**UNIVERSIDAD POLITÉCNICA DE MADRID**
**Escuela Técnica Superior de Ingeniería de Sistemas Informáticos**


# Large scale Time Series Visual Analytics combining Data Mining and Deep Learning

## DOCTORAL THESIS

Submitted for the degree of Doctor by:

**María Inmaculada Santamaría Valenzuela**
Master's Degree in Mathematics and Secondary Education in Mathematics

Madrid, 2025



UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

**Doctoral Degree in PhD in science and computer technologies for Smart Cities**

# Large scale Time Series Visual Analytics combining Data Mining and Deep Learning

## DOCTORAL THESIS

Submitted for the degree of Doctor by:

**María Inmaculada Santamaría Valenzuela**
Master's Degree in Mathematics and Secondary Education in Mathematics

Under the supervision of:
Dr. David Camacho Fernández (Supervisor)
Dr. Victor Rodríguez Fernández (Cosupervisor)

Madrid, 2025

Title: Large scale Time Series Visual Analytics combining Data Mining and Deep Learning

Author: María Inmaculada Santamaría Valenzuela

Doctoral Programme: PhD in science and computer technologies for Smart Cities

Thesis Supervision:

Dr. David Camacho Fernández, Catedrático de Universidad, Universidad Politécnica de Madrid (Supervisor)

Dr. Victor Rodríguez Fernández, Contratado Doctor, Universidad Politécnica de Madrid (Co-supervisor)

External Reviewers:

Thesis Defense Committee:

Thesis Defense Date:

This work has been funded by Grant PLEC2021-007681 (XAI-DisInfodemics) and PID2020-117263GB-100 (FightDIS) funded by MCIN/ AEI/10.13039/501100011033; by "ERDF A way of making Europe", by the "European Union" or by the "European Union NextGenerationEU/PRTR"; by EMIF managed by Calouste Gulbenkian Foundation, under the project MuseAI - Detecting and matching suspicious claims with AI; by "Convenio Plurianual with the Universidad Politécnica de Madrid in the actuation line of Programa de Excelencia para el Profesorado Universitario"; by PCI2022-134990-2 (MARTINI) of the CHISTERA IV Cofund 2021 program, funded by MCIN/AEI/10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR"; by Grant PLEC2021-007681 (XAI-DisInfodemics); by the research project CIVIC: Intelligent characterisation of the veracity of the information related to COVID-19, granted by BBVA FOUNDATION GRANTS FOR SCIENTIFIC RESEARCH TEAMS SARS-CoV-2 and COVID-19, by European Commission under IBERIFIER - Iberian Digital Media Research and Fact-Checking Hub (2020-EU-IA-0252); by H2020 TMA-MSCA-DN TUAI project "Towards an Understanding of Artificial Intelligence via a transparent, open and explainable perspective" (HORIZON-MSCA-2023-DN-01-01, Grant agreement nº: 101168344); by Strategic Networking and Development Program funded by the Ministry of Science and ICT through the National Research Foundation of Korea (RS-2023-00267476); by European Commission under IBERIFIER Plus - Iberian Digital Media Observatory (DIGITAL-2023-DEPLOY- 04-EDMO-HUBS 101158511); by Comunidad Autonoma de Madrid, CIRMA-CAM Project (TEC-2024/COM-404); and by the European Union funds under the DIGITAL-2023-SKILLS 05, Topic DIGITAL-2023-SKILLS-05-CYBERACADEMY, through the project CYBERSECURITY for Industry 4.0 Technologies in Operational Technologies (CYBERSEC4OT).

# Agradecimientos

La historia comienza en Málaga, donde una pequeña cabeza forja su personalidad de la mano de familia, compañeros y docentes, a los cuales agradeceré siempre su compañía y apoyo. Una decisión que no habría sido posible sin el ejemplo de mi padre, a quien agradezco su constancia en el arte de la escritura y los estudios así como la capacidad de concentración y el de mi madre, con el impulso para continuar hasta el final en los estudios avanzados en matemáticas aunque hayan acabado en el lado oscuro de la estadística y la informática. Gracias también a mi hermano, por ser siempre la punta de lanza, las reflexiones en carretera y las canciones más aleatorias. Gracias titos y primos por la acogida cada vez que bajo y gracias a los que ya no están por los valores transmitidos. Gracias, Gloria, Inés, Ro y Joaquín (e Inés C.) por seguir aguantando la decisión. Gracias también, Iván, Estefi, Isa, José Francisco, Laura... (PUMA) por los aprendizajes en los que hemos avanzado. Gracias, Paloma, por cuidarme por teléfono y coro de Málaga y comunidad por mantener el significado de Casa cada vez que paso por allí. Gracias también, Guille Locke por poner la semilla de aprender LaTeX antes siquiera de empezar la carrera y mantenerte atento a los pequeños pasos. Gracias Dani, Arturo y Lidia por seguir atentos desde entonces. Y Rick y José Ángel, que todavía conseguís avivar el chat con retos matemáticos que no consigo entender.

De camino por Madrid, no me olvido de la yaya, que antes de ofrecerse la oportunidad me insistía en que algún día acabaría aquí, y aquí estoy, porque la consultoría se me quedaba corta. Y de Alba, Marta y Jose que me acompañaron en esa etapa y aún tengo en cuenta en las pequeñas decisiones. Y mucho menos me olvido de quien conocí poco después y que me ha ayudado en el desarrollo personal e informático de esta persona, evitando la desmotivación y levantando las caídas y tirando para delante de la casa en los momentos intensos. Te quiero, Enrique. Gracias, Tere (sargento), que a pesar del tiempo que hace desde que nos conocimos en la empresa, sigues apoyando. A Marta, Anto, Dani (atún), Mar, Marcos, Hua, Fer, Dani (Sevilla) e Irene, que viniendo de otro mundo me acogísteis como una más tras conocer a Enri y de un modo u otro me habéis echo formar parte de otras familias que me han apoyado e iluminado en este camino, guiando el proceso de formas más técnicas o espirituales. Gracias Natalia y coro de la Parroquia de Santa María del Val, por acompañarme en aquella época.

Pasando por Valladolid. A Edu, Mono, Fran, Sergio, Yuri, Arturo y, sobre todo, Rocío, Silvia y Arancha: muchas gracias por acompañarme en el primer intento, como guía, amiga y segunda madre. Abristeis una oportunidad que aquí continúa. Gracias a la familia del Coro 9 y 15, en especial, Alvarito y Cintia, por seguir aún a día de hoy ahí con vuestra sonrisa particular y a Edu por su acompañamiento y al grupo de vida por su suporte para no caer.

De vuelta por Madrid, gracias Sole, Martín, Nuri, Miguel, Carmen por la acogida en esta nueva casa y el apoyo en el día a día. Helena, gracias por aguantarme,

apoyarme, aconsejarme y obligarme a tomarme descansos antes de que Ángel me quite el carné de cordones (gracias, Ángel, por acercarte en los momentos de agobio). Sergio "argentino", gracias por sortener mi espalda y escucharme cuando te he necesitado. Sergio "psicólogo", Alex Barreiro, Dani, Hugo, Clara, Adri, Pablo gracias por estar ahí como amigos y confiar. Javi H., gracias por sacarme de los apuros más imprevistos cuando ha sido necesario. Javi T., gracias por seguir atento en la distancia y por los recuerdos que sigo guardando con cariño. Gracias Guille por tu cariño y cercanía en cada seminario. Gracias Cristian por tu ayuda con Moodle y los ánimos con R cuando he estado agobiada. Gracias Aurea, Julio, Jorge, Rafi, Paqui, Vane, Alfonso, Alfredo, Estrella, Natalia y todos los que en el pasillo me sacáis una sonrisa al pasar por allí. A Casanova y Juanma que me han soportado en los despachos con lecciones de vida que para mí se quedan. Gracias, Nana, Francisco, Óscar y todos los hermanos del coro de Madrid por cuidar por el camino, respetando los fallos.

Final y especialmente, gracias a David y Víctor (y a MA que me acercó conocerles, y Sonia y Marina por vuestra comprensión y cariño) por guiarme en esta segunda oportunidad y soportarnos entre todos para conseguir que llegue a buen término.

A todos vosotros y a todos los que no he mencionado, gracias por hacerme llegar hasta aquí.

# Resumen y Discusión General

## Resumen

El análisis de series temporales toma un papel crucial en aplicaciones científicas e industriales, facilitando tareas como la detección de anomalías, predicción, la detección de patrones y la segmentación. A pesar de su importancia, las aproximaciones clásicas basadas en Inteligencia Artificial suelen tener problemas de escalabilidad e interpretabilidad. Esta tesis presenta una aproximación novedosa para el análisis de series temporales largas que integra técnicas de Minería de Datos y Aprendizaje Profundo para mejorar la eficiencia e interpretabilidad en la exploración de series temporales.

La metodología y el trabajo de investigación, desarrollado en esta tesis se basan en la herramienta DeepVATS, una herramienta de análisis visual de series temporales diseñada para mejorar la interpretabilidad de las técnicas basadas en Aprendizaje Profundo. Se basa en la interacción con el espacio latente de los modelos que fundamentan dichas técnicas y el gráfico que contiene los datos originales de la serie temporal. La arquitectura que fundamenta la aplicación es "Masked Timeseries AutoEncoder", una arquitectura de Aprendizaje Profundo que captura las características estructurales de las series temporales en su espacio latente, permitiendo un análisis visual interactivo de calidad de la serie temporal para las tareas previamente mencionadas. Sin embargo, no tiene la capacidad de detectar la tendencia de las series temporales.

La metodología dual propuesta se define para mejorar la herramienta de análisis visual mediante la integración de técnicas de Minería de datos y los incipientes modelos fundacionales para series temporales, añadiendo interpretabilidad y reduciendo el tiempo de ejecución durante el análisis.

La primera parte de la metodología usa la matriz de similitud (MPlot) como una herramienta para el análisis preliminar con el objetivo de detectar patrones y anomalías, reduciendo el tiempo de espera requerido para obtener un primer análisis a la vez que se entrena el modelo de Aprendizaje Profundo en el que se basa la aplicación. Esta aproximación permite un análisis más interactivo y eficiente en recursos de series temporales largas, añadiendo en DeepVATS la utilidad de detectar tendencias en series temporales, cuya ausencia era una limitación clave de la herramienta.

En la segunda parte de la metodología, la investigación evalúa la efectividad de los modelos fundacionales para series temporales para modelar las propiedades de las series temporales y capturarlas en la topología de su espacio latente. Con la integración del modelo fundacional más influyente que está preparado para múltiples tareas de series temporales multivariables (MOMENT) en DeepVATS, este estudio explora cómo las representaciones aprendidas por dicho modelo pueden mejorar tanto la precisión como la eficiencia en las tareas propuestas.

Los resultados experimentales muestran cómo la combinación de las matrices de similitud con las técnicas de aprendizaje profundo proveen de un entorno escalable e inter-

pretable para series temporales. La aproximación propuesta no solo acelera el proceso de exploración sino que también mejora la detección de tendencias sin requerir demasiadas configuraciones manuales de parámetros.

Las contribuciones de esta tesis tienen amplias implicaciones para distintos dominios, entre los que se incluyen el análisis financiero, el diagnóstico médico o la monitorización de procesos industriales. La investigación prepara el camino a la integración de metodologías más interactivas e interpretables dentro de las herramientas de análisis visual de series temporales.

# Discusión General

La disertación, su metodología e implementaciones se llevan a cabo con el fin de responder a las preguntas en la Sección 1.2. En el primer paso, se realiza un análisis de escalabilidad que muestra cómo el entorno de experimentación tiene un rendimiento excelente en el análisis de conjuntos de datos pequeños y medianos, pero afronta problemas de rendimiento a la hora de analizar conjuntos de datos más extensos (ver Sección. 3.4). Para arreglar el problema, el análisis se centra en añadir herramientas del estado del arte que mejoren la eficiencia y la interpretabilidad. En el desarrollo de la tesis, que utiliza la herramienta DeepVATS (entorno de análisis visual e interactivo de series temporales basado en Deep Learning (DL)) como entorno de experimentación, se han analizado dos herramientas diferentes: los gráficos de las matrices de similitud (MPlot) y los modelos fundacionales para series temporales (TSFM).

Las secciones siguientes muestran los resultados de seguir el path propuesto en la tercera solución, concretada en dos preguntas de investigación (RQ, *research question*) analizadas a lo largo de la tesis. Se muestran también las líneas futuras de trabajo.

### Primera pregunta de investigación

La primera pregunta de investigación (**RQ1**) es "¿Pueden los MPlot ayudar en el desarrollo de herramientas de análisis visual basadas en aprendizaje profundo más eficientes e interpretables?". La respuesta a esta pregunta se analiza en base a dos preguntas más concretas (ver Sección 4.1: MPlot integration into DeepVATS).

La primera pregunta, "¿Son los MPlot suficientemente eficientes en su versión reducida a 10K puntos de la serie temporal? Esto es, ¿mejoran la eficiencia del modelo central de DeepVATS (MTSAE)?" (**RQ1.1**), se relaciona con la eficiencia. La tabla 4.1 muestra cómo el uso de una versión reducida a 10K puntos puede suponer una diferencia grande en el tiempo de ejecución, reduciéndolo de más de dos horas a siete segundos. Es, por tanto, evidente que esta versión es mucho más rápida que el algoritmo original (MTSAE), aunque podría perder información debido a una compresión de los datos originales.

La segunda pregunta, "¿Son los Distance Matrix Plot (MPlot) fáciles de interpretar? Concretamente, ¿superan las capacidades de DeepVATS en la representación de comportamientos de series temporales?", está relacionada con las capacidades de los MPlot para la detección de patrones y la facilitación de su detección en una inspección visual para verificar si mejoran los resultados previos obtenidos con MTSAE. Para responder esta pregunta, los MPlot se han integrado en DeepVATS y se han comparado tanto en tiempo de ejecución como en un análisis visual contra MTSAE. Los resultados indican que esta aplicación es muy buena para la detección de tendencias tanto locales como globales, pero quizá no sea tan buena como MTSAE para la detección de patrones o anomalías en un primer vistazo. Como MTSAE no ha mostrado capturar correctamente la tendencia de las series temporales, la adición de los MPlot en DeepVATS resulta en una mejora para el análisis visual de tendencias.

Juntando ambas respuestas, podemos deducir que los MPlot pueden mejorar la eficiencia de DeepVATS permitiendo su ejecución de manera simultánea al entrenamiento del modelo. De este modo, se obtiene una respuesta positiva para **RQ1**, limitándose su uso a series univariables y el hecho de que quizás se complique el análisis de las anomalías dependiendo de las características de la serie temporal.

Para resolver estos problemas, se proponen de manera inicial dos líneas futuras. La primera, basada en comprobar el uso de variables relacionadas con los MPlot para mejorar el análisis, aún no ha sido implementada. Sin embargo, la segunda, basada en incluir modelos fundacionales para series temporales en el módulo de aprendizaje profundo de DeepVATS, da pie a la segunda parte de la metodología y su análisis se muestra en la sección siguiente.

### Segunda pregunta de investigación

La segunda pregunta de investigación (**RQ2**) es "¿Pueden los modelos fundacionales para series temporales mejorar el desarrollo de herramientas para el análisis visual de series temporales basadas en aprendizaje profundo?". Esta sección muestra los detalles obtenidos a partir de esta pregunta a partir del análisis realizado.

En el estado del arte, se muestra cómo la familia de modelos fundacionales para series temporales MOMENT captura los comportamientos de las series temporales en su espacio latente [1]. Sin embargo, los experimentos llevados a cabo en el entorno de experimentación de DeepVATS no reproducen los mismos resultados: el espacio latente generado por MOMENT no conseguía capturar las dinámicas de las series temporales de manera tan efectiva como lo hace el modelo MTSAE desarrollado para DeepVATS en ninguna de las tres variantes preentrenadas. Como resultado de esta discrepancia, se obtienen tres preguntas de investigación más específicas:

- **RQ2.1** "¿Como de bien capturan se capturan los comportamientos de las series temporales en los espacios latentes de los modelos fundacionales para series temporales? ¿Son más sencillos; o, al menos, similares en dificultad, de interpretar que aquellos espacios latentes arrojados por el modelo MTSAE de DeepVATS?"
- **RQ2.2** "¿Suponen mejoras cuantitativas en los resultados de tareas clásicas de análisis de series temporales una mejora en la interpretabilidad del espacio latente?"
- **RQ2.3** "¿Mejora la interpretabilidad al afinar los modelos? ¿Cuál es el grado de ajuste necesario para obtener agrupaciones descriptivas y de alta calidad para una serie temporal en específico?"

Estas tres preguntas están relacionadas con la interpretabilidad de MOMENT, asumiendo que son rápidos de usar al tratarse de modelos fundacionales que puedes usar de manera directa sobre conjuntos de datos nuevos sin necesidad de volverlos a entrenar. Los experimentos muestran que MOMENT no alcanza la buena interpretabilidad de MTSAE en ninguna de las tareas donde destaca MTSAE, tampoco en la detección de tendencias. Esto resulta en una respuesta negativa a **RQ2.1**.

El error obtenido al ajustar el modelo se mejoró enormemente (con porcentajes de hasta el 20% - véase la figura 4.12), pero los espacios latentes continuaban sin ser fáciles de interpretar en ninguna de sus configuraciones. En consecuencia, no se puede concluir que las mejoras en términos de error estén directamente relacionadas con la interpretabilidad del espacio latente, ya que parecen ser independientes (**RQ2.2**). Del mismo modo, dado que el ajuste fino realizado con distintos tamaños del modelo no arrojó diferencias a nivel visual, no se puede dar respuesta a la pregunta **RQ2.3**.

La ausencia de gráficos interpretables da la impresión de que no resulta de provecho introducir los modelos fundacionales dentro de DeepVATS como una herramienta visual. Sin embargo, los resultados son muy extraños al compararlos con el estado del arte. Además, no dependen del tamaño de los conjuntos de datos utilizados para el entrenamiento, pero no hemos intentado

entrenar el modelo con la serie completa o resetear su configuración de pesos al estado inicial para evitar que esté sesgado con entrenamientos previos. En consecuencia, parece oportuno seguir avanzando en esta línea, con las líneas futuras que se mencionarán en la sección 4.2.1.

## Conclusiones

DeepVATS es una herramienta potente diseñada para el análisis visual sencillo de series temporales tanto univariables como multivariables. La herramienta permite a los usuarios interactuar con la proyección de espacios latentes y el gráfico de los datos reales de la serie temporal, facilitando la detección de patrones en los datos. El análisis de escalabilidad realizado en la sección 3.4 ha demostrado que la aplicación tiene una eficacia excelente para conjuntos de datos pequeños (desde 49.3K hasta 98.6K elementos) y conjuntos de datos de tamaño elementos (hasta 493.1K elementos). Sin embargo, aparecieron problemas de eficiencia al procesar series temporales largas, con una degradación visible a partir de los 3.7M elementos y no es capaz de ejecutar a partir de los 7.4M elementos.

Para mejorar la usabilidad de la aplicación con conjuntos de datos grandes, se proponen tres líneas de investigación. La primera, eliminar procesos redundantes: las variables de tipo `reactive` deberían revisarse para determinar si pueden ser convertidas a `reactiveVal` para asegurar el uso de valores guardados en caché. Esta línea se ha llevado ya a cabo, facilitando el análisis posterior en la investigación presente. La segunda línea, enfocada en asegurar reducciones de dimensionalidad de alta calidad, se sugieren dos estrategias: adoptar una implementación alternativa de UMAP [2], y explorar la aplicación de PCA seguido de UMAP en lugar de usar únicamente UMAP [3]. En nuestro caso, se selecciona el uso de PCA seguido de UMAP, permitiendo un análisis mucho más adecuado ya que se mejoró enormemente la estabilidad y las operaciones de zoom para el análisis local y global de espacios latentes.

