2109.12218

Hochreiter, S., & Schmidhuber, J. (1997, 12). Long Short-term Memory. Neural computation **9**(), 1735-80. 10.1162/neco.1997.9.8.1735

Hurvich, C. M., & Tsai, C. L. (1989). Regression and time-series model selection in small samples. Biometrika **76**(), 297–307.

Jeffrey T. Freymueller, A. B. A. N. B. B. Y. F. N. K. K. L. H.-P. P., Rebecca Bendick, & van Dam, T. (2019). Measuring the Restless Earth. Grand Challenges in Geodesy. http://geophysics.eas.gatech.edu/anewman/research/papers/Freymueller_etal_measuring-the-restless-earth_2019.pdf

Lam, M. W. Y., Chen, X., Hu, S., Yu, J., Liu, X., & Meng, H. (2019). Gaussian Process Lstm Recurrent Neural Network Language Models for Speech Recognition. En ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (p. 7235-7239). 10.1109/ICASSP.2019.8683660

Leyton, F., Ruiz, S., Baez, J. C., Meneses, G., & Madariaga, R. (2018). How Fast Can We Reliably Estimate the Magnitude of Subduction Earthquakes? Geophysical Research Letters **45**(18), 9633-9641. https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018GL078991 https://doi.org/10.1029/2018GL078991

MacKay, D. J. C. (1998). Introduction to Gaussian processes.. https://api.semanticscholar.org/CorpusID:116281095

Navarro, H.-K. N. M. L., C. (2014). A Survey on Parallel Computing and Its Applications in Data-Parallel Problems Using GPU Architectures. Communications in Computational Physics **15**(2), 285–329. http://global-sci.org/intro/article_detail/cicp/7096.html https://doi.org/10.4208/cicp.110113.010813a

Neal, R. M. 2012. Bayesian learning for neural networks (Vol. 118). : Springer Science & Business Media.

Plafker, G., & Savage, J. C. (1970, 04). Mechanism of the Chilean Earthquakes of May 21 and 22, 1960. GSA Bulletin **81**(4), 1001-1030. https://doi.org/10.1130/0016-7606(1970)81[1001:MOTCEO]2.0.CO;2 10.1130/0016-7606(1970)81[1001:MOTCEO]2.0.CO;2

Porr B, B. L. C. H. D. R. ., Daryanavard S. (2022). Real-time noise cancellation with deep learning. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0277974 https://doi.org/10.1371/journal.pone.0277974

Ruiz, S., Aden-Antoniow, F., Baez, J. C., Otarola, C., Potin, B., del Campo, F., … Bernard, P. (2017). Nucleation Phase and Dynamic Inversion of the Mw 6.9 Valparaíso 2017 Earthquake in Central Chile. Geophysical Research Letters **44**(20), 10,290-10,297. https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017GL075675 https://doi.org/10.1002/2017GL075675

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing **26**(1), 43-49. 10.1109/TASSP.1978.1163055

Salvador, S., & Chan, P. (2004, 01). Toward Accurate Dynamic Time Warping in Linear Time and Space. En (Vol. 11, p. 70-80).

Sarle, W. S. (1995). Stopped Training and Other Remedies for Overfitting. En Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics (pp. 352–360).

Schwarz, G. (1978). Estimating the Dimension of a Model. Annals of Statistics **6**(), 461–464.

Shumway, R., & Stoffer, D. 2010. Time Series Analysis and Its Applications: With R Examples. : Springer New York. https://books.google.cl/books?id=ahdvcgAACAAJ

Shumway, R. H., & Stoffer, D. S. 2000. Time Series Analysis and Its Applications. : Springer.

Thomas, A., Melgar, D., Dybing, S. N., & Searcy, J. R. (2023, May). Deep learning for denoising High-Rate Global Navigation Satellite System data. Seismica **2**(1), . https://seismica.library.mcgill.ca/article/view/240 10.26443/seismica.v2i1.240

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2023). Attention Is All You Need.

Visser, I., Raijmakers, M., & Maas, H. (2009, 07). Hidden Markov Models for Individual Time Series. En (p. 269-289). 10.1007/978-0-387-95922-1_13

Wegener, A. (1912). die entstehung der Kontinente. Petermanns Geographische Mitteilungen (), 185-195.

Anexos

Anexo A: Diseño de Software

Este anexo consta de las diferentes implementaciones de software realizadas para entrenar los diferentes modelos.

A.1. Definición Base

Dado que el sistema GNSS consta de tres componentes axiales, se debe considerar en el software implementado una forma inteligente de procesar cada uno de estos. Se define un objeto abstracto **GNSSObject** que puede ser cualquier estrucutura de información asociada a cada eje.

Para eso se propone una clase padre **Switcher** fig. A.1 que permita intercambiar rápidamente el eje y, en consecuencia, todos los componentes ligados.