Las modificaciones mencionadas suponen una mejora sustancial de la eficiencia de la aplicación (e.g., eliminando un total de 28 segundos que eran requeridos para el cómputo de las proyecciones en el caso del conjunto de datos `Solar 4 Seconds` y aportan estabilidad en la proyección del espacio latente cuando se usa la GPU, haciendo que DeepVATS sea una herramienta más robusta para el análisis virtual e interactivo de series temporales.

Tras la inclusión de las modificaciones mencionadas, la investigación continúa con la integración de MPlot en la aplicación. Esta integración, usando una versión más eficiente y flexible de MPlot (incluyendo la función específica para medir tendencias), mejora DeepVATS en el análisis de series univariables. Así mismo, permite al usuario tener una aplicación interactiva sencilla que permite la interacción con el MPlot para visualizar su relación con los datos de la serie temporal y el MatrixProfile (variable de distancias relacionada con el MPlot). También aporta una funcionalidad potente nueva para un primer análisis visual del comportamiento de la serie temporal, resultando en un gráfico obtenido en menos de diez segundos (véase la tabla 4.1), suponiendo una mejora en la interactividad de DeepVATS. Aunque MPlot no se pueda usar para series temporales multivariables, el MatrixProfile tiene una definición análoga para el caso. La adición de esta herramienta en la serie temporal como una variable exógena para el entrenamiento de los modelos podría resultar interesante para obtener un análisis conciso. Además, los modelos fundacionales nuevos [1, 4, 5] podrían resultar de utilidad para la detección de tendencias, mejorando el análisis con DeepVATS. Así, para mejorar el desarrollo de las funcionalidades de la herramienta, estos modelos son incluidos en el modelo de DL de la misma. Se esperaba que esta integración permitiera la inclusión de nuevos modelos centrales en DeepVATS que no requieren de un entrenamiento largo, permitiendo un análisis más rápido que mantenga la efectividad de

la aplicación. Sin embargo, tras comprobar las proyecciones del espacio latente usando la técnica de ejecutar PCA seguido de UMAP, éstas no resultaron sencillas de interpretar. Tras comprobar su ejecución con un ajuste fino similar a la técnica de entrenamiento de MTSAE, la investigación concluye que una mejora en términos cuantitativos de errores no está directamente relacionada con la interpretabilidad del espacio de proyecciones, dificultando su mejora por este camino. Aún así, el problema puede estar relacionado con que la inclusión se ha realizado en la manera más naive, por lo que resulta necesario proponer perspectivas distintas a las propuestas: modificando las funciones de distancia que calculan los errores y evaluando su impacto en la interpretabilidad del PP, el uso de otras técnicas de proyección para asegurar la independencia del camino en el que los vectores son proyecciones, preprocesado de los datos para llevar a cabo tareas más específicas, y mejorando el proceso de ajuste fino mediante la congelación de algunas capas del modelo y la comprobación de los gráficos de proyecciones finales (incluso en detrimento de la mejora del error).

La integración de esas técnicas y las nuevas herramientas de analíticas visuales introducidas en el Capítulo 5 supone una buena manera de juntar las áreas de DL y DM para mejorar el análisis de series temporales. La sección siguiente resume las líneas de investigación futuras detectadas para la mejora de la interpretabilidad y eficiencia de las herramientas de analíticas visuales basadas en DL.

## Trabajo futuro

Una vez completada la disertación, se han analizado y mejorado las capacidades y limitaciones de las herramientas de análisis visual (particularmente, DeepVATS, para el análisis de series temporales), identificando distintas líneas futuras de investigación (LF) que podrían extender el trabajo presente:

- **LF1. Usando variables derivadas del MatrixProfile** como variables exógenas podría resultar un mejor análisis al tener más información basada en distancia en las entradas conjuntas del modelo.
- **LF2. La integración de modelos fundacionales** de series temporales más nuevos en el módulo de DL. Este camino se inició con la adición de MOMENT en la aplicación DeepVATS, pero se pueden añadir más modelos fundacionales multitarea que puedan aparecer o, incluso, cualquiera de los modelos fundacionales mencionados en la sección 3.6.
- Mejorando el proceso de ajuste fino:
  - **LF3. Modificar la distancia usada para medir el error (LF3).** Modificar la distancia utilizada al calcular el error no debería tener implicaciones evidentes en términos del ajuste del modelo a la serie temporal. Sin embargo, ¿cambia la modificación del espacio latente? ¿La selección de una distancia más global resulta en una mejor topología del espacio latente? En ese caso, sería una buena opción el uso de modelos fundacionales con entrenamientos pequeños y una distancia específica para cada task. Esta línea ha mostrado ser prometedora tras una primera prueba utilizando el soft-DTW [6] en la sección 4.2.8.
  - **LF4. Aplicar preprocesado de datos**. Esta opción debería ayudar en cualquier caso, ya que siempre es una buena idea preprocesar los datos antes de usarlos en cualquier tarea. Sin embargo, el objetivo no es solo el preprocesado de los datos para su análisis posterior sino que también hacer copias preprocesadas de los datos para el ajuste

fino y del modelo y usar posteriormente los datos originales para comprobar si el espacio latente resultante de la nueva inferencia tiene una topología más sencilla de interpretar.

  - **LF5. Uso de otras técnicas de proyección (LF5).** La técnica de proyección utilizada en la experimentación es PCA seguida de UMAP para extender los resultados de [7]. Sin embargo, el artículo original de MOMENT [1] usa PCA yt-SNE. Podemos comprobar si el espacio latente es más fácil de interpretar cuando se usan esas técnicas en nuestro entorno de experimentación.
  - **LF6. Congelar las capas del modelo durante el ajuste** puede ayudar en la precisión del espacio latente.. El ajuste fino de modelo se puede realizar con distintas técnicas. Una de las más extendidas es la congelación de parámetros en capas específicas de manera que se mejora el modelo enfocándose únicamente en algunas partes del mismo. Ahora, la pregunta siguiente es: ¿podríamos, incluso con pérdidas mayores, obtener proyecciones del espacio latente más interpretables mediante la congelación de los parámetros de la capa final?

- **LF7. Mejorar el código** (primera solución propuesta en la sección 3.4). Revisando la posibilidad de trocear en etapas el cómputo asociado a la obtención de los vectores latentes de MOMENT. Esta propuesta debería mejorar la eficiencia en el análisis de conjuntos de datos largos.
- **FL8. Nuevas técnicas de análisis visual de otras áreas.** Como se menciona en la disertación, el uso de herramientas de analítica visual está creciendo, impulsado por la necesidad de hacer los procesos más explicables o interpretables. Cada herramienta adopta sus propias técnicas. Realizar una revisión exhaustiva del estado del arte (que abarque no sólo las técnicas actuales para el análisis de series temporales, sino también aquellas aplicadas en otros modelos de machine learning y en el análisis del espacio latente) podría resultar muy enriquecedor. Este enfoque ayudaría a ampliar las capacidades de DeepVATS, integrando distintas perspectivas en una única aplicación de código abierto y facilitando así la comprensión tanto de los datos analizados como del proceso de análisis seguido.
- **FL9. Modificar el modelo MTSAE.** La mejora del análisis de patrones globales en series temporales, como la tendencia, en series temporales puede involucrar la modificación del modelo MTSAE; por ejemplo, añadiendo dilataciones acumulativas a lo largo de las capas para la realización de convoluciones dispersas o la adición de núcleos más grandes. De esta manera, se podrían capturar patrones más largos de la serie temporal.

De manera conjunta, estas direcciones apuntan a un futuro en el que las herramientas de análisis visual se tornarán de manera incremental precisas, eficientes e interpretables, forjando las bases para análisis más de e interactivos en campos que usan herramientas relacionadas con los procesos HOTL y transformando potencialmente prácticas analíticas más amplias.

# Abstract

Time series (TS) analysis plays a crucial role in scientific and industrial applications, facilitating tasks such as anomaly detection, prediction, pattern detection, and segmentation. Despite its significance, traditional Artificial Intelligence-based approaches often struggle with scalability and intepretability. This thesis presents a novel approach to large-scale visual TS analysis joining Data Mining and Deep Learning techniques to enhance the efficience and interpretability of TS exploration.

The study is done within DeepVATS's benchmark, a Deep Visual Analytics tool for time series analysis designed to improve the interpretability of Deep Learning-driven techniques. It is based in the interaction with the latent space of those techniques' backbone models and the original TS data plots. The backbone architecture of the application is Masked Timeseries AutoEncoder, a Deep Learning architecture that captures the insights of the time series in its latent space, allowing a good visual interactive analysis of the TS for the mentioned tasks. However, it lacks the ability to detect trends within TS. Thus, it provides a great basis for analyzing the capabilities of the new integrated tools.

A dual methodology is proposed to enhance the visual analytics tool by integrating Data Mining techniques and the raising Time Series Foundation models. This integrations add interpretability and reduce execution time within the analysis.

The first part of the methodology uses MPlot as a preliminary analysis tool to detect patterns and anomalies, reducing the waiting time required to obtain a prior analysis while training the backbone Deep Learning model on which the application is based. This approach enables a more interactive and resource-efficient analysis of large time series, adding the detection of the trend in univariate time series into DeepVATS, which was a key limitation of the tool.

In the second part of the methodology, the research evaluates the effectiveness of Time Series Foundation models in modeling the properties of the TS and capturing them in the topology of their latent space. By integrating the most influent multivariate and multi-task Time Series Foundation models (MOMENT) into DeepVATS, this study explores how the representations learned by that model can improve both accuracy and efficiency in the proposed tasks.

The experimental results show that the combination of Distance Matrix Plot with DL techniques provides a scalable and interpretable framework for time series analysis. The proposed approach not only accelerates the exploration process but also improves trend detection without requiring extensive manual feature engineering.

The contributions of this thesis have broad implications for multiple domains, including financial analysis, medical diagnostics, or industrial process monitoring. The research prepares the way for the integration of more interactive and interpretable visual TS analysis methodologies.

# Contents

# List of Figures

# List of Tables

# List of acronyms

**ABC** Artificial Bee Colony . . . 24
**AI** Artificial Intelligence . . . 1
**ARIMA** Autoregressive integrated moving average . . . 15
**CNN** Convolutional Neural Network . . . 18
**CPU** Central Processing Unit . . . 43
**CUSUM** cumulative sum chart . . . 31
**DCAE** Deep Convolutional auto-encoder . . . 41
**DL** Deep Learning . . . xiii
**DIKW** Data-Information-Knowledge-Wisdom . . . 1
**DM** Data Mining . . . 2
**DR** dimensionally reduction . . . 61
**DTW** Dynamic Time Warping . . . 18
**DVA** Deep Visual Analytics . . . xxiii
**ECG** Electrocardiogram . . . 56
**EMD** empirical mode decomposition . . . 28
**ETS** Exponential Trend Seasonality . . . 27
**FAIR** Findable, Accessible, Interoperable, Reusable . . . 1
**FL** Future Line . . . 105
**GAN** Generative Adversarial Networks . . . 21
**GPU** Graphics Processing Unit . . . 41
**HITL** human in the loop . . . 8
**HOTL** human on the loop . . . 6
**IoT** Internet of Things . . . 122
**KNN** K-Nearest Neighbors . . . 20
**LASSO** Least Absolute Shrinkage and Selection Operator . . . 24
**LLM** Large Language Model . . . 3
**LOCF** Last Observation Carried Forward . . . 21
**LSTM** Long-Term Short-Term . . . 56
**ML** Machine Learning . . . 13
**MLP** multilayer perceptron . . . 21
**MP** Matrix Profile . . . 18
**MPlot** Distance Matrix Plot . . . xiii
**MCS** Multiple Classification System . . . 25

**MSE** mean squared error . . . xxvi
**MSELoss** mean squared error loss . . . xxvi
**MTSAE** Masked Timeseries AutoEncoder . . . xxvi
**MVP** Masked Value Prediction
**NaN** Not a Number . . . 20
**NN** Neural Network . . . 67
**NOCB** Next Observation Carried Backward . . . 21
**NRT** Near Real Time . . . 22
**PCA** Principal Components Analysis . . . 41
**PELT** pruned exact linear time . . . 31
**PP** Projections Plot . . . xxix
**RQ** research question . . . 4
**RF** Random Forest . . . 24
**RNN** Recurrent Neural Networks . . . 18
**SA** Signal Analysis . . . 13
**SMA** Simple Moving Average . . . 97
**SOM** Self-Organising Map . . . 37
**STL** Seasonal and Trend decomposition using LOESS . . . 30
**SVM** Support Vector Machine . . . 17
**TDA** Topological Data Analysis . . . 37
**TFT** Temporal Fusion Transformers . . . 27
**TS** Time Series . . . xix
**TSFM** Time Series Foundation models . . . 3
**t-SNE** T-distributed Stochastic Neighbor Embedding . . . 3
**TSP** Time Series data Plot
**UMAP** Uniform Manifold Approximation and Projection . . . 3
**VA** Visual Analytics . . . 13
**VAR** Vector Autoregression . . . 28
**W&B** Weights and Biases . . . 41
**WWCA** Weighted Water Cycle Algorithm . . . 30
**XAIR** eXplainable Artificial Intelligence ready . . . 1

Chapter 1

# Introduction

*Y tal vez viva ahora mejor*
*más agusto y más tranquilo en mi interior.*

— El Canto Del Loco

Data collection and modeling are essential for understanding, preventing, and improving events and behaviors in multiple domains. This relevance has resulted in extensive research since the 1950s [16], continuing to today with advanced approaches in Artificial Intelligence (AI) [17, 18, 19, 20]. For instance, the detection and prevention of rare events, such as anomalies in oil-producing wells, can prevent detrimental financial implications [21]. Modeling also helps optimize resource allocation in natural disasters [22], facilitates behavior analysis areas such as Unnamed Aerial Vehicle driving [23], or Social Network Analysis [24, 25, 26, 27], and aids in optimizing business processes [28], and is applied in hydroinformatics to analyze water resources [29].

This transformation is conceptually structured by the Data-Information-Knowledge-Wisdom (DIKW) [30], a widely used epistemological model that describes how raw data (D) are contextualized as information (I), synthesized as knowledge (K) and applied as wisdom (W). Recent studies integrate this model with Findable, Accessible, Interoperable, Reusable (FAIR) [31] and eXplainable Artificial Intelligence ready (XAIR) [32, 33] principles, providing guidance for data management, documentation and explainability. On the one hand, FAIR facilitates the transition from data to information by ensuring findability, accessibility, interoperability, and reusability through structured metadata and persistent identifiers [31]. On the other hand, XAIR enables the transition from information to knowledge, by introducing epistemic metadata, including knowledge claims, foundation and validity assertions, thus supporting scientific reuse and trust [30]. These principles enrich the data workflow and are essential in building explainable AI systems.

An effective tool in the transition from information to knowledge in the DIKW pyramid is data visualization. As noted by Ramsey et al. [34], visualization plays a central role in building trust in Machine Learning models, especially in domains where decisions must be fast and interpretable. In high-risk scenarios such as crisis response or disaster management, where machine learning models support detection, response, and decision-making [22, 35], interactive visual explanations allow domain experts to explore, question, and validate decision logic, especially in interpretable models like decision trees [34]. These explanations improve understanding and trust through

transparent, socially responsive mechanisms that address the human need for contrastive, selective, and context-aware explanations [36, 37].

The role of visualization is also critical in biomedical images, where explainability is central to the adoption of the different techniques. Nazir et al. [38] highlight that deep neural networks, despite their increased predictive power, often lack interpretability. Visualization techniques such as Grad-CAM and processing the results to build a heat map provide post hoc explanations that can bridge the gap between a model's decision and human understanding. This solution was proposed by Liz-López et al. [39] in the context of diagnosing pediatric bacterial pneumonia using X-ray images, demonstrating how visual interpretations can support clinical validation and improve clinician performance.

The future of AI development depends not only on the accuracy of the modeling but also on the clarity of its processes. As Liu et al. [40] emphasizes, AI visualization is emerging as a discipline that extends across the entire lifecycle of Machine Learning, from data collection and model debugging to deployment and explanation. Interactive visualization, in particular, allows practitioners to consult, explore, and validate models dinamically, which reinforces the DIKW flow and enhances the explainability as a human-centered collaborative process.

Based on the principles outlined above, this thesis focuses on the area of time series analysis, where DL and Data Mining (DM) techniques have demonstrated their value in modeling complex temporal behaviors. Although many methods already incorporate mechanisms to improve interpretability, there is still considerable potential to strengthen the connection between model outputs and human reasoning. In particular, the incorporation of interactive visual tools offers new opportunities to enrich our understanding of temporal patterns, support decision-making, and drive greater transparency in model behavior. The following sections explore this integration in depth and examine how visual explanations can be enhanced as a complement for DL and DM algorithms to provide more intuitive and practical insights in the time series context.

## 1.1 Context and Motivation

Time Series data are omnipresent in domains like finance [41, 42, 43, 44] or health [45, 46, 47, 48, 49, 50, 51, 52], where tracking changes over time is fundamental for understanding and anticipating real-world phenomena. Their temporal structure introduces specific challenges, such as non-stationarity or feature dependencies, that make their analysis both complex and essential for informed decision making.

A wide range of tasks have been developed to extract insight from TS data, including anomaly detection [18, 19, 20, 17], classification [53, 54], pattern detection [55] and forecasting [56]. These tasks serve various practical purposes. Anomaly detection is used to detect rare events related to health problems [57], intrusion detection in connected networks [58] or hard-disk failure detection [59]. Classification helps in autonomous transportation decision making [60], space object detection and classification [61], and biomedical signals analysis [62]. Pattern detection supports power grid event diagnosis [63] and recognizing behavioral patterns [64]. Forecasting enables prediction of the orbit of space objects [65, 61], photovoltaic power [66], or air quality [67]. These tasks are performed across various domains, each employing distinct methodological perspectives. Schmidl et al. [8] present the following classification of the techniques depending on their area of study: classic ML [68, 69, 70], Signal Analysis [71, 72, 73, 74], Stochastic Learning [69, 75, 76, 77, 78], Statistics [79, 80, 81], DL [82, 83], DM [84, 85] or specific Outlier Detection

techniques [8].

Focusing on the Deep Learning area, models embedd time series into high-dimensional latent spaces, where each sequence is mapped to a vector capturing abstract temporal features "learned" by the model. These embedding spaces are model-specific and multidimensional, making them difficult to interpret. To unveil their structure, dimensionality reduction techniques such as Uniform Manifold Approximation and Projection (UMAP) [86] or T-distributed Stochastic Neighbor Embedding (t-SNE) are commonly used to project embeddings into 2 or 3 dimensions for visual inspection. This allows the analyst to explore clusters, transitions, and outliers in a way that would otherwise be inaccessible. To improve interactive visualization analysis of these embedding spaces, and thus support model interpretability and expert-driven exploration, this work integrates algorithmsfrom DM and DL into a visual analytics tool.

In the field of Deep Learning (DL), significant effort has been made to understand the embedding representation space (latent space) produced by the different algorithms. There are many tools for interactive visualization of the projection of embedding spaces for datasets analysis [87, 88]. Among these tools, DeepVATS [7]), stands out as an excellent tool to analyze the embedding space of large scale Time Series, combining DL algorithms with visual tools. DeepVATS is based on the training of a DL model for a later analysis of its projection of the embedding space, as its architecture captures the structure of the time series in a vectorized form, giving a more intuitive comprehension of it. This process involves a large consumption of time [89], so the interactivity is reduced as the size of the time series grows. To avoid this waiting time, techniques from the emerging research on Time Series Foundation models (TSFM) can be used to gain advantages. TSFM significantly brings Large Language Model (LLM) techniques to time series analysis greatly reducing the execution time while obtaining the embedding space for specific datasets [83, 82].

In the field of Data Mining (DM), distance matrices and their associated methods such as SCAMP [90] or SCRIMP++ [91] and tools such MPlot, which are plots that show the distance between subsequences of the time series, [11] have stood out as a great approach for time series analysis [8]. Although, as will be shown in future sections, the computation of these plots requires a lot of computational resources resulting even in Out Of Memory errors, they have been scaled up to perform large-scale time series data analysis [57]. Thus, they provide an efficient way for fast and resource-efficient analysis of the different time series behaviors such as anomalies, patterns, or novelets. The use of MPlot can give a good overview of time series patterns even when time series are downsampled for reduced resource necessities [92]. Thus, MPlot helps to gain a first view of the time series while waiting for Deep Learning training.

In addition, hybrid methods that combine techniques from both distance-based and DL algorithms, such as the one proposed by [93], have shown the capability to provide better results in the analysis of TS.

The main hypothesis of this thesis is that the limitations of visual interactive analysis for TS can be overcome through the integration of tools from both DL and DM approaches, enhancing the interpretability and interactivity of the associated plots.

## 1.2 Research questions

The challenge outlined in the previous section highlights the necessity of balancing efficiency with interpretability in the field of large-scale interactive visual analysis. This thesis aims to fill these gaps by integrating DM and DL techniques, providing new insights into how time series can be efficiently analyzed and visualized. Figure 1.1 shows the tree of hypotheses and research question (RQ)s, summarizing the information detailed in the present section.

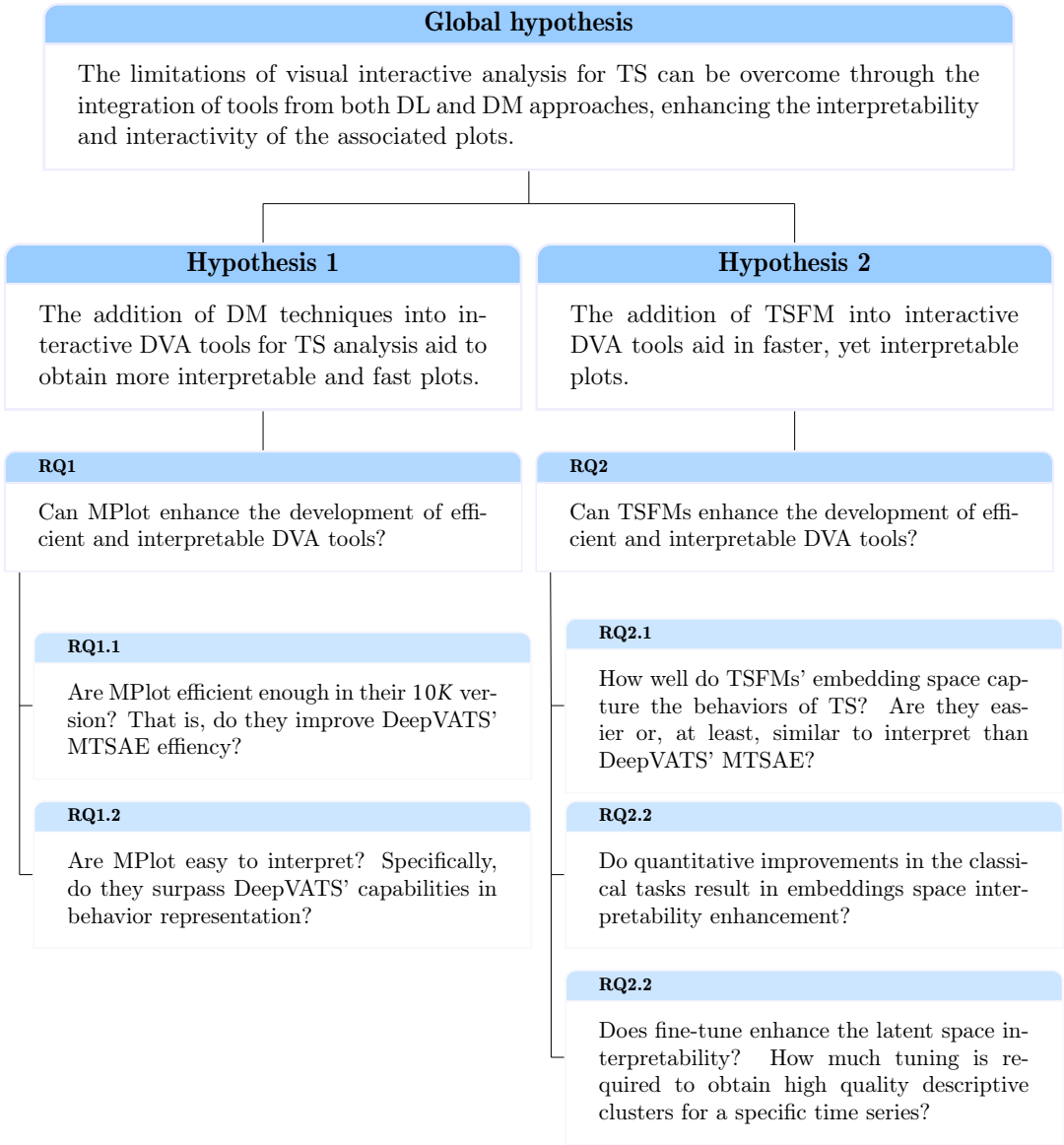



**Figure 1.1:** Research questions: schema showing the global hypothesis of the thesis, its division in two paths and the associated research questions. This diagram serves as a conceptual framework that guides the dual methodological strategy developed in subsequent sections.

The global hypothesis is that the limitations of visual interactive analysis for TS can be overcome through the integration of tools from both DL and DM approaches, enhancing the interpretability and interactivity of the associated plots.This hypothesis is specified in the two lines:

- **H1** addition of DM techniques into interactive DVA tools for TS analysis aid to obtain more interpretable and fast plots.

- **H2** addition of TSFM into interactive DVA tools aid in faster, yet interpretable plots.

To give an answer to the hypothesis this, the thesis focuses on the use of MPlot and TSFMs for fast and resource-efficient TS behavior analysis. The interpretability of MPlot has been proved and its computational algorithms have been adapted to be fast and scalable, including a version scaled down to a maximum of 10K timestamps that compresses long time series information. Similarly, TSFM have been shown to retain time series properties within their latent space and are fast because they do not need to be retained for use in new datasets. Also, they can be fine-tuned to produce more meaningful latent representations.

These approaches raise the following RQs that will be addressed and answered in this dissertation:

1. Can MPlot enhance the development of efficient and interpretable DVA tools?

2. Can TSFMs enhance the development of efficient and interpretable DVA tools?

Focusing on the experimental framework of the thesis, the RQs are refined to the specific questions described below.

The question **RQ1**, which concerns MPlot, is divided into:

1. Are MPlot efficient enough in their $10K$ version? That is, do they improve DeepVATS' MTSAE effiency?

2. Are MPlot easy to interpret? Specifically, do they surpass DeepVATS' capabilities in behavior representation?

The question **RQ2**, related to TSFM, is divided into:

1. How well do TSFMs' embedding space capture the behaviors of TS? Are they easier or, at least, similar to interpret than DeepVATS' MTSAE?

2. Do quantitative improvements in the classical tasks result in embeddings space interpretability enhancement?

3. Does fine-tune enhance the latent space interpretability? How much tuning is required to obtain high quality descriptive clusters for a specific time series?

These questions serve as the foundation of this research, guiding the experimental design and evaluation methodology. By addressing them, this thesis contributes to the development of more interactive, scalable, and interpretable tools for large-scale time series visual analysis.

## 1.3 General Methodology

This section describes the general methodological approach adopted in the thesis. It provides an overview of the foundation of the dual research strategy based on DL and DM, details the tool development process, and explains the evaluation strategy designed to validate the proposed hypotheses. The methodological structure follows an iterative engineering framework and is

aligned with the requirements of human on the loop (HOTL) systems. The methodology steps are summarized in Figure 1.2.

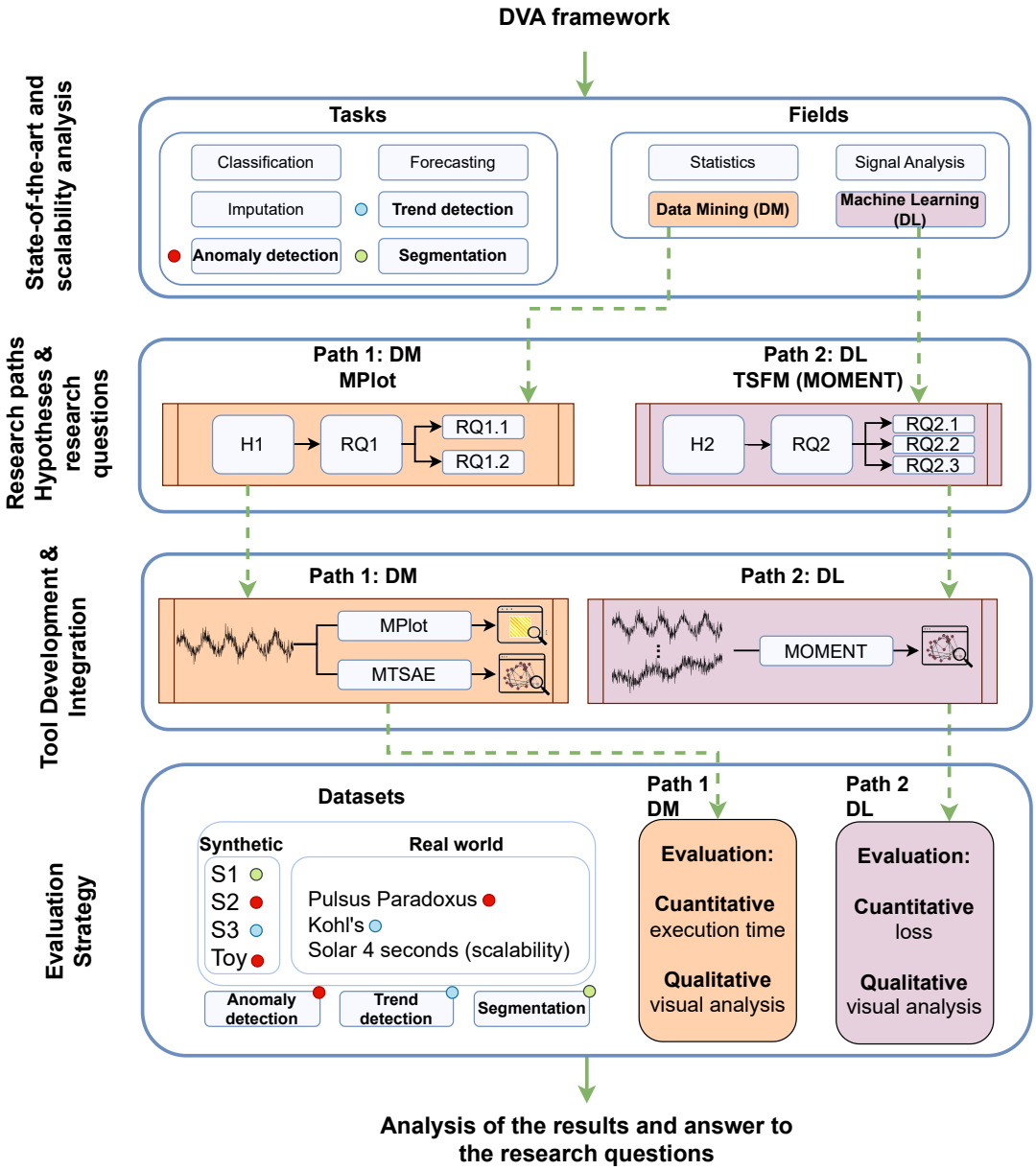



**Figure 1.2:** Main steps within the methodology. The research starts with an scalability analysis of a DVA tool, followed by an exploratory analysis to look for better solutions that yields to a dual path methodology based on the hypotheses and research questions in Section 1.2. These paths are followed by integrating MPlot and MOMENT into DeepVATS as experimentation tool, using the datasets and metrics in Evaluation Strategy.

### Research motivation and methodological orientation

The focus of the present research is to improve the interpretability and usability of large-scale TS analysis after detecting scalability issues in a DVA analysis. The proposed solution integrates DL and DM techniques into a single exploratory environment to help analysts gain faster and

clearer understanding of complex temporal patterns.

The methodological strategy is aligned with the software engineering framework outlined by Adrion et al. [94], which consists of four iterative stages: observe, propose, build, and evaluate. This framework provides the basis for validating the hypotheses (see Section 1.2 and Figure 1.1). These hypotheses define the dual research strategy based on DM and DL, which corresponds respectively to Hypotheses **H1** and **H2**. Each hypothesis is addressed through a specific path, linked to a well-defined set of questions (**RQ1** and **RQ2**) as well as their subcomponents.

**Exploratory analysis and research direction**

To analyze the hypotheses, the research starts with an exploratory review of the state of the art. This analysis revealed new tools that could enhance the architecture of current VA tools for better responsiveness and clarity for large-scale datasets. This led to the adoption of a dual methodological approach aligned with Hypotheses **H1** and **H2**.

**Data Mining - MPlot** This research path explores whether MPlot improves interpretability and computational efficiency in exploratory analysis. It addresses **RQ1** and its subcomponents **RQ1.1** and **RQ1.2**. The research explores the effectiveness of MPlot as a preliminary analysis tool to detect patterns and anomalies in time series before (or while) the application's backbone DL model is trained.

**Deep Learning - TSFM** This second path evaluates the use of TSFM for TSs representation. It corresponds to **RQ2** and its subcomponents **RQ2.1**, **RQ2.2**, and **RQ2.3**, which test the interpretability and training efficiency of TSFM's representations in comparison to the MTSAE baseline.

The decision to combine DL and DM techniques results from the fundamentally different ways in which each paradigm processes and represents information. While Deep Learning excels at capturing high-level abstract patterns through learned representations, it often lacks direct interpretability. In contrast, DM provides clearer and more immediate insight by exploiting the structural properties of the data. The integration of the two approaches allows the methodology to address a broader spectrum of analytical needs.

Deep Learning models, particularly recent TSFM, encode complex temporal behaviors into rich embedding spaces without requiring prior knowledge of the structure of the data. These models offer both scalability and multitask capabilities. Their use of pre-trained representations significantly reduces training costs. However, the abstract nature of these embeddings makes it difficult for users to understand the reasons behind the model's decisions. Addressing this interpretability gap is a key challenge for VA tools used to support interactive expert-guided exploration.

DM methods, in contrast, approach the data more explicitly. Techniques like MPlot calculate direct distances between subsequences to visually show structural patterns and anomalies. These plots offer interpretable visualizations that are computationally lightweight and well suited for fast preliminary analysis. Their efficiency helps users identify meaningful patterns even before a DL model completes its training phase.

This methodology also benefits from the iterative interaction between DL and DM phases during user exploration. For example, early pattern discovery with MPlot helps refine the scope of DL projections by identifying relevant time segments. In the DM phase, once TSFM generates

meaningful embeddings, users can trace specific cluster behaviors back through the original plots, effectively validating or contesting earlier hypotheses. This feedback loop improves the interpretability cycle and the parameter decision.

In terms of usability, the architecture allows analysts to begin interacting with the data through DM tools even before DL modules finish processing. This parallel workflow reduces idle time, leveraging the experience of the analyst to interpret patterns visually.

By integrating and balancing these dual pathways, the methodological proposal supports a more robust and transparent analytic process. Rather than enforcing a strict sequential pipeline, it encourages navigation between Data Mining and Deep Learning strategies based on the task at hand, the experience of the analyst or the available resources. This flexibility aligns well with the real-world demands of modern VA environments.

**Tool development and integration**

The design of DeepVATS's architecture aligns with current paradigms that emphasize the role of humans in AI-driven processes. Traditionally, the human in the loop (HITL) approach involved direct human participation in tasks such as annotation or model supervision [95]. The more recent configuration, HOTL, extends HITL to introduce a supervisory role in which the user monitors the behavior of the model and intervenes based on its performance metrics and visual feedback [96].

The two proposed approaches are developed and integrated into DeepVATS as a unified experimental tool designed for VA. The development phase involves designing, implementing and connecting the MPlot and TSFM modules into the existing system architecture. The TSFM module was implemented in `Python`, while the visual interface was built using `R Shiny`. Integration between these components was achieved through the `reticulate` package, which allows for smooth `Python` execution within `R`. This approach enabled the reuse of fast `feather` data exchange formats while maintaining modularity and reproducibility. While `reticulate` introduces some overhead due to interlanguage communication, the combination proved robust and functionally efficient.

From a software engineering point of view, the development followed the iterative methodology proposed by Adrion [94]. During the "observe" phase, the limitations of existing tools (e.g., high latency, black-box projections) were analyzed. In the "propose" phase, MPlot and MOMENT were selected as candidate tools for integration. The "build" phase included the integration of both tools into the exploration application. Finally, the "evaluate" phase focused on usability and responsiveness from both quantitative and qualitative perspectives.

**Evaluation strategy**

The evaluation strategy is structured to assess whether the integration of DM and DL techniques into the visual analytics tool improves its interpretability and usability for HOTL scenarios. The evaluation process is divided into two complementary perspectives: quantitative and qualitative.

The quantitative evaluation considers different aspects for each research line. For the DM approach (**RQ1**), the analysis examines the time required to generate the plots and the system's responsiveness within the interaction phase. These measurements are used to address **RQ1.1** and aim to determine whether the use of MPlot improves the interactive capabilities of DeepVATS,

particularly in the early stages of visual exploration. For the DL approach (**RQ2**), the evaluation considers the reduction in loss as a metric of how the model captures the behavior of the TS.

The qualitative evaluation focuses on the interpretability of the visualizations produced by each method. For the DM approach, the evaluation analyzes the readability of the plots in terms of their capability to help users recognize patterns, trends, and anomalies within reduced similarity matrix plots. The objective is to support a preliminary analysis of the TS, enabling users to extract early insights and determine the initial parameters to configure the DeepVATS' MTSAE model, all within a faster and more responsive interface (**RQ1.2**). For the DL approach, the evaluation examines whether the embedding space of TSFM models captures TS behaviors and whether their projections are as interpretable as those generated by the MTSAE model (**RQ2.1**).

By combining both perspectives, the analysis explores the correlation between loss reduction in classical tasks and the clarity of the latent space (**RQ2.2**). It also examines the impact of fine-tuning on the generation of descriptive, task-specific clusters, as well as the amount of training required to produce clear and meaningful visualizations (**RQ2.3**).

Together, these evaluation strategies offer a comprehensive analysis of whether the proposed tools truly support the analyst's role within VA tools, focusing not only on performance metrics but also on human-centered interpretability and usability.

In summary, this methodology highlights the advantages of combining DL and DM techniques within a single interactive framework. The capability to alternate between DM and DL-based insights enhances both flexibility and depth of analysis. Fast, interpretable visualizations such as MPlot assist in generating hypotheses and tuning parameters, while DL projections offer access to deeper latent structures. The thesis analyzes how the integration of both paradigms provides a robust foundation for HOTL workflows that neither offer separately.