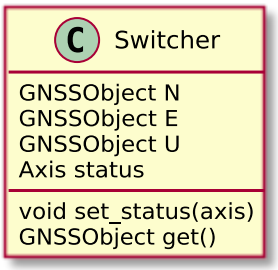

Figura A.1: Clase Switcher para intercambiar ejes

Se define una enumeración Axis fig. que permitirá establecer el estatus de que eje se está operando en el momento.

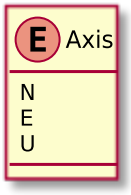

Figura A.2: Definición de enum Axis

A.2. Preprocesamiento

Estas clases permiten procesar cada dato del dataset de manera adecuada y estandar, además de proveer algunas funcionalidades para calcular las pendientes en un tramo.

A.3. Iteradores

Teniendo en cuenta que el dataset completo está disponible en archivos **data** para cada estación. Por razones de tamaño no es posible mantener todo el dataset cargado en memoria, por lo que se aplica una técnica que permita acceder con discreción sectores de datos asociados según el modelo de grafos por vecindad.

Se tiene **Dataset** fig. A.3 que consta de un registro de rutas y de grupos por vecindad, con los que se lee el conjunto de archivos y luego se itera por ventanas sobre la serie-tiempo completa.

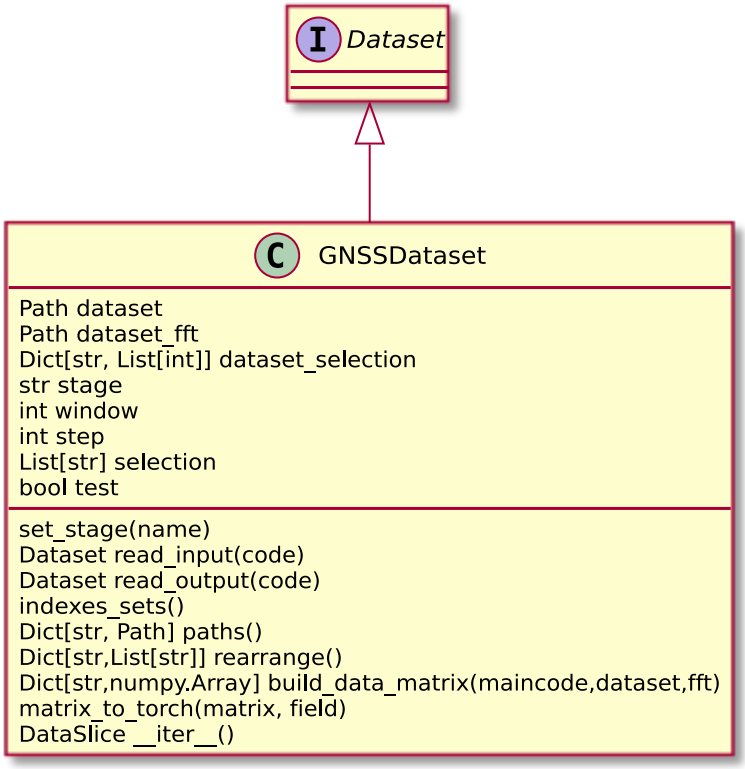

Figura A.3: Definición de clase Dataset con algoritmo de lectura eficiente

Luego, la clase **DataSlice** fig. A.4 está enfocada en gestionar las matrices asociadas al tramo de tiempo correspondiente de todo el grupo de vecindad, usando torch ya que permite cargarlas a la GPU.

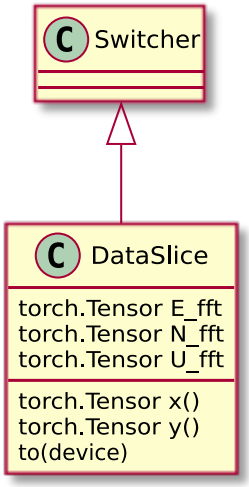

Figura A.4: Definición de clase DataSlice para contener los datos de un tramo específico

Por último, la clase **DataLoader** fig. A.5 permite asociar ventanas de Dataset en lotes para entregarlos al algoritmo de entrenamiento.

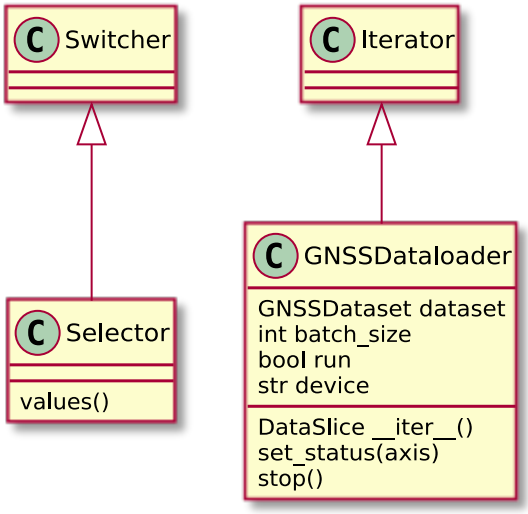

Figura A.5: Definición de clase DataLoader para entregar la data en lotes

A.4. Modelos

Los modelos de máquinas de aprendizaje definidos por clase.