## 1.4 Publications

The research conducted in this doctoral thesis has resulted in the publication of various scientific contributions, covering both conference proceedings and peer-reviewed journal articles. These publications serve as evidence of the progression of the work and its relevance within the scientific community. This section gathers these publications, highlighting the main findings and advances achieved within the research process.

### 1.4.1 Journals

This subsection highlights the relevance of journal articles by presenting their respective Journal Impact Factor (JIF) indicators, including the JIF rank, as they appear in the Journal Citation Reports (JCR) for the corresponding year. These metrics provide an objective measure of the influence and prestige of journals within the Computer Science field, thus underscoring the quality and impact of the research disseminated through these publications.

**J1:** Inmaculada Santamaria-Valenzuela, Victor Rodriguez-Fernandez, and David Camacho (2024). On the integration of large scale time series distance matrices into deep visual analytic tools. Cognitive Computation. ISSN: 1866-9956, eISSN: 1866-9964. Volume 17, article number 29, 2025
DOI: https://doi.org/10.1007/s12559-024-10394-x

JIF= 4.3 (54/197: Computer Science, Artificial Intelligence; 76/310: Neurosciences) (year 2023) **[Q1]**

- **Overall contribution:** The paper explores the integration of distance matrices into DVA tools, specifically DeepVATS, to improve the analysis of large-scale time series. By incorporating MPlot, the tool allows a faster preliminary analysis of time series with more than 7 million elements. the research validates this approach with synthetic and real-world datasets, demonstrating its usefulness in trend, patterns, and anomalies detection.
- **Contribution of the PhD candidate:**
  - First author of the paper.
  - Implementation of MPlot into DeepVATS.
  - Execution of the experiments and analysis of results.
  - Creation of the manuscript, figures, and tables.

**J2:** Decoding Latent Spaces: Assessing the Interpretability of Time Series Foundation Models for Visual Analytics. International Journal of Interactive Multimedia and Artificial Intelligence (IJIMAI), 15th April 2025, to appear.
DOI: `https://doi.org/10.48550/arXiv.2504.20099`
JIF= 3.4 (78/197: Computer Science, Artificial Intelligence; 62/170: Computer Science, interdisciplinary applications) (year 2023) **[Q2]**

- **Overall contribution:** The paper explores the integration of TSFM, specifically the MOMENT family, into DVA tools. By integrating these transformer-based models into DeepVATS, the hypothesis is that the tool will support faster and more flexible analysis of large-scale TS through pre-trained multitask models. The research evaluates the interpretability of the PP within both synthetic and real-world datasets, focusing on segmentation, anomaly detection, and trend detection tasks. The zero-shot models do not appear to have interpretable embeddings, and, although fine-tuning improves the loss metric, they yield a limited improvement in visual interpretability. This work represents —to the best of our knowledge— the first documented integration of TSFM into DVA tools.
  - First author of the paper.
  - Implementation of TSFM into DeepVATS.
  - Execution of the experiments and analysis of results.
  - Creation of the manuscript, figures, and tables.

### 1.4.2 Conference papers

The research presented in this thesis has also been distributed through various international conferences, which serve as platforms for sharing innovative ideas and receiving feedback from the scientific community. This subsection highlights the relevance of these conference papers by providing information on their indices, such as their inclusion in IEEE Xplore, SpringerLink, or

Scopus, and their ranking in the CORE system. These indicators reflect the quality and impact of the conferences at which the research was presented.

**C1:** Inmaculada Santamaria-Valenzuela, Victor Rodriguez-Fernandez, and David Camacho (2023). Exploring Multiple Classification Systems for Online Time Series Anomaly Detection. In Proceedings of 2023 International Conference on Network, Multimedia and Information Technology (NMITCON), Nitte Meenakshi Institute of Technology, sep. 01 - 02, 2023, Bengaluru, India, pp.1-6.
DOI: `https://doi.org/10.1109/NMITCON58196.2023.10276028`

- **Indexed in**:
  - IEEE Xplore (DOI: `10.1109/NMITCON58196.2023`).
- **Overall contribution:** The paper analyses different techniques for online time series anomaly detection, including statistical methods, machine learning models, and hybrid approaches. The paper explores the effectiveness of ensemble models for anomaly detection and discusses the integration of ensemble-based architectures intoDeepVATS as a visual exploration tool. the research provides a survey includes different methods and their applicability to real-world anomaly detection tasks.
- **Contribution of the PhD candidate:**
  - First author of the article.
  - Analysis of the state-of-the-art.
  - Creation of the manuscript, figures, and tables.

**C2:** Inmaculada Santamaria-Valenzuela, Victor Rodriguez-Fernandez, and David Camacho (2023). In: Nguyen, N.T., Huynh, CP., Nguyen, T.T., Le-Khac, NA., Nguyen, QV. (eds) The 13th Conference on Information Technology and Its Applications. CITA 2024. Lecture Notes in Networks and Systems, vol 882. Springer, Cham. July 19-20, 2024.
DOI: `https://doi.org/10.1007/978-3-031-74127-2_21`
CORE-C (CORE2023)

- **Indexed in**:
  - Springer Nature: `https://link.springer.com/chapter/10.1007/978-3-031-74127-2_21`
  - Scopus: `https://www.scopus.com/record/display.uri?eid=2-s2.0-85209567731&origin=inward&txGid`
- **Overall contribution:** This paper presents a scalability analysis of the DeepVATS framework, a Deep Learning-based tool for large time series analysis. the research evaluates performance using the Monash benchmark, particularly the "Solar Power dataset", and explores the impact of the size of the dataset on the execution time by resampling it at different frequencies. The analysis highlights bottlenecks related to UMAP, `Shiny` reactivity, and `R-Python` communication and proposes solutions to improve computational efficiency.

- **Contribution of the PhD candidate:**
  - First author of the paper.
  - Development and execution of scalability experiments.
  - Identification of computational inefficiencies and proposal of optimizations.
  - Creation of the manuscript, figures, and tables.

**C3:** Inmaculada Santamaria-Valenzuela, Víctor Rodriguez-Fernandez, Jong Hyuk Park and David Camacho (2024). Scalability analysis of DeepVATS: A Framework for DL Visual Analytics on Large Time Series Data. In the 2024 World Congress on Information Technology Applications and Services (World IT Congress 2024). February 14-16, 2024, Jeju, Korea. Pp. 1-7. Web: `http://www.worlditcongress.org/2024/`

- **Overall contribution:** This paper introduces DeepVATS, a tool that combines DL and Visual Analytics for analyzing large Time Series data. It details the architecture, execution pipeline, and the key advantages of DeepVATS over existing tools. Additionally, the paper explores future research directions, such as enhancing scalability and incorporating additional explainability techniques.
- **Contribution of the PhD candidate:**
  - First author of the article.
  - Analysis of DeepVATS and its possible future research directions.
  - Creation of the manuscript, figures, and tables.

**C4:** Inmaculada Santamaria-Valenzuela, Victor Rodriguez-Fernandez, and David Camacho (2024). DeepVATS: A Tool for DL Visual Analytics on Large Time Series Data. In the XX Conference of the Spanish Association for Artificial Intelligence (CAEPIA 2024). ISBN 978-84-09-62724-0, pp. 235-236 (Short Paper).

- **Overall contribution:** This short paper presents DeepVATS, a tool that integrates Deep Learning and Visual Analytics for large time series analysis. It describes the architecture, its execution pipeline, and the key advantages of DeepVATS over the existing tools. The paper also discusses future research directions, including scalability improvements and the addition of other explainability techniques.
- **Contribution of the PhD candidate:**
  - First author of the article.
  - Analysis of DeepVATS and its possible future research directions.
  - Creation of the manuscript, figures, and tables.

## 1.5 Thesis structure

To provide a clear and cohesive picture of the research process, this thesis is divided into five chapters. Each chapter is designed to systematically develop the theoretical and practical foundations and contributions within the present research. The chapters are described in the following, outlining the progression of the work.

**Chapter 2. State of the Art.** This chapter presents the theoretical background that supports the research. It consists of two main sections:

- **Section 2.1: Time Series and associated Machine Learning tasks** formally introduces TS and the main DL tasks associated with their analysis, such as classification, imputation, anomaly detection, forecasting, trend detection and segmentation. It also provides a comparative overview of existing techniques from key fields: Machine Learning (ML), DM, Statistics, and Signal Analysis (SA).

- **Section 2.2: Deep Visual Analytics tools for time series analysis** explores the use of Visual Analytics (VA) tools for time series analysis, focusing on interactive visualization techniques based on DL approaches and their applications.

**Chapter 3. Materials and Methods.** This chapter describes the methodology carried out to achieve the research goals, including the specific tools and frameworks used throughout the thesis. It contains a total of six sections:

- **Section 3.1** introduces the DeepVATS tool, which serves as an experimental environment to test the proposed methodologies.

- **Section 3.2** details the datasets selected for experimentation, chosen to evaluate the ability of the models to retain crucial information for pattern recognition, trend detection, and anomaly detection.

- **Section 3.3** presents the MTSAE architecture, which acts as the DL backbone of DeepVATS and serves as the baseline model for experimentation.

- **Section 3.4** discusses a scalability analysis performed to evaluate the performance of DeepVATS-MTSAE with large-scale datasets.

- **Section 3.5** introduces the MPlot, a technique with high scalability and strong capabilities for visual analytics of large time series, particularly in segmentation, classification, and anomaly detection.

- **Section 3.6** introduces foundation models and Transformers, discussing their adaptation to time series to enable zero-shot or few-shot learning. This section explores their potential as a significant enhancement for visual interactive analytics tools.

**Chapter 4. Results.** This chapter demonstrates how the integration of novel tools improves the interpretability and interactivity of DeepVATS. It has two sections:

- **Section 4.1** evaluates the performance of the MP technique.

- **Section 4.2** evaluates MOMENT, a multitask foundation model developed and integrated as part of this work.

**Chapter 5. Discussion.** Divided into four sections, this chapter critically reflects on the findings and methodology:

- **Section 5.1** highlights the most relevant findings, focusing on answering the RQs.
- **Section 5.2** presents a critical review of the methodology, including the integration of the different tools and the identification of limitations.
- **Section 5.3** analyzes the contributions of the work to the research community and potential real-world applications.

**Chapter 6. Conclusions.** Finally, the thesis concludes by summarizing the main contributions and findings obtained throughout the work. **Section 6.1** proposes future directions for improving the interpretability of embedding spaces in visual analytics tools.

Chapter 2

# State of the art

*You're my guardian angel*
*Hiding in the woods*
*What is your sound?*

— Ylvis

As described in Chapter 1 Introduction, TSs are a key tool to model temporal data across various domains. Their ability to capture sequential dependencies over time makes them particularly suited for modeling dynamic phenomena and supporting critical decision-making processes. For instance, the detection and prevention of rare events, such as anomalies in oil-producing wells, can prevent detrimental financial implications [21]. Modeling can also be useful for optimizing resources in natural disasters [22] and in behavior analysis [23, 27, 24, 25, 26, 97]. TSs one of the main tools used for the modeling and analysis of the data [16, 18, 19, 20, 17] as they represent an easy way to relate events to specific times and are useful to predict future events which can be helpful, for example, for predicting orbits in space objects [65].

Given their versatility, TSs data can be studied from many different perspectives, each offering unique analytical advantages depending on the objective. The diversity of approaches arises from the inherent complexity of time-dependent data and the varied objectives of the analysis. Whether the focus is on discovering latent patterns, detecting anomalies, forecasting values, or improving interpretability, different research areas have developed methods to address these challenges.

These approaches range from traditional Statistical techniques such as Autoregressive integrated moving average (ARIMA) [98] or exponential smoothing models [99], and signal processing techniques to modern Deep Learning models (e.g. MOMENT family [1]), as well as classic Machine Learning, Stochastic Learning, hybrid methods that integrate both statistical techniques and machine learning methods [66] or specific Outlier Detection techniques [8]. Fig. 2.1 ilustrates the broadness of anomaly detection techniques available for TS data, which underscore the interdisciplinary nature of the field.

These research areas do not focus only on anomaly detection; they also cover a broad range of tasks, including pattern detection, segmentation, or trend detection. Based on the clasification

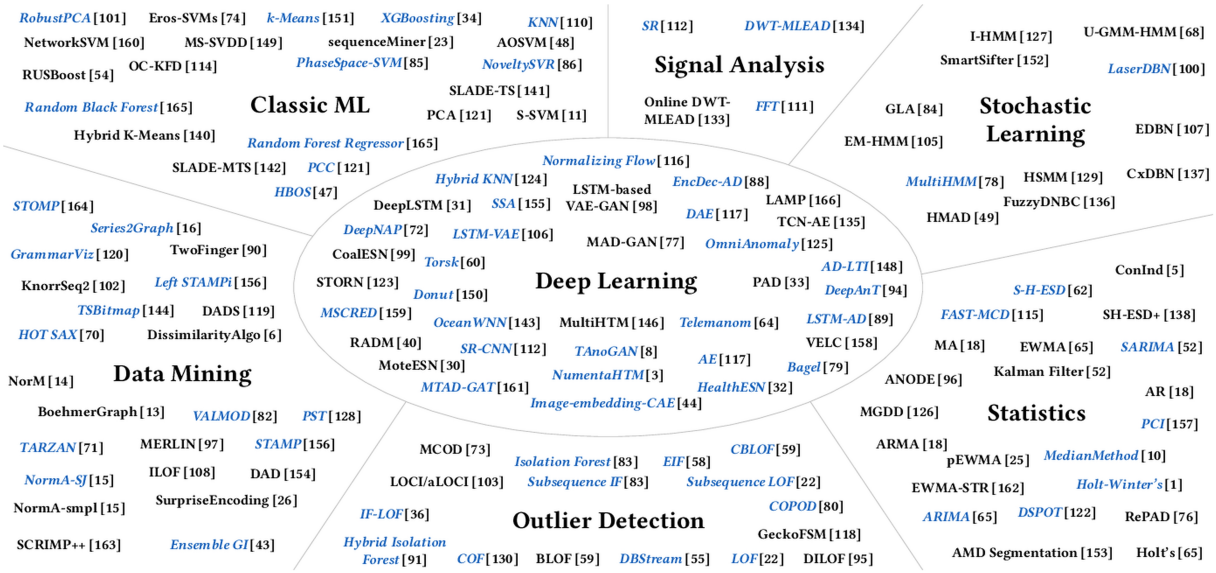


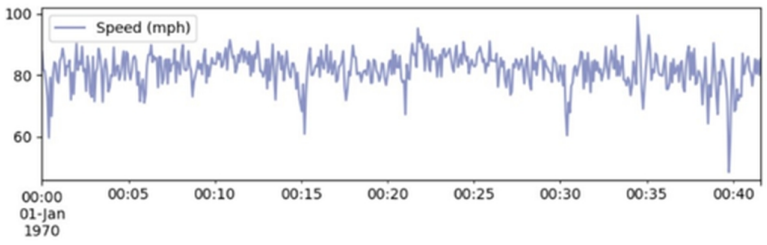

**Figure 2.1:** 158 anomaly detection methods for TS data structured by their area of study collected by Schmidl et al. [8].

proposed by Schmidl et al. [8], and grouping related domains (DL, Stochastic learning, and classic ML are considered subfields of the broader ML category). Section 2.1 formally introduces TSs concept and presents an overview of the main tasks related to TSs analysis and how these tasks are overtaken by the main related research domains: statistics, signal analysis, Machine Learning, and Data Mining. Each of these fields contributes with specific methodologies and perspectives, offering complementary strengths in modeling, interpreting, and extracting insights from time-dependent data.

Once the main analytical tasks have been introduced, the chapter turns its focus in Section 2.2 to the visual tools and techniques developed to support the interpretability and exploration of TS data. In particular, recent advances in VA tools have shown significant promise in bridging the gap between complex model outputs and human understanding. By enabling the representation of latent structures and temporal dynamics through visual plots, these tools give a more transparent and interactive approach to TSs analysis. The integration of dimensionality reduction, interactive interfaces and image-based encodings makes it possible to transform embedded TSs data into interpretable visual plots, supporting both expert-driven insights and algorithm validation.

## 2.1 Time Series and associated Machine Learning tasks

A **TS** is a sequence of real-valued observations $x_t$ indexed by time $t$ (see Def. 2.1.1). A TS is called **continuous** if observations are defined for all $t$ in a subset of $\mathbb{R}$, and **discrete** if observations are collected sequentially over time with $t$ belonging to a subset of $\mathbb{N}$. This thesis focuses on discrete TS with equidistant time steps (i.e., regular intervals between observations). Usually, each time index $t$ is associated to a **timestamp** as a label to denote the reading moment for better comprehension of the data. When data is analyzed while data is obtained, the TS (whether its analysis) is said to be **online** (which is the usual case in IoT). Fig. 2.2 shows an example of a discrete TS with non-regular time intervals.




| 1 | timestamp | value |
|---|---|---|
| 2 | 2015-08-31 18:22:00 | 90 |
| 3 | 2015-08-31 18:32:00 | 80 |
| 4 | 2015-08-31 18:57:00 | 84 |
| 5 | 2015-08-31 19:07:00 | 94 |

**Figure 2.2:** Real-time traffic speed data obtained from `speed_6005.csv` of Numenta Anomaly Benchmark [9]. The data correspond to the Twin Cities metro area in Minnesota and was collected by the Minnesota Department of Transportation. The data has been rescaled by computing the mean in five-minute batches.

**Definition 2.1.1: Time Series and subsequences**

A **time series** $X^n = \{x_t\}_{t=0}^{n} = \{x_0, x_1, \ldots, x_n\} \in \mathbb{R}^{d \times n}$ is a sequence of real-valued numbers, $x_* \subset \mathbb{R}^d$. A **subsequence** is an ordered subset of $X$: $X^{(k,m)} = \{x_t\}_{i=k}^{k+m-1}$, following the same order than the original TS. The values $n, m$ are called the length of $X$ and $X^{(k,m)}$ respectively. When $d = 1$, $X$ is said to be **univariate**, otherwise, **multivariate**.

TSs analysis allows for the extraction of information from data, such as trends or pattern, which can be attributed to dependency relationships between observations. As mentioned, there are different research areas that involve TS analysis: classic ML, Data Mining, SA, Syhocastic Learning, DL, Outlier Detection and statistics. Although the research focuses on the DM and DL fields, this section introduces all the areas, grouping them in their larger groups: Statistics, SA, ML and DM.

**Statistics.** Focuses on modeling the underlying structure of TSs data using probabilistic and statistical techniques. The goal is to approximate the TSs data distribution as a collection of statistical variates and analyze both temporal and intervariate correlations. Common approaches include autoregressive models like ARIMA [98], exponential smoothing models [99], decompsition techniques, and state-space models like Kalman filters [100] or Hidden Markov Models [25], which has been used to analyze the behavior of online game players.

**Signal Analysis.** Focuses on the decomposition, transformation, and analysis of TS as signals, typically using methods such as Fourier transforms [101] or wavelets [102]. It is commonly applied in engineering and biomedical domains, especially for denoising [103], feature extraction [104] or trend isolation [105]. For example, wavelet transforms are commonly used to classify clinical signals [45], and spectral analysis is used to detect anomalies through frequency domain shifts [106]. Signal-based methods often overlap with ML as their techniques are used in conjunction, specially in preprocessing pipelines or when features are obtained from raw signals.

**Machine Learning.** Utilizes different models, whether supervised or unsupervised, to automatically model TS and extract meaningful patterns for analysis. This group includes classic ML and DL. Classical ML methods typically rely on predefined feature extraction steps followed by a lightweight model such as Support Vector Machine (SVM) or decision trees. DL models,

such as Recurrent Neural Networks (RNN) or Convolutional Neural Network (CNN), are larger, with multiple execution layers used to automatically learn feature representation directly from the raw data. Also, inside Machine Learning, within Machine Learning, Stochastic Learning is recognized as a branch that uses stochastic processes to model TS where randomness plays a role. As an example, Jung and Oh [107] uses a stochastic position-based disturbance model to avoid periodic disturbances in the control performance of motion systems.

**Data Mining.** Involves working directly with raw TSs data to extract patterns or subyacent behaviors without requiring explicit modeling. Techniques often rely on distance measures (e.g, Euclidean or Dynamic Time Warping (DTW) [108]) to identify similarities between subsequences. From these computations, higher-level structures like motifs or anomalies can be discovered. A notable example is the Matrix Profile (MP) framework [84, 109], which enables scalable motif discovery, anomaly detection, and segmentation by computing allpairs distances eficient computation.

Although anomaly detection is sometimes considered a separate research area (see Fig. 2.1, in this dissertation, it is treated as a task to avoid conceptual overlap between tasks and methodological domains.

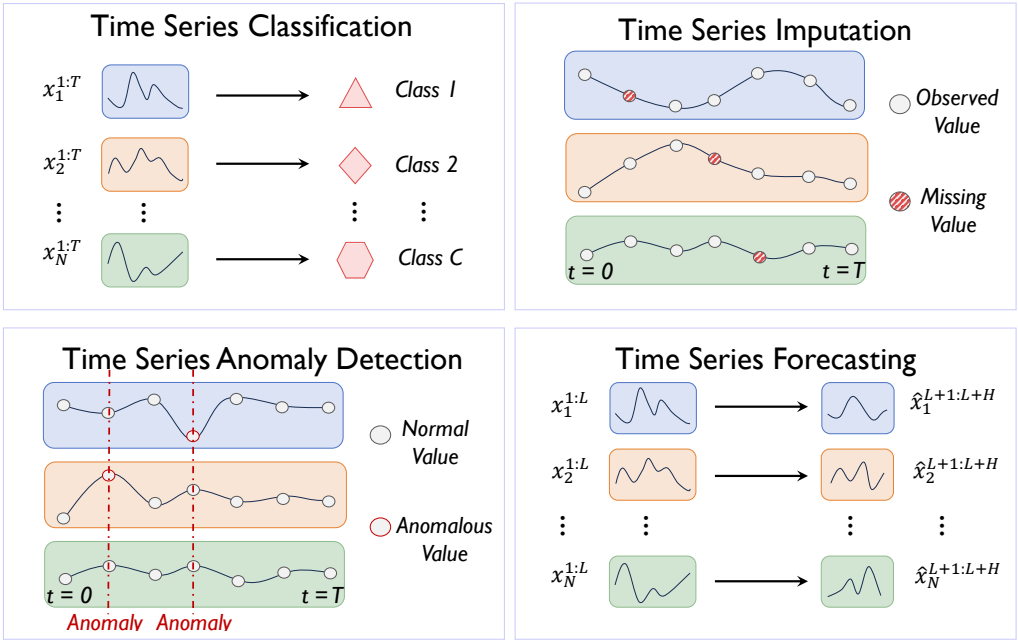



**Figure 2.3:** Summary of the main TS analysis tasks. Figure modified from [10] (Fig. 3)

The main tasks in TSs analysis include classification, forecasting, imputation, and anomaly detection (see Fig. 2.3). In addition, this work incorporates two relevant tasks: trend detection (or trend analysis) and segmentation. This choice is motivated by the fact that, although often considered preprocessing steps, these are essential tasks and, in some cases, the final goal [110, 111, 112, 79]. In particular, the evaluation setting in this work includes scenarios in which the achievement of these tasks is the main goal. The following sections describe each of these tasks in detail, highlighting their specific goals and techniques commonly used to address them within the previously introduced research areas.

### 2.1.1 Time Series classification

The classification tasks consist of the categorization of TSs in different labeled categories based on its temporal patterns and structure (see Def. 2.1.2 and fig 2.3). TSs classification is a challenging problem due to the complexity of time-dependent data. Unlike traditional classification, where instances are independent of one another, TSs data has temporal dependencies that must be preserved for accurate classification. Methods like Dynamic Time Warping (DTW) and more advanced ML techniques are often used to address these challenges. DL, in particular, has gained attention due to its ability to learn complex representations from raw TS data without requiring extensive feature engenieering [113]. The main goals of TSs classification are the improve of accuracy and speed of the classification algorithms, and making these models more interpretable, specially in high-stakes applications such as healthcare or finance. Research continues to focus on developing robust models, including ensemble methods and convolutional architectures, that can generalize between different types of TSs [114].

**Definition 2.1.2: TSs classification**

The goal of the TSs classification task is to define a function $f : \mathbb{R}^{d \times n} \to \mathbb{R}^C \mid X \mapsto f(X) = I_{category}$ with $d > 0$ and $N > 0$ that associates to each TS $X^n \subset R^{d \times n}$ an index $I_{category}$ associated with one of the $C$ categories.

A basic example of classification task is, using `statsmodels.tsa.stattools.adfuller` function that computes the classic Augmented Dickey Fuller test [115] function for defining a function $f$ that gets a TS $X$ and returns a p-value, $p$ and if $p < 0.05$ then it returns "stationary" and "non-stationary" otherwise. Thus, this results in a binary classification for univariate TSs.

**Statistics.** Statistical methods offer robust solutions for TSs classification by modeling temporal dependencies. Between the classicals algorithm is the ARIMA models, which capture autocorrelation and seasonality within the data. In [116], the authors applied ARIMA to classify and forecast the training data of athletes. Another example of a purely statistical approach is the Bayesian TSs Classifier (BTsC) introduced in [117], which decodes visual stimuli from intracranial neural signals using bayesian inference for clasifying signals to decode colors in a visual task.

**Signal Analysis** In the SA field, Wavelet transform [45] technique is widely used for preprocessing signals (cleaning and decompose) and extracting features. Also, the Learnable wavelet transform [118], Meta-transfer metric learning [60] and Kullback-Leiber (KL) divergence [74] techniques have been used for TS classification.

**Machine Learning.** Feature engineering is a widely used technique that involves extracting descriptive characteristics from a TS sample to serve as inputs for traditional ML algorithms [68]. Within the subfield of stochastic learning, approaches such as Motion Code [69] aim to build robust models for TSs classification and forecasting by leveraging Sparse Variational Multi-Stochastic processes. Other models follow a similar objective by employing Gaussian mixtures of reconstructed phase spaces to enhance TSs classification accuracy.

**Data Mining.** Distance-based methods are commonly used within the DM field [46]. They are based on comparing every pair of TS data samples in the dataset based on the distance (e.g. euclidean distance) between them. Classification is performed based on the calculated distances,

where two TSs that are close in terms of distance will belong to the same class. Also, the TSs classification is performed using SVM with Gaussian Elastic Metric kernel [119].

**Hybrid models.** The information obtained from statistical models can be used for decision making and feature extraction to fed ML algorithms. This combination is commonly referred to as statistical learning. One technique within this approach approach is the nonlinear index and thresholding technique, which relies on the definition of indices (either predefined or learned through statistical learning) based on domain-specific and data-driven features. These indices are then used with the threshold values to perform the classifications [62]. Another example of this hybrid approach is the use of highly comparative TSs analysis (htcsa), used to extract thousands of statistical features from TSs for their posterior analysis. As an example, Meynaghizadeh-Zargar et al. [120] uses htca to build attributes from electroencephalogram (EEG) signals, which are subsequently classified with Machine Learning models. Similarly, Irani and Metsis [121] propose a framework that combines bayesian modeling with DL models to obtain a better classification, improving TSs prediction algorithms. Finally, Elen and Avuçlu [93] is an example of a combination of the four areas. It uses SA techniques for feature extraction, Naïve Bayes for probabilistic clasification (statistical modeling), the distance-based DM K-Nearest Neighbors (KNN) method, and classical ML models (SVM and random forest) for supervised learning. Combining all of these techniques results in an effective hybrid classification model.

### 2.1.2 Handling missing data in time series

The imputation task consists of recovering missing data in observations (see Def. 2.1.4 and Fig. 2.3). For example, by taking a linear interpolation between the last known value and the next known value. The focus is on addressing gaps in time-dependent datasets, a usual challenge in fields like economics [122, 123], healthcare [51, 52], and environmental monitoring [124] where the online data collection is sessential. Missing values often arise due to sensor failures, data transmision errors or irregular sampling rates, which can impact the acccuracy of the analyses. The main goal of TSs imputation is to accurately "fill in" these gaps to ensure that models retain temporal continuity and do not lose significant information [125].

**Definition 2.1.3: Mask**

Let $X \in \mathbb{R}^{d \times n}$ be a TS and $M = (m_{ij}) \in \mathbb{R}^{d \times n}$ a "mask" matrix, defined as $m_{i,j} = 1$ if $X(i,j)$ is known, 0 otherwise. Let's denote as $M(X)$ the TS X masked by $M$; that is, $M(X) = \begin{cases} \text{NaN} & \text{if } M(i,j) = 0 \\ X(i,j) & \text{otherwise} \end{cases}$, where Not a Number (NaN) is used to hide the unknown values.

**Definition 2.1.4: TSs imputation**

The imputation task consists on defining a function $f : (\mathbb{R} \cup \{\text{NaN}\}) \to \mathbb{R}^{n \times d}$ that, given a masked matrix $M(X)$, returns the associated X.

The imputation task for TSs includes developing algorithms that are both accurate and computationally efficient, capable of handling various types of missing data patterns, and that can fit irregularly spaced time intervals. The specific techniques of the different areas are presented below.

**Statistics.** Statistical methods, such as Expectation-Maximization (EM) [126], Multiple Imputation, Little's MCAR Test, Selection Models or Maximum Likelihood [127, 128] are key techniques in TSs imputation. These methods traditionally estimate missing values by modeling the distributions and relationships within observed data, providing an effective solution in cases with limited data or when a clear structure is essential for imputations. There exist techniques specific for large datasets applications like the Tall-Project stimator, for factor-based imputation of missing values and covariances in large economic panels [122], and Target-PCA, a stadistic method for stimating a latent factor model using the information from auxiliary panel data sets. Target-PCA tries to apply the idea of transfer learning from ML to statistics to take advantage of previous knowledge in large panels analysis [123].

**Signal Analysis.** SAs for TSs imputation focuses on capturing and preserving continuity in data patterns, which is crucial in fields such as healthcare monitoring and industrial systems. These approaches help to ensure that the temporal dynamics and inherent characteristics of TSs are maintained throughout the imputation. Ziembik and Śródka [129] applies imputation to complete radioisotopes concentration signals for biomonitoring living organisms while Peng et al. [130] introduce a generic frequency domain diffusion algorithm based on regularization for multivariate TSs with missing input data.

**Machine Learning.** In TSs imputation, ML methods, particularly RNN and Generative Adversarial Networks (GAN) [131], have become outstanding due to their effectiveness in handling complex temporal dependencies in missing data. RNN-based approaches, such as GRU-D [70] and BRITS [132], are specifically designed to address temporal decay, preserving long-term dependencies necessary for accurately estimating missing values. GANs are also widely applied, using adversarial training to generate realistic imputations that reflect the distribution of the original series [133].

**Data Mining.** DM techniques for TS imputation focus on identifying and preserving patterns and relationships within incomplete datasets, aiming to ensure that the imputed values remain consistent with the underlying data structure. These approaches can include clustering, pattern recognition, and rule-based systems that help guide imputation by leveraging similar historical patterns or groups within the data. For example, clustering methods can group similar TS segments to predict missing values based on their shared characteristics, while association rule mining can extract frequent patterns in time-stamped data to inform imputation models [134, 135].

In multivariate TSs, data mining methods are particularly useful to understand correlations between variables and to ensure that the imputed values align with these relationships, making them valuable in fields like finance and environmental monitoring, where maintaining variable dependencies is essential [136].

**Hybrid methods.** Kotan and Kırışoğlu [137] propose a hybrid imputation model that combines statistical techniques (mean, median, mode), classical ML (KNN, SVM, random forest) and multilayer perceptron (MLP) (from DL). Another example is the one proposed by Karnati et al. [138], which address missing values in pollution data using a combination of Last Observation Carried Forward (LOCF) and Next Observation Carried Backward (NOCB) with linear interpolation, spectral smoothing (through Fourier's transform [101] and KNN), and KNN imputation). These methods serve as preprocessing before forecasting models based on ARIMA,

Prophet [139], random forest, and LSTM variants, showcasing a complete hybrid pipeline from imputation to prediction.

### 2.1.3 Anomaly detection in time series

Detecting unexpected patterns in data is crucial for many situations, such as the detection of credit card fraud, the monitoring of variations of physiological variables over time for observation of healthcare care, or the proper performance of applications [140]. The problem of identifying patterns in data that deviate from the expected behaviour is known as **anomaly detection**. The formal definition of anomaly detection is given as follows.

**Definition 2.1.5: Anomaly or outlier**

Given a TS $X \in \mathbb{R}^{d \times n}$, an element $x_t$ is said to be an **anomaly** or **outlier** if it deviates significantly from the other data points, for example, if its value is much larger or smaller than the others. It can also represent a change point, for instance, if there is a sudden shift in the trend at that point.
Additionally, **sequence anomalies** may be observed when a subsequence $X^{t,m}$ of fixed length $m$ differs substantially from the rest of the $m$-subsequences (e.g. in terms of mean value, trend, seasonality, amplitude ...).

**Definition 2.1.6: Anomaly detection**

Given a TS $X \in \mathbb{R}^{d \times n}$ and $m \in \mathbb{N}$, $0 < m < n$. **Anomaly detection** aims to define a function $f : \mathbb{R}^{d \times m} \to \{0, 1\}$ that given a subsequence (point if $m = 1$), $X^{t,m}$, returns 1 if $X^{t,m}$ is an anomaly, 0 otherwise.

There exist many occasions where processing information in real time is necessary, such as to analyze daily physical activity using activity monitors [141, 142]. These data are collected in what is known as streaming or on-line TSs. The data of this series is collected and updated (near) in Near Real Time (NRT). Unlike conventional TSs that rely on historical data, online TSs are used to monitor and analyze events as they occur. When data arrive in NRT, the order or arrival time of the data is crucial, as it provides valuable additional information to improve the results. Also, in healthcare, anomaly detection can be used for detecting heart issues like Pulsus Paradoxus [143, 11] that can be detected from electrocardiograms (ECG).

Several methods have been developed to detect anomalies in TS. From statistical approaches such as moving average and standard deviation-based techniques to more advanced algorithms such as autoregressive integrated moving average (ARIMA) [98], exponential smoothing [144], and machine learning-based approaches such as SVM [145], random forests [146], and DL architectures such as recurrent neural networks [147]. These techniques aim to capture temporal dependencies, identify abnormal patterns, and provide valuable insights for the detection of anomalies and subsequent decision-making processes. This section briefly introduces these techniques and their applications.

#### Statistics

The ARIMA method is an statistical TSs analysis technique that combines autoregressive (AR), differencing (I), and moving average (MA) components to model patterns within the data and predict future values. This technique has recently been used to analyze the consumption of fossil

fuels in Slovakia [148]. The authors fit multiple ARIMA models with different combinations of parameters to ensure that the selected ARIMA models explain all the autocorrelation in the series and are adequate for modeling actual TS and predicting future values to get reliable predictions. Another example is the use of ARIMA to predict and control the impact of economic activities related to globalization, industry, and urbanization on human health and the environment. The objective is to better understand the negative effects of these activities and implement appropriate control measures.

As another example of the application of statistics models, the exponential smoothing model is a forecasting technique to predict future values in the TSs data. It is based on the concept that more decent data carry more weight in the prediction than older data. Exponential smoothing is widely used for TSs forecasting. It can be used to improve the estimation of seasonal coefficients in demand forecasting for homogeneous groups of products or series [144]. This technique has been used for different purposes, such as forecasting energy demand [149], prediction of hotel daily room demand [150] or the study of surface water temperature.

#### Signal Analysis

SA in TSs anomaly detection involves techniques that identify deviations in data patterns by examining the underlying frequency, amplitude and waveform characteristics of the series. Methods such as Fourier transforms and wavelet analysis are commonly employed to decompose TSs into components, allowing for the identification of irregularities that may indicate anomalies. These techniques are valuable in applications such as healthcare care, where deviations in physiological signals can indicate potential health problems, or in industrial monitoring, where sudden changes in machine signal patterns can signal impending failures [134, 151].

Another approach within signal analysis is spectral analysis, which focuses on detecting anomalies by identifying shifts in the frequency domain. This technique is especially useful in scenarios with periodic or oscillatory data, such as monitoring of electrical networks, where unexpected changes in frequency patterns can reveal anomalies linked to faults or outages [106, 152, 153].

These methods allow to detect anomalies with high sensitivity and precision, especially in those datasets where anomalies are characterized by subtle, complex deviations in the signal behavior.

#### Machine Learning

ML methods use algorithms to automatically learn and improve data without explicit programming. The goal is to complete tasks and make decisions intelligently and quickly. They can be subdivided into two main types:

- **Supervised learning.** These methods use a labeled data set to train a model that can detect anomalies in future data sets. Examples of such methods include decision trees, neural networks, and k-nearest neighbors. One possible application of this method is the classification of animal behavior state based on environmental characteristics [154].

- **Unsupervised learning.** These methods do not require labeled data and are used to identify anomalous patterns in the data. Examples of such methods include Principal Component Analysis (PCA), clustering methods, and density-based clustering models. These models can be used, for example, for text classification [155] or High-Performance Computer unsupervised anomaly detection [156].

Focusing on Stochastic Learning, the proposal by Su et al. [77] aims to develop a robust model using stochastic RNNs to detect anomalies in multivariate TSs. In addition, Yasami et al. [78] present an approach based on stochastic learning automata to detect intrusions in a computer network (understood here as a digital communication infrastructure connecting devices). In addition, fractional stochastic gradient descent has been used to identify anomalies in smart home networks [157]. Similarly, Dai et al. [158] use a bi-stable stochastic resonance system with adaptive parameter tuning to improve weak signal detection in magnetic anomaly scenarios, showing how noise-driven stochastic dynamics can aid in anomaly recognition under high interference environments.

**Data Mining**

Data mining techniques for anomaly detection in TSs focus on identifying patterns, dependencies, and structures within the data to look for irregular or unexpected behaviours. These methods are usually divided into three groups.

First, clustering [159, 160], is based on grouping similar data points, idenrifying outliers that do not fit into any cluster. For example, in sensor data from insudtrial equipment, clustering can detect machinery faults by grouping "normal" operational states and labeling outliers as potential anomalies, indicating machines failures.

Second, association rule mining [161, 162, 163], identifies frequent groups of items or patterns within the data, with deviations often suggesting anomalies. For example, in retail transactions, association rules help to detect fraudulent transactions by identifying usual item combinations or purchase behaviours that deviate from regular patterns.

Finally, outlier detection [48, 164], focuses on detecting points that differ significantly from the rest of the data, using for example density-based techniques. In network traffic analysis, outliers in data streams can flag potential security threads by identifying unusual packet flows of bandwith spoikes, which could indicate an anomaly in the network performance [165].