A.5. Entrenamiento

Primero, se necesita una clase que permita cambiar de eje para la ejecución del entrenamiento del modelo, esta clase se hereda en el conjunto de clases que se usan en la creación del modelo y el entrenamiento.

Se define una interface genérica fig. A.6 que se relaciona con cada objeto que se instancia por cada uno de los ejes, dependiendo de la clase particular que lo usa.

Figura A.6: Todo objeto que va en un Switcher N,E,U se considera GNSSObject

Las clases que heredan de Switcher fig. A.7 tendrán un comportamiento similar, al cambiar de eje cambiaran de modelo, criterio y función de pérdida.

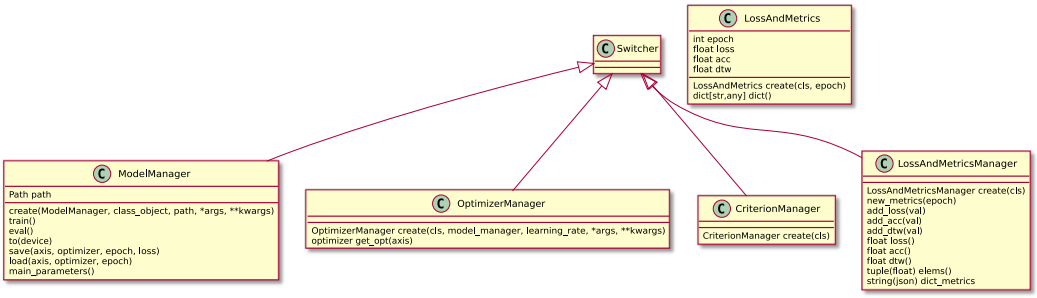

Figura A.7: Definición de clases que heredan de Switcher

Se define una clase UniversalManager fig. A.8 que contiene los elementos principales del entrenamiento, con el fin de que cada acción de cambio del switcher se gestione de manera global.

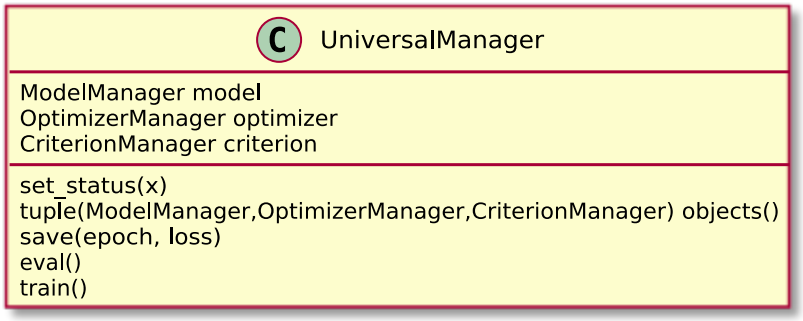

Figura A.8: Definición de UniversalManager para la gestión general del entrenamiento

También, se define la etapa del entrenamiento que se está ejecutando, controlando esto con las clases Stage fig. A.9 y UniveralIterator que gestiona los iteradores sobre el dataloader.

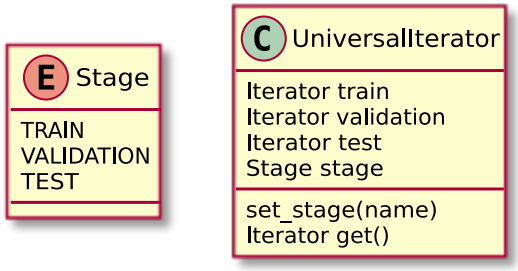

Figura A.9: Definición de clase Stage

A.6. Administración de los modelos: API Rest

Para registrar cada entrenamiento de modelo, sus parámetros y las rutas en que se mantendrán almacenados los archivos de los modelos se crean los siguientes modelos que repre-

sentarán cada uno una tabla en una base de datos **postgres**.

1. Machine
2. EpochRegister

También se define una clase Manager que contiene un cliente conectado a la base de datos y los métodos que permiten interactuar con Machine y EpochRegister.

Con esto, la interfaz que atiende la Rest Api contendrá las siguientes funciones y rutas.

En las solicitudes GET se tiene:

1. get_machine
2. get_register
3. get_machines
4. last_register
5. get_params
6. get_acc_loss

En las solicitudes POST se tiene:

1. new_machine
2. new_register

Este servicio se mantiene activo en un servidor alojado en *DigitalOcean*, de manera que cada vez que se realiza un experimento se registren sus parámetros, teniendo así una estimación del monto de experimento y registro de sus productos principales.