**Hybrid methods**

There exist hybrid methods that integrate both statistical techniques and machine learning methods to detect anomalies in TSs data. Mubarak et al. [66] introduces an example joining Random Forest (RF) regressors and a Least Absolute Shrinkage and Selection Operator (LASSO) model. Thus, a combination of the strengths of both approaches is offered, allowing for more comprehensive and accurate TSs modeling and greater adaptability to different situations and missing data.

In addition, Ullah et al. [166] propose a hybrid model that selects the most important features through AlexNet [167] (DL) and substitutes its classification layer with an AdaBoost [168, 169] (classic Machine Learning) to detect normal and theft consumers within a electrical smart grid. The parameters are optimized through the Artificial Bee Colony (ABC) , from bioinspired AI, and classes are balanced through NearMiss (DM).

Anomalies detection is usually done with techniques related to other tasks, usually combining them. Figure 2.4 summarizes the different approaches, from prediction, classification, clustering and their combination in ensemble models.

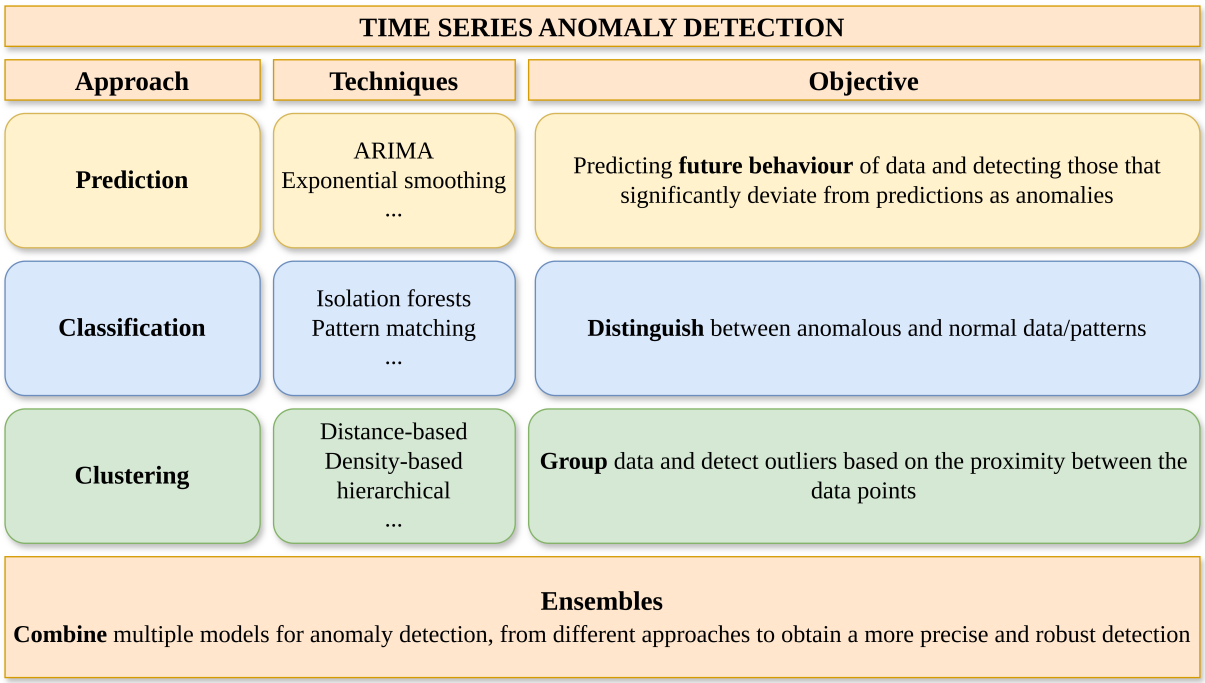



**Figure 2.4:** TSs anomaly detection approaches and techniques.

Finally, Multiple Classification System (MCS) or ensemble [170, 171, 172, 173] is one of the most effective techniques in other areas of machine learning, such as classification [174] or clustering [175]. These systems start with an initial database, which is divided into several subsets to apply a different anomaly detection model to each subset. Subsequently, the results are combined to provide a prediction. Recent studies have shown that MCS is among the most promising research directions for obtaining robust and accurate anomaly detectors [68]. It is a useful technique for enhancing the performance of detectors by reducing the model's dependency on the dataset and complementing the weaknesses of individual detectors while refining their strengths.

An important portion of the data generated by digitalisation, the Internet of Things (IoT), and massive data generation is collected as TSs. Thus, the data are stored as a sequence of data points ordered chronologically based on their acquisition time in NRT. Due to their nature, the data in the series are often correlated over time, making data modelling essential for understanding their behaviour [176]. The detection of anomalies in online TSs poses unique challenges due to their nature. The absence of complete datasets and the requirement for (near) real-time anomaly detection are specific challenges that need to be addressed in models designed for this type of series [173].

The analysis of NRT data faces three main challenges. First, the data set is always incomplete. The correlation between the data points can vary as the data arrive, rendering the previously established model obsolete. This requires incremental training of the model [177], where the model parameters must be adjusted to accommodate the patterns arriving at regular intervals. Secondly, due to continuous updates in the data, each collected data point can only be analyzed once with each state of the database, a concept known as one-pass learning [178]. Lastly, and related to the previous points, there is the issue of non-stationary data distribution, which changes over time. This is known as concept drift, where the underlying concept that the model is trying

to learn fluctuates over time, representing the relationship between the input data and the target variable. The model adaptation in each iteration, combined with the data updates, mitigates the negative impact of concept drift on the accuracy of the model's results. These challenges make anomaly detection in online TSs a greater challenge compared to traditional approaches, and represent a relatively new and emerging research field that has gained increasing interest in recent years [179].

### 2.1.4 TSs forecasting

TSs **forecasting** focuses on the prediction of future values based on historical time-ordered data (see Def. 2.1.7 and Fig. 2.3). This analysis has become instrumental in many fields by enabling data-driven decision-making and resource optimization. The application areas range from energy and environmental sectors to healthcare and finance, where accurate predictions offer benefits in planning and efficiency. Each domain uses TSs forecasting to address different challenges, leveraging models that can capture temporal patterns for both short- and long-term predictions.

**Definition 2.1.7: TSs forecasting**

TSs **forecasting** aims to define a function $f : \mathbb{R}^{d \times L} \rightarrow \mathbb{R}^{d_2 \times H}$ that given $X^{(t,L)}$ returns $Y^{(t+L,H)}$, with $L$ being the length of the look-back window, $H$ the forecast horizon and $Y \sim d_2$ being the TS resulted of selecting $d_2$ dimensions from the sequence $X$. The task can be further divided into long-term/short-term TSs forecasting depending on $H$.

In the energy sector, accurate demand forecasting supports the balance between supply and demand, which is crucial to reducing energy waste and optimizing grid resources. Utilities often rely on forecasting models to predict electricity consumption and renewable energy generation such as solar or wind power, which are inherently variable. By incorporating forecasts, grid operators can integrate renewable resources more effectively, reducing dependence on non-renewable energy, lowering costs, and ensuring reliability in supply [180, 181].

In meteorology and environmental monitoring, forecasting models are essential to aniticipate extreme weather events, monitor air quality, and understand climate trends. For example, prediction of weather patterns, such as wind speed and temperature fluctuations, helps prepare for natural disasters and mitigate risks [22, 182]. This application extends to fields like agriculture, where weather forecasts inform crop management decisions [183, 184], and public health, where pollution forecasts allow communities to take protective measures against poor air quality [185]. It is also useful in agriculture the prediction of prices of the different products [186].

Healthcare applications of TSs forecasting include monitoring the spread of infectious diseases and predicting hospital resource needs. In public health, TSs models provide insights by forecasting trends in disease prevalence, which aids in resource allocation and policy-making. Additionally, in chronic care management, these models support the prediction of disease progression in patients with long-term conditions, improving care by allowing for proactive interventions [49, 50, 187].

Financial and economic forecasting represents another significant area of application, where predicting market behaviors, stock prices, exchange rates, and interest rates is crucial for managing financial risk and capitalizing on investment opportunities. TSs forecasting models are widely employed in this sector due to the substantial impact even slight changes in economic indicators can have on markets. By anticipating trends, investors and policy makers can make

informed decisions that mitigate potential losses while taking advantage of profitable opportunities [81, 188, 189, 190].

In transportation and logistics, forecasting models contribute to optimize traffic flow, reduce congestion, and improve supply chain efficiency. For example, models predict peak traffic times, enabling transportation authorities to adjust public transit schedules accordingly. In logistics, demand forecasting informs inventory management and route planning, leading to cost savings, enhanced customer satisfaction, and minimized environmental impact in optimized energy and fuel use [69, 191, 192, 193, 194, 195].

In marketing and retail, demand forecasting aids companies in anticipating consumer needs, which enables more effective inventory management and targeted marketing. By understanding demand patterns, businesses can maintain optimal stock levels, reduce waste and bnetter align propotional activities with consumer bbehaviour, ultimately increasing profitability and operational efficiency [196, 197, 198].

Although TSs forecasting is usually used for receiving the same number of variates than predicted, this is not always true. There exists models for the four different possibilities:

**Univariate TSs models** like exponential smoothing and its Exponential Trend Seasonality (ETS) variants or the N-BEATS model [199].

**Models that get $d$ variates and return** 1 such as the ARIMAX and ETSX versions [200] that use exogeneous variates, TimeGPT [201] or Prophet [139].

**Get $d$ and predict $d_2$.** The Temporal Fusion Transformers (TFT) model [63], which uses exogenous variates in multivariate TSs. In the oposite way, Chauhan et al. [202] propose an algorithm for predicting diferent number of features than the used for training so it is possible to train with more variates than those predicted in case of signals loss.

**Get $d$ variates and predict $d$ variates** like Vector Autoregresion (VAR) or PatchTST [203].

This algorithms are also present in the previously mentioned research areas, as overviewed in the next paragraphs.

**Statistics.** Statistical models in TSs forecasting rely on mathematical principles and assume a degree of stationarity in the data, making them particularly suitable for structured patterns or trends. Techniques like ARIMA are widely used in agricultural forecasting to predict product prices, helping stakeholders anticipate future supply and demand dynamics [186]. In economic trend analysis, comprehensive statistical reviews of TSs models enable a macro-level understanding of economic factors, allowing for forecasts that aid in long-term economic planning [81]. Also, sparse generalized Yule–Walker estimation for large spatio-temporal autoregressions with an application to $NO_2$ satellite data [187]. For inventory and marketing efficiency, statistical models help optimize stock levels and promotional activities by analyzing past sales data to predict demand, thus minimizing waste and aligning resources effectively [197].

**Signal Analysis.** In TSs forecasting, SA is essential for extracting processable information from dynamic data streams, which are often characterized by considerable noise or variation. However, nowadays it is difficult to find purely SA algorithms for prediction. Most methods model the TSs and then turn to statistical inference or Machine Learning algorithms for prediction. Previous

publications show pure techniques as in the case of Renaud et al. [204], which proposed methods based on wavelets to filter and predict signals.

**Machine Learning.** ML techniques in TSs forecasting focus on capturing complex patterns and dependencies in data that often exhibit non-linearity or high variability. For instance, the BasicTS benchmark, applied in the forecasting of energy demand and renewable generation, uses DL models to handle multivariate TSs with high dimensions, helping energy suppliers optimize grid resources by accurately predicting energy consumption and fluctuations in renewable output [180]. Furthermore, in agricultural and healthcare contexts, Machine Learning models for multivariate forecasting are instrumental in predicting disease progression or crop demand, supporting proactive measures in patient care and supply management [49, 183]. In finance and retail, ML models enable real-time market forecasting, helping firms anticipate stock price trends and consumer demand shifts, which are critical for informed investment and inventory decisions [188, 190, 196]. Another example is PatchTS [203], which has been used to forecast long ocean wave TSs and adapted to predict short-term urban water demand. Also, Stochastic learning approaches are widely applied in TSs forecasting where data are highly uncertain or influenced by randomness. In fields like energy and finance, these models (e.g., day-ahead prediction models and market behavior analysis) predict consumption and price trends based on probabilistic methods, accounting for potential fluctuations and providing information on risk and reliability management [181, 189]. Similarly, in transportation, stochastic models are employed to predict traffic patterns, helping to optimize scheduling and resource allocation under uncertain traffic conditions [193]. These models are especially valuable when forecasting needs to account for abrupt shifts or outliers, such as those seen in energy usage or market prices, providing stakeholders with probability-based forecasts to make risk-averse decisions.

**Data Mining.** Data mining approaches in TSs forecasting are essential in domains where vast and diverse data sources are processed to extract patterns. In environmental and public health monitoring, data mining models analyze large datasets, such as air quality indices, to forecast levels of pollution and help implement community health interventions [182, 184]. In traffic forecasting, data mining is used to predict congestion patterns and inform traffic management strategies, contributing to more efficient and sustainable urban mobility [191, 195]. Similarly, in retail, consumer demand prediction models leverage data mining techniques to anticipate purchase behavior, allowing retailers to align inventory with expected sales patterns [198]. This data-driven approach is invaluable for making real-time adjustments based on detected trends and anomalies, particularly in high-frequency applications like retail and environmental monitoring.

**Hybrid methods.** Recent works explore the development of hybrid models for TSs prediction. Di Mauro et al. [205] integrate multivariate TSs forecasting with Deep Learning and ensemble strategies to predict cellular network quality metrics, using LSTM and CNN combined with statistical models, Vector Autoregression (VAR), to capture inter-feature dependencies and improve forecasting accuracy in real-world environments. Abbasimehr et al. [206] propose a hybrid model specifically designed to handle seasonal and chaotic patterns by combining signal analysis through empirical mode decomposition (EMD), with the Prophet DL model. Hussein and Awad [207] introduce a model that joins RNN with genetic algorithms for forecasting electricity consumption, where the evolutionary component is used for hyperparameter optimization rather than for prediction itself. Another example is the study of Kumar et al. [208], which joins empirical Fourier decomposition and DL models to predict wind speed. In the field of intelligent transport systems, Zhang et al. [194] introduce a hybrid DL model that combine asynchronous spatio-temporal

graph CNN with semi-autoregressive prediction and temporal encoding, successfully capturing both spatial and temporal irregularities in traffic signal data. Similarly, Venkateshwari et al. [195] apply TSs forecasting models alongside IoT-based signal data to support urban planning decisions, highlighting the potential of combining real-time environmental signals with predictive analytics. In the environmental sector, Ajina et al. [182] use artificial neural networks to forecast short-term weather conditions, joining basic signal preprocessing with DL techniques for greater accuracy.

### 2.1.5 Trend detection

The **trend detection** tasks consist on the identification of systematic changes in a TS (short- or long-term), isolating the underlying trend component from random or seasonal fluctuations.

In other words, it involves determining wether a series exhibits a significant sustained upward or downward trend over time, and locating inflection points where the slope changes (e.g. transitions from a upwards trend to a downwards trend, see Fig. 2.5). Definition 2.1.8 formalizes the concept.

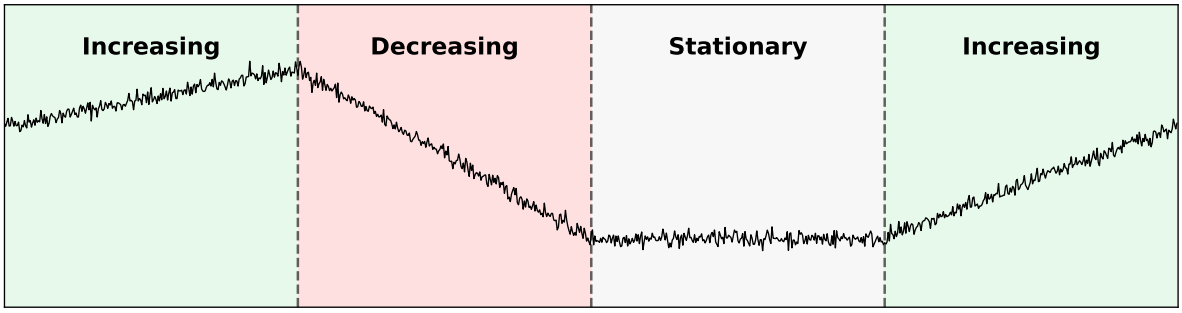



**Figure 2.5:** Trend detection example within a randomly generated TS. In green, upwards trend, in red, downward trend, and in grey, no trend (stationary in mean).

**Definition 2.1.8: Time Series trend detection**

Let $X \in \mathbb{R}^{d \times n}$ be a multivariate TSs. The goal of the **trend detection** task is to define a function

$$f : \mathbb{R}^{d \times n} \to \mathcal{T}_n,$$

where $\mathcal{T}_n$ is the set of all possible trend annotations of length $n$, such that

$$f(X) = (\tau_1, \tau_2, \ldots, \tau_n), \quad \tau_t \in \{\texttt{increasing}, \texttt{decreasing}, \texttt{stationary}\}.$$

The function $f$ assigns to each time step $t$ a label that indicates the local trend direction at that point. Alternatively, the function may output a set of trend segments

$$f(X) = \{(s_i, e_i, d_i)\}_{i=1}^{k},$$

where each $(s_i, e_i)$ represents a segment $X^{(s_i, e_i)}$ of the TS with a consistent trend direction $d_i \in \{\texttt{increasing}, \texttt{decreasing}, \texttt{stationary}\}$.

This task is relevant in both as a standalone goal (e.g., understanding energy consumption evolution [209] or trend pattern of pigeon pea [210]) and as a preprocessing step to facilitate downstream models (e.g., making the TSs stationary). It can also be useful to enhance future

tasks like forecasting or anomaly detection. Trends can be detected in different ways. Mainly qualitative (by visual inspection), quantitative (e.g., by measuring local statistics over sliding windows), and statistical analysis (e.g., testing the stationarity). Some of the techniques are now introduced and classified by research area.

**Statistics.** Classical statistical techniques for trend detection include the Mann-Kendall test (with or without Sen's slope estimator and new adaptations) [79], the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test for stationarity [211]. Bayesian methods have also emerged to detect time-varying changes trend dynamics [80].

**Signal Analysis.** Trend extraction is commonly tackled by isolating the low-frequency components of a signal using time–frequency analysis. Wavelet transforms [118, 212, 213], and EMD [212, 214] can smooth fluctuations to highlight the latent trend. The Hodrick–Prescott filter is also applied to extract the underlying long-term component [215].

**Machine Learning.** Recent ML techniques for TSs forecasting, such as `N-BEATS` [199], `Auto-Former` [216] or `FedFormer` [217], often incorporate mechanisms to model or capture underlying trends, improving long-term forecasting. However, an examination of the current literature (based on targeted keywords searches and exploratory analysis) suggest the abscense of DL models explicitly designed from trend detection as a standalone task. This aspect is mostly considered a preliminary step within the broader forecasting pipeline.

**Data Mining.** From the DM perspective, trend detection involves finding long-term or emerging patterns in large TSs. Techniques like MP have been used for real-time discovery of structural breakpoints in noisy data, enabling the detection of unexpected upward or downward drifts in high-frequency domains such as sensors or finance [109].

**Hybrid methods.** Hybrid models are used to detect trends in complex TSs. For example, Perla et al. [218] propose a DL architecture based on a stacked auto-encoder optimized with a deep kernel-based random vector functional link network (DKRVFLN-AE), whose parameters are optimized using a metaheuristic Weighted Water Cycle Algorithm (WWCA). They use this architecture to capture non-linear trend patterns in volatile forex data. Similarly, Hasan [112] address trend modeling in sales forecasting by integrating classical decomposition techniques like Seasonal and Trend decomposition using LOESS (STL) [219] with Machine Learning models. STL has been used in trend extraction for both economical [112, 220] and medioambiental analyses [221].

### 2.1.6 Time Series segmentation

The segmentation of TSs can be considered either as a preprocessing step for data mining task, as trend analysis techniques (see that Fig. 2.5 represents four segments distinguished by their trends) or as a discretization problem in terms of DM tasks.

Segmentation algorithms take an input TS and divides it into discrete contiguous subsequences (segments or intervals) to reveal regions with consistent temporal patterns (see Fig. 2.6). The goal is to identify segments whose structure differs from adjacent ones, based on changes in statistical properties, shape, or learned representations. Definition 2.1.9 formalizes this idea.

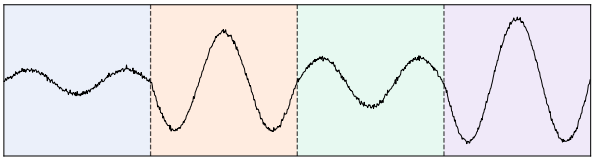


**Figure 2.6:** Random TS with four clear segments.

**Definition 2.1.9: Time Seriess segmentation**

The goal of the TSs segmentation task is to define a function

$$f : \mathbb{R}^{d,n} \to \bigcup_{k \in \mathbb{N}^+} \left( \mathbb{R}^{d \times n_1} \times \cdots \times \mathbb{R}^{d \times n_k} \right)$$

such that $f(X) = \left( X^{(1)}, \ldots, X^{(k)} \right)$ satisfies:

- Each $X^{(i)} \in \mathbb{R}^{d \times n_i}$ with $n_i > 0$,
- $\sum_{i=1}^{k} n_i = n$,
- The segments are ordered and contiguous: $X = X^{(1)} \| \cdots \| X^{(k)}$.

The number of segments $k$ may be fixed or dynamically determined by $f$.

This process allows researchers and analyst to identify patterns, trends and anomalies over time, enabling better insight extraction. It allows to focus on specific periods for more accurate change-point detection or forecasting based on past trends. In finance, segmentation is used to detect distribution shifts - such as changes in mean or variance - that partition TSs into homogeneous intervals for trend prediction [222]. In healthcare, the segmentation of biomedical TSs - such as Electrocardiogram - is used to understand phisiological states and suport clinical interventions [111].

#### Statistics

Statistical time series segmentation methods look for change points by identifying significant shifts in their underlying statistical properties - such as abrupt changes in mean, variance, median, or slope - which often indicate structural changes in the behavior of the data. [223]. Common techniques include the cumulative sum chart (CUSUM), the Pettitt test, and the pruned exact linear time (PELT) algorithm.

CUSUM is a method introduced by Page [224] that accumulates the deviations of individual observations from a target or reference value. A change point is detected when the cumulative sum of these deviations exceeds a threshold. CUSUM is particularly effective in detecting subtle and abrupt shifts in the mean or variance. It has been adapted for robust segmentation in financial TSs [222].

The Pettitt test is a nonparametric method derived from the Mann-Whitney test (Wilcoxon rank-sum test) [225]. This test is used to detect a single point of change in the central tendency - mean, median, mode - of a time series without assuming a specific distribution [226]. It is widely used in environmental and hydrological sciences to detect abrupt changes in rainfall, temperature, and other climate-related variables [227, 228].

Presented by Killick et al. [229], PELT is a method for the detection of exact multiple change

points that is both computationally efficient and versatile. Although it was shown to be exact and fast, recent studies have noted limitations when applied to high-variance or context-sensitive regimes. For example, Takeshita and Kosugi [85] use PELT as a baseline in the context of network traffic prediction, and observe that it struggles to isolate high-variance segments due to its reliance on global variance smoothing. Their results show that a feature-driven segmentation method based on changes in network configuration outperformed PELT in isolating volatile segments.

**Signal Analysis**

Signal-based methods segment TSs by detecting changes in waveform shape or frequency components. These approaches are particularly suitable for physiological or acoustic data, where transitions between states may be smooth or masked by high levels of noise. Classical signal processing techniques include low-pass filters (e.g., Savitzky–Golay [230] or Kalman [100] filters), Fourier [101] and Wavelet [102] transforms, and the Hilbert transform [231].

For instance, Xiao and Wang [71] apply an improved wavelet packet transform and Hilbert Transform to heart sound signals, achieving accurate segmentation even under noisy conditions. Similarly, Solórzano et al. [72] segmented the gait cycle (the sequence of leg movements from one footfall to the next of the same foot) by analyzing the instantaneous phase of joint angles using time-frequency representations.

Radar signal segmentation has also been addressed through frequency-based approaches. For instance, Wang et al. [73] segment pulse sequences in multi-function radar by detecting switching points in the pulse amplitude, identifying transitions between waveform units or radar works (primary tasks of recognizing multi-function radar states) based on changes in the intercepted signal's power measured across multiple sensors.

**Machine Learning**

Machine Learning architectures for TSs segmentation aim to learn internal representations that highlight differences between distinct temporal states, even under complex or noisy conditions. ClaSP [232] formulates segmentation as a binary classification problem, identifying change points by comparing the dissimilarity between short time windows located before and after each candidate position (possible change points) in the TSs; higher dissimilarity suggests a potential segment boundary. This approach has been shown to be effective in multivariate segmentation for human activity recognition [233]. It has also inspired the online variant ClaSS [234], adapted using sliding windows and lightweight classifiers, enabling real-time classification with constant memory. These methods are well-suited to high-dimensional or noisy signals where classical statistical test may fail to capture subtle non-linear transitions.

SVM have also been successfully applied to segment satellite image TSs for mapping land use and land cover, supporting agro-environmental policy making in contexts of conflict between conservation and agricultural production [235].

In addition, recent architectures have expanded segmentation capabilities in complex insustrial and biomedical tasks. PrecTime [236] combines RNN and CNN, achieving near-human performance in segmenting sensor data from hydraulic pump testing. Transformer-based models such as SleepTransformer [237] and U-Sleep [238] have been used in sleep staging tasks, mapping EEG signals to discrete stages with high precision. UTMAS [239] applies a U-Transformer [240] to

capture long-range dependencies for supervised human action segmentation. These models have been adopted in the recent literature as benchmarks for general-purpose segmentation tasks, showing strong results in noisy, weakly supervised, or real-world contexts [241].

#### Data Mining

Data mining techniques for TSs segmentation prioritize scalability, domain-agnostic analysis, and minimal supervision. These approaches often aim to identify boundaries based on shape similarity or contextual variation, without requiring explicit statistical models.

Matrix profile-based methods compare all subsequences within a TS to detect repeated motifs and transitions. They have been widely applied to multivariate segmentation tasks and form the basis of many modern general purpose segmentation algorithms [84].

Feature-driven segmentation, as explored by Takeshita and Kosugi [85], involves detecting change points by monitoring domain-specific indicators, such as connection density or throughput, rather than raw signal values. Their method demonstrated improved accuracy in isolating high-variance segments and enhancing network traffic forecasting compared to classical statistical approaches like PELT.

#### Hybrid methods

Hybrid segmentation methods combine tools from Statistics, Signal Analysis and Machine Learning to better segment TSs, especially when the data are noisy, complex, or come from different sources. These approaches can improve both accuracy and interpretability in real-world scenarios.

For behavior recognition, Yue et al. [242] use hybrid segmentation to classify cattle activity using sensor data. Their method combines statistical change-point detection (PELT), wavelet transforms for denoising, feature extraction and ML classifiers like RandomForest or ExaTrees.

For environmental monitoring, Suárez-Sierra et al. [243] propose a method to segment air quality data. Their approach focuses on TSs of threshold exceedances (moments when polution levels exceed predefined safety limits). It co mbines statistical modeling, Bayesian inference, and genetic algorithms. Events are modeled as a non-homogeneous Poisson process, and genetic algorithm is used to find the segmentation that best fits the data. The method performed well in both synthetic and real-world datasets.

### 2.1.7 Multi-task algorithms

Table 2.1 summarizes the main algorithms and techniques overviewed in the previous sections. The table categorizes the methods according to their research origin and shows their applicability to the introduced tasks.

As illustrated in the table, some techniques show verstility by appearing in multiple tasks. For example, MP stands out as a particularly adaptable method, contributing to anomaly detection, forecasting, trend detection, and segmentation. Similarly, wavelet-based approaches appear in all tasks except imputation, highlighting their broad applicability. Also, ARIMA is applied for classification, anomaly detection, and forecasting, while SVM is found in classification, anomaly detection, and segmentation.

**Table 2.1:** Summary of TSs tasks and associated algorithms/techniques by research area.

| Task | Statistics | Signal Analysis | Machine Learning | DM | Hybrid |
|---|---|---|---|---|---|
| **Classification** | ARIMA [116], Bayesian TS Classifier [117] | Wavelet Transform [45, 118], KL Divergence [74], Meta-transfer [60] | Feature Engineering [68], ClaSP [232, 233], ClaSS [234], Motion Code [69] | Distance-based [46], SVM + Gaussian Kernel [119] | hctsa + ML [120], Bayesian model + Prophet [121], SA+Naïve Bayes +DM+ KNN + Machine Learning [93] |
| **Imputation** | Tall-Project estimator [122], Target-PCA [123] | Frequency-domain diffusion [130] | GRU-D [70], BRITS [132], GAN [133] | Distance-based clustering [135] | Mean/Mode + Classic Machine Learning [137] |
| **Anomaly Detection** | ARIMA [148], Exponential Smoothing [144, 149, 150] | Spectral Analysis [106, 244], Wavelet + Fourier Transform [134, 151] | RNN [147], SVM [145], Ensembles [68], Fractional Stochastic Gradient Descent [157], Bi-stable Stochastic resonance [158] | Matrix Profile [109], Rule Mining [161, 162, 163] | RF + LASSO [66], AlexNet + AdaBoost + NearMiss [166] |
| **Forecasting** | ARIMA [186], sparse generalised Yule-Walker [187] | Wavelet [204] | PatchTST [203], BasicTS [180], Motion Code [69], `N-BEATS` [199], `AutoFormer` [216] or `FedFormer` [217] | Matrix Profile [84] | EMD + Prophet [206], STL + ML [112], DKRVFLN-AE [218], Fourier decomposition + DL [208] |
| **Trend Detection** | Mann-Kendall test [79], Kwiatkowski-Phillips-Schmidt-Shin test [211], Bayesian models [80] | Wavelet Transform [212, 213], EMD [214, 212], Hodrick-Prescott filter [215] | Not found | Matrix Profile [109] | DKRVFLN-AE + WWCA [218], STL + ML [112] |
| **Segmentation** | CUSUM [224, 222], Pettitt test [226, 227, 228], PELT [229, 85] | Wavelet packet transform + Hilbert Transform [71], Wavelet Packet [71], Instantaneous Phase of joint angles [72] | PrecTime [236], SleepTransformer [237], U-Sleep [238], U-Transformer [240], ClaSP [232, 233], ClaSS [234], SVM [235] | Matrix Profile [84], Feature-driven segmentation [85] | Wavelet + PELT + RF [242], Poisson Process + GA [243] |

One of the more representative for the present thesis is MP, a tool from DM that emerges as a particularly adaptable method, contributing to anomaly detection, forecasting, trend detection, and segmentation. It is especially useful because it has been improved for good scalability and is associated with the MPlot, a visual plot (introduced in Section 3.5) that facilitates an initial exploration of TSs patterns.

In addition, Section 3.6 presents the MOMENT and MTSAE algorithms from ML, which can also be applied to multiple tasks. MOMENT is a DL architecture specifically designed to address multiple tasks using different versions of the model. Its performance has been evaluated for imputation, anomaly detection, classification, and forecasting. In contrast, MTSAE constitutes a encoder-decoder model whose latent space has been analyzed across various datasets to assess its capability to capture information relevant to diverse tasks (anomaly detection, pattern detection, trend detection, and segmentation). This representation-based perspective offers insight into how well the model generalizes across analytical objectives and allows the improvement of the interpretability of DVA tools.

## 2.2 Deep Visual Analytics tools for time series analysis

The visual analysis of the original data and the results of the different methods and technologies used for tits analysis is essential for the understanding of its properties. Visual analytics refers to the use of interactive visual interfaces and plots to identify emerging patterns and trends within the data, supporting its understanding. Dimensional reduction techniques are essential for the analysis and interaction with large TSs data [245], as they allow for the application of visual analytics, valuable to interpreting large TS data in fields such as finance, enabling human understanding and supporting algorithm-assisted decision making [246, 247].

This section introduces some examples of visual analytics tools used for TSs analysis, including the one used to test the experimental scenarios within de research (DeepVATS).

### 2.2.1 Interactive tools for visualizing the latent space of DL models

In the field of DL, significant effort has been made to understand the embedding space generated by the different models. This effort has led to the emergence of a specialized reserarch area known as DVA, which focuses on the development of visualization tools to enhance the interpretability and analysis of these models.

Numerous tools exist for interactive visualization of embedding spaces for datasets analysis [87, 88]. Supporting the idea that such tools are particularly valuable for TSs dataset, Suschnigg et al. [248] recently introduced a prototype for interactive visual exploration of anomalies in industrial multivariate TSs data. This system is focused on the anomaly detection task and integrates domain-specific visualizations.

Taking a step further, DeepVATS [7] stands out as an excellent multi-task tool that supports not only anomaly detection but also tasks such as segmentation or trend analysis. The application allows for the exploration of the embedding space of long TSs, by integrating Machine Learning algorithms with interactive VA tools. Although it does not incorporate dedicated algorithms with pattern discovery within the embedding space, it enables the user to visually navigate through it and reason about its complex structures. This visual navigation supports hypothesis generation and iterative model refinement.

DeepVATS applies the HOTL approach by allowing the user to inspect and interpret the Deep Learning model's latent space in a visual and interactive manner. As illustrated in Figure 2.7, the analysis begins with the load of a long TS (univariate or multivariate) that is passed through a DL model for its training. The resulting embedding is then visualized, allowing the user to identify data behaviors such as repeated patterns, trends, or anomalies. Based on these insights, the analyst can adjust the model's input parameters and reinitiate the train-visualize-inspect loop as times as needed. This loop favors an interactive workflow in which visual exploration and model refinement are closely linked.

Figure 2.8 shows the summary of the different tasks in which the tool has proven good capabilities to detect and generate interactive interpretable plots. The results in Rodriguez-Fernandez et al. [7] show how the application can be used to easily analyze segments, seasonalities, repetitive patterns, and anomalies within a TS dataset, while it did not show conclusive results for the trend detection task.

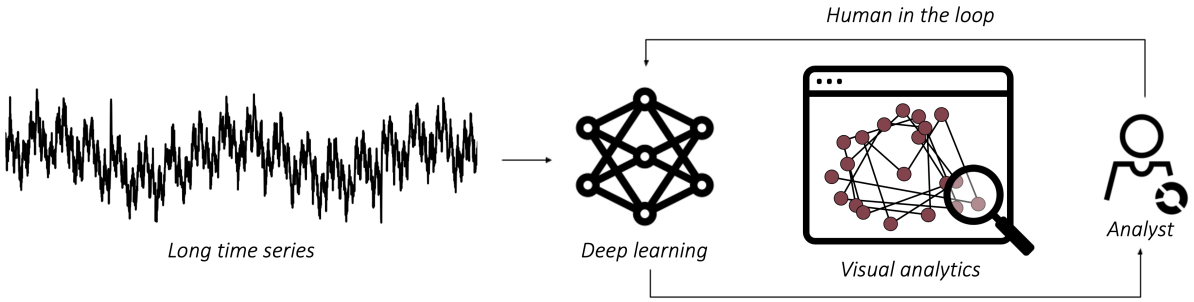



**Figure 2.7:** DeepVATS execution pipeline for long TSs analysis. Figure obtained from [7].

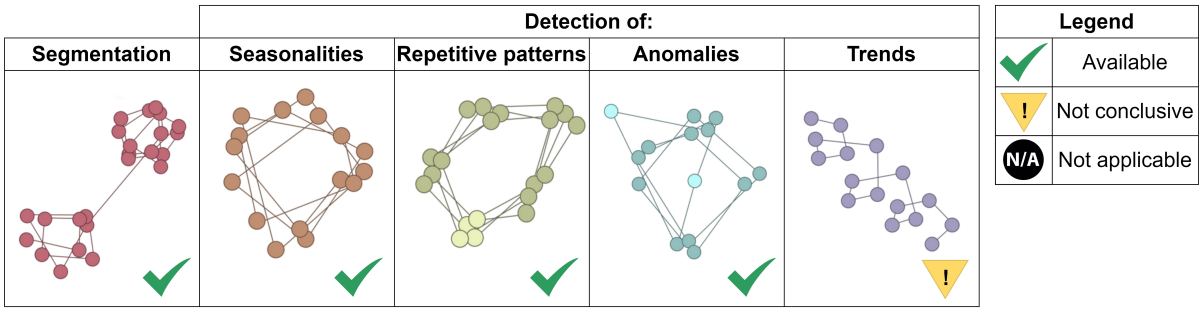



**Figure 2.8:** Summary of DeepVATS capabilities for TSs tasks. Image adapted from [7].

**Enhancing the interpretability by analyzing the embedding space as images**

The visual analysis of TSs can be treated as image analysis since plots are special types of image where the analyst usually looks for clusters or specific patterns. Using DL image analysis may help in the detection of those patterns in an automatic way instead of just direct human analysis, given a wider range of the kinds of pattern that may be detected.

Thus, the visual approach to TSs analysis, Yousef et al. [249] explore how visual analytics techniques can transform industrial process data into meaningful visual representations for fault detection and diagnosis. The visualization of TSs data converts sequential data into images through tools such as the Gramian Angular Field [250, 251] and the Recurrence Plot, which is similar to similarity matrix plots (MPlot [57]). These techniques capture both static and recurrent behaviors of the analyzed systems. This approach reframes fault detection as an image classification problem, maximizing the potential of convolutional neural networks (CNNs) to identify specific patterns within generated images that correspond to different operational modes of the system.

By integrating computer vision algorithms into an accessible visual system, this method enhances the interpretability of TSs analysis, which typically requires laborious manual feature extraction and is limited in scalability and generalisability [252]. This visual analysis of TSs data also addresses the interpretability challenges inherent in CNNs by providing process operators with intuitive graphical representations to identify fault conditions more effectively [249]. The empirical validation of this approach has been conducted in both simulated systems, such as the Continuous Stirred Tank Heater (CSTH), and in real industrial settings, demonstrating competitive accuracy and superior visual interpretability compared to traditional methods.

**Enhancing the interpretability by analyzing the embedding space as graphs**

The use of graphs in DL has advantages in the analysis of communities. One of the advantages is the scalability when clustering datasets. Thus, one can question how well graph-DL models work on the analysis of TSs embeddings or the TSs itself. Recently, some algorithms of graph-DL for TSs are appearing. As a generic model, ATiSE (Additive TS) [253] is a temporal KG embedding that incorporates time information into the entity/relation representations by using the additive TS decomposition. The focus is on the analysis of the evolution of entity/relation representations over time. In the meteorological field, TS are analyzed for detecting outliers using graph embedding for outliers detection [244]. Also, in multivariate TS forecasting, the MTHetGNN framework based on heterogeneous graph embedding has also appeared [254].

Graph-DL for embedding analysis can offer a significant advantage in terms of scalability when handling large-scale datasets, compared to classical clustering algorithms. This space could be interpreted as a graph if a suitable distance metric is defined for each TSs task. This provides a tool to decide which nodes should be joined (perhaps by rules related to KNN) and, consequently, a tool to redefine the TSs features in terms of properties of the nodes of the graph.

**Other approaches for enancing visual analytics**

In addition to the methods already mentioned, there are other techniques that can help to improve the understanding and exploration of the latent space. For example, the use of advanced indices [255, 256] and concrete metrics depending on the different tasks for the validation of clustering algorithms can make a difference in the scalability of the Visual Analytics tools [257], resulting in faster VA tools.

Recent research includes the use of Topological Data Analysis (TDA) to explore latent space. This method captures the general shape and structure of the data, helping to spot anomalies or unusual events [258, 259]. Another useful technique is Self-Organising Map (SOM), which creates visual layouts that keep the original structure of the data. These are helpful for making sense of complex TS with many samples [260, 261].

Chapter 3

# Materials and methods

*If $A, C, E$ are three points on one line, $B, D, F$ on another,*
*and if the three lines $AB$, $CD$, $EF$ meet $DE$, $FA$, $BC$, respectively,*
*then three points of intersection $L, M, N$ are collinear.*

— Pappus & Coexer

The present chapter introduces the main tools for the technical development. The first section describes DeepVATS knowledge extraction algorithms and how a data analyst can check its results. The second section introduces the main theory on MPlot, a visual image containing the similarity matrix information from TSs. Finally, the third section shows how foundation models have been built for TSs.

## 3.1 DeepVATS: technical description

VA refers to the use of interactive visual interfaces and plots to identify emerging patterns and trends within the data, supporting its understanding. Dimensional reduction techniques are essential for the analysis and interaction with large TSs data [245].

DeepVATS [7] provides a tool inspired by TimeCluster [245] that integrates VA and DL (Deep) to extract valuable insights from TSs through interactive projections and TSs data plots. It uses dimensionality reduction (DR) techniques for efficient analysis and interaction with large TSs.

As Fig. 3.1 shows, DeepVATS is based on three modules: the Deep Learning module, the Storage module and the Visual module. These modules interact to perform three main tasks. These tasks include training neural networks to obtain meaningful data representations (embeddings); projecting the content of the trained neural network's latent space in two dimensions (2D) to detect clusters and anomalies; and finally, providing interactive plots to explore different perspectives of the projected embeddings. Although not entirely accurate, the first two modules will occasionally be referred to as back-end and the third as front-end. This terminology is used for clarity, as the first two handle the computation and the last one, the visualization and interactivity.

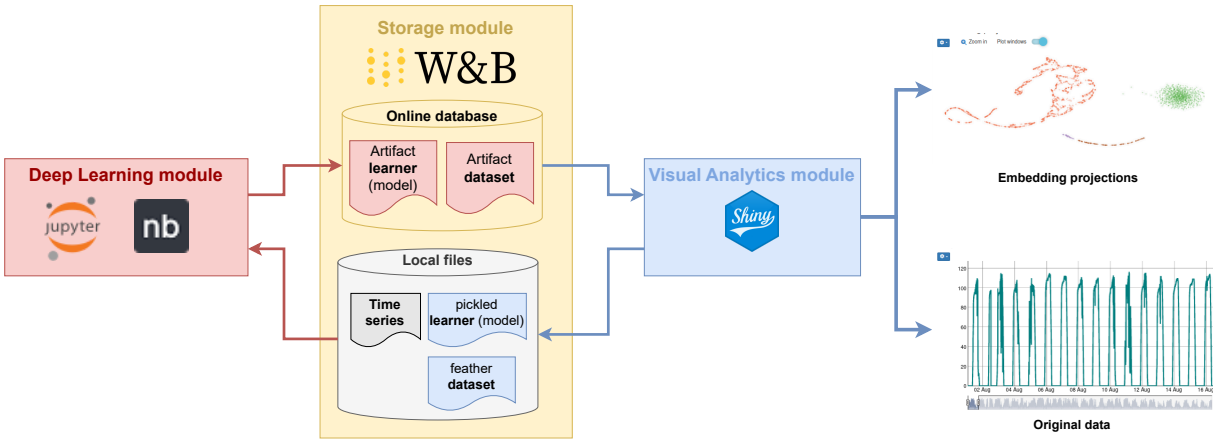



**Figure 3.1:** DeepVATS work-flow diagram.

### 3.1.1 Repository description

The DeepVATS code base[1] integrates three main modules: DL Module, Storage Module and Visual Analytics (VA) Module.

The DL Module is a `Python` library implemented using `Jupyter Notebooks` [262] and `nbdev` [263]. The `Jupyter Notebooks` contained in the folder `nbs` are used for the generation of `dvats` library. This library is the base for both the `nbs_pipeline` notebooks and the `r_shiny_app` (see Fig. 3.2). As detailed in the github repository, the docker containers must be preconfigured, assuming that another folder `work_dir` already exists in the same path as the global "deepvats" for the use of the tool functionalities.

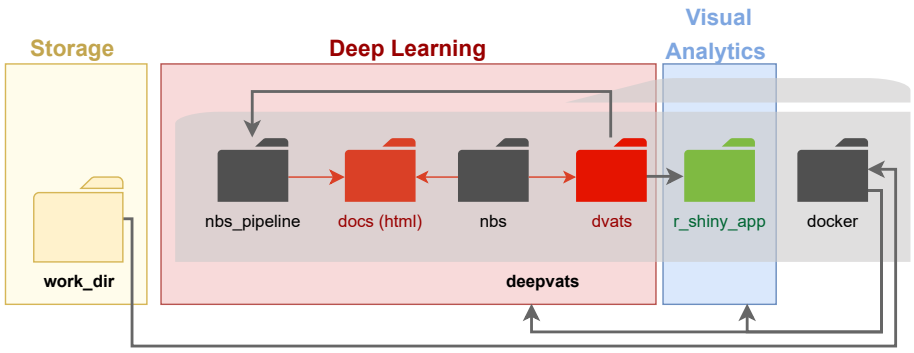



**Figure 3.2:** Main folder structure of the DeepVATS framework. The `nbs` directory contains general-purpose `Jupyter` notebooks used to develop the core DL functionalities. These are compiled into the `dvats` folder, which serves as a reusable library of functions. The `nbs_pipeline` folder includes task-specific notebooks that comprise the DL Module, built on the utilities provided by `dvats`. The `docs` directory holds the autogenerated documentation, while the `r_shiny_app` folder contains the VA Module, which also depends on functions from `dvats`. Red arrows indicate folders generated during the library compilation process using `nbdev`; black arrows denote input or dependency directories.

[1]The source code and the dockers used for the execution of the tool are hosted in the GitHub repository `https://github.com/vrodriguezf/deepvats`. The installation instructions are shown in the README file.

### 3.1.2 Storage Module

The storage module is a simple way to store the data and communicate it between the back-end and the "front-end", ensuring a good flow of the data within the computation. It is based on your local file system and the Weights and Biases (W&B) database [264] (`wandb.ai`), which can also be used to track experiments, as it keeps logs of executions, including specific metrics such as the percentage of Graphics Processing Unit (GPU) usage.

This module is responsible for all the data processing and computations. It provides tools for loading datasets from/to the storage module and use them. To execute an experiment, the first step is to define the associated configuration in the `config/*.yml` files. Some examples with free, accessible datasets are included via direct `Python` execution.

### 3.1.3 Deep Learning Module

The first step of the DL module is to connect the Storage Module in order to build the dataset and learner artifacts that will later be used for its analysis in both the DL and the VA modules.

As Fig. 3.3 shows, once the experiment is configured, `01_dataset_artifact` is used to load a dataset to the Storage Module. The dataset can be originally saved in different data formats (such as `csv`, `txt` or `tsf`). Later, `02_<a,b>_encoder_<DCAE, MVP>` is executed. In this notebook, the dataset $X$ in the generated artifact is separated into windows using `SlidingWindow`, that given the dataset ($X$), the window len ($w$) and the stride ($s$) returns the dataset for the associated sliding window: $X_{windowed} \in \mathbb{R}^{\frac{n-w}{s}+1 \times d \times w}$. Once the time series has been windowed, `tsai`'s `TimeSplitter` is used to split the time series into a training part and a test one based on a percentage ($p$) of the total size. Thus, $X_{train} \in \mathbb{R}^{((1-p)\cdot\frac{n-w}{s}+1) \times d \times w}$ and $X_{test} \in \mathbb{R}^{(p\cdot\frac{n-w}{s}+1) \times d \times w}$ are obtained, being the training set the beginning of the TSs and the last part, the end of the TSs. This value is usually set up to `0.2`.

Once the dataset has been built, the dataloaders, `dls`, are built. This is done with the `tsai`'s `get_ts_dls` function with `X_windowed` as dataset, the obtained split, `tfms = [toFloat(), None]` transformation to ensure the datatype, a configurable `batch_size` specified for the experiment, `bms = [TSStandarize(norm_by_sample, use_single_batch)]` with both parameters configured for the experimentation. This function will normalize the TSs. If `norm_by_sample` is `True`, the mean and the variance are computed for each individual sample; otherwise, an entire batch will be used. Similarly, if `use_single_batch` is `True`, a single training batch will be used to calculate the mean and the variance; otherwise, the entire dataset is used. The result is a `TSDataLoaders`, which is an extension of a `fastai.data.core.DataLoader`.

Once the dataloaders are created, the associated fastai learners' models [265] are defined and trained. As the filenames indiquate, there are currently implemented two algorithms for data encoding: Deep Convolutional auto-encoder (DCAE) [245] and MTSAE [7].

After the model is trained and saved as artifact, the VA module can be used for the analysis. However, the functionalities can be pre-tested in `03a_embeddings`, which is based on `tsai`'s `get_acts_and_grads()` for extracting the embeddings from the trained model [266] and `04a_dimensionality _reduction` can be used for getting the embeddings from the learner and applying UMAP [86], t-SNE or Principal Components Analysis (PCA) (`rapidsai`'s `cuml` implementations [267]). Finally, `hdbscan` is used for clustering [268] through its implementation in the `scikit-learn` library [269] and the plots are shown.

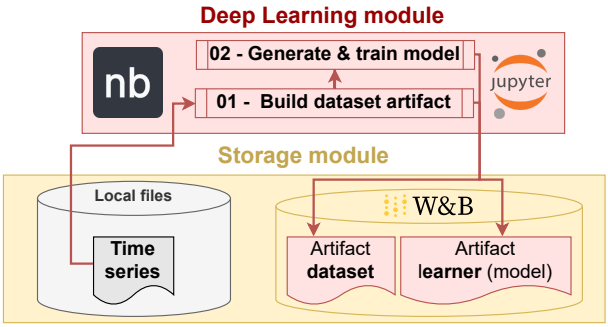



**Figure 3.3:** DeepVATS' load dataset workflow.

In summary, this module allows users to:

- Load your data set from both local files and the W&B database. It allows basic transformations for normalizing the data or selecting different columns in case of multivariate TSs.
- Train the DL model to obtain embeddings. This step uses a MTSAE based on the `tsai`'s `MVP` method [266]. The trained model with the resulting embeddings is saved by the Storage Module.
- Reduce the embedding dimensions. There are different option based on:
  - UMAP [86], which is frequently used for analyzing the embedings [87].
  - t-SNE [270], used, for example, in the TensorFlow's embedding projector [88].
  - PCA [271]. It is being integrated as a first step before computing UMAP to ensure the integrity of the obtained clusters.

### 3.1.4 Visual Analytics module

In order to get dynamic plots depending on the dataset, the encoder, projection algorithms' parameters, and clustering parameters, the VA module is implemented. It includes their parameters so that different configurations can be tested and TSs can be easily analyzed. This module is based on `R` and uses shiny to build an app where a Projections Plot (PP), with the projection of the dimensionally reduced embeddings, and a time series data plot (TSP), with the original data, are shown. As will be described in the next sections, both plots are interactive, so insightful details can be obtained from them.

As Fig. 3.4 shows, the visual module performs the following main steps. First, it downloads all artifacts from the W&B system. Second, from a selected dataset and an associated encoder, it obtains the embeddings and projects them into a PP using `dvats`. At the same time, it generates the TS plot associated with the dataset artifact. The application also includes the option of clustering the projections for better analysis.

The VA module is opened as a browser window that contains the following main elements (see Figs. 3.5, 3.6). On the left, selectors to obtain specific datasets, encoders, and associated configurations:

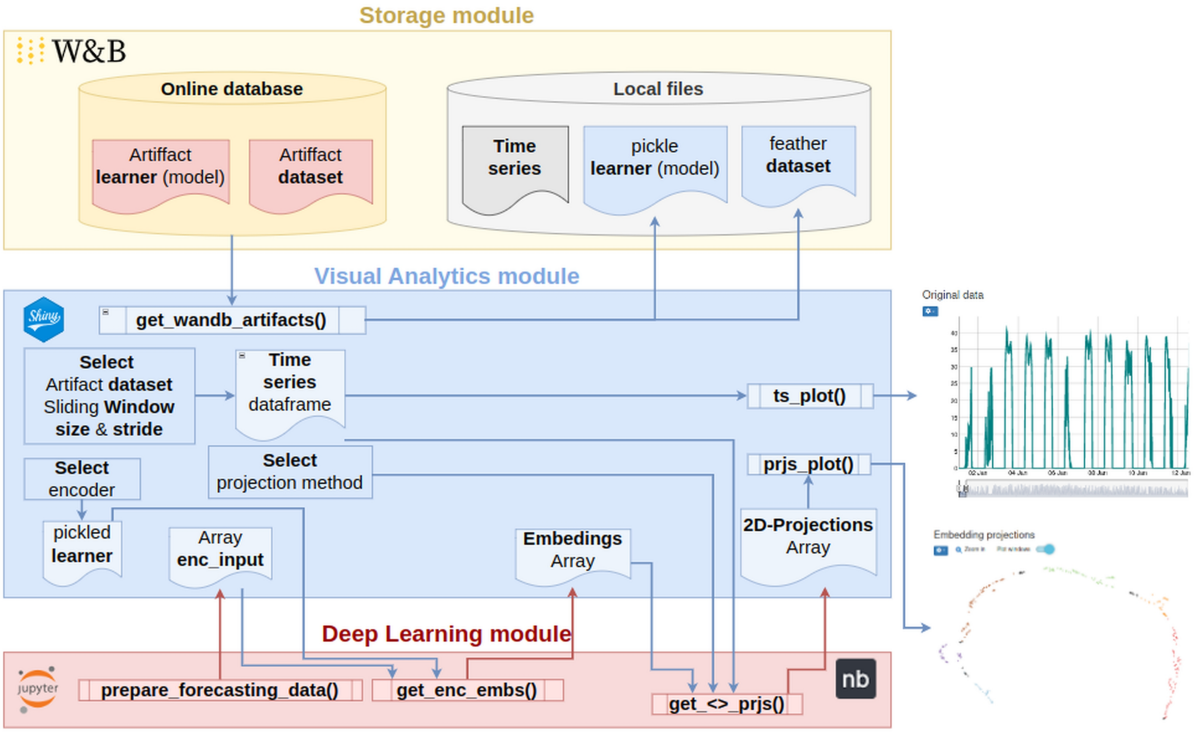



**Figure 3.4:** DeepVATS Visual Analytics (VA) module workflow and interaction with Storage module schema

- Encoder selector. It gives a list with all W&B encoder artifacts trained for the loaded dataset. Default: last uploaded artifact for the dataset.
- Slider window size selector. The minimum and maximum values are defined in the learner artifact metadata, with the maximum value set as the default.
- DR parameters slider selectors: `n_neighbors`, `min_dist` and `random_state`. Default: configured in the artifact configuration file.
- Projections algorithm selector: UMAP, t-SNE, PCA
- GPU/Central Processing Unit (CPU) selector. Default: GPU.
- Clustering option: Calculate and show clusters. For clustering the embedding projections dataset. When calculate and show cluster is selected, hdbscan parameters sliders are shown.

On the right side, there are three tabs. First, `projections` tab, which shows the interactive plots. Second, the `information` tab, which shows the metadata of the selected artifacts. Finally, the `Logs` tab, which summarizes the main logs, including the execution times.

Focusing on the `Projections` tab, the application allows to interact with the plots in different manners. First, changing the aesthetics through the embedding projections options menu for better visualization (see Fig. 3.7). This includes different parameters for the aesthetics and a download button for saving the embedding projections plot.

Once the embeddings projections are configured, more insights on the relation between the


**Figure 3.5:** DeepVATS' Visual Analytics (VA) Module first view example.

embeddings and the TSs data can be obtained. By selecting points in the embeddings projections, their corresponding windows are displayed in the TSs and vice versa. To avoid the application from crashing, the number of timepoints to display has been reduced to 10k. This also gives better rendering performance. The lines below approximately show the windows in the same position as in the graphic timeline. The time selector can be used so that different windows can be analyzed. Also, zoom can be used for better comprehension of the embeddings structure by pushing the zoom in-zoom out button (see Fig. 3.8).

Figure 3.9 focuses on the tabs names in the visual module. The information tab shows the dataset and the encoder metadata, and the Logs tab which, saves a datatable with basic logs of the app.

In summary, this window allows the user to interact with the embedding projections plot and the TSP by selecting different parameters for the computations so better insights on the relation between the embeddings space and the dataset can be obtained.

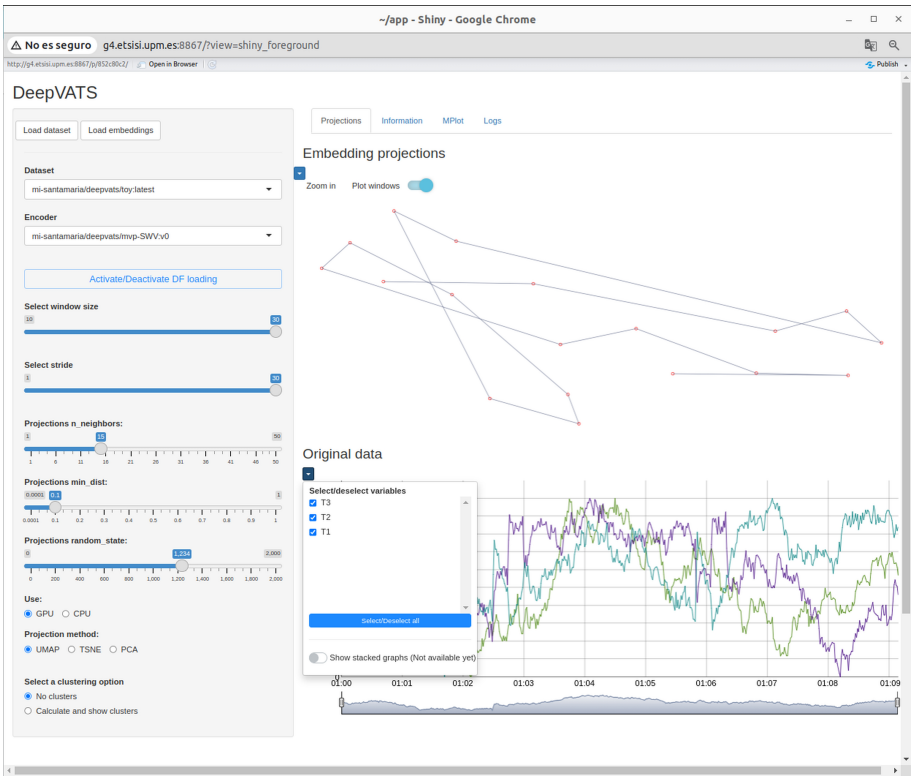



**Figure 3.6:** Example of execution in the Visual Analytics app.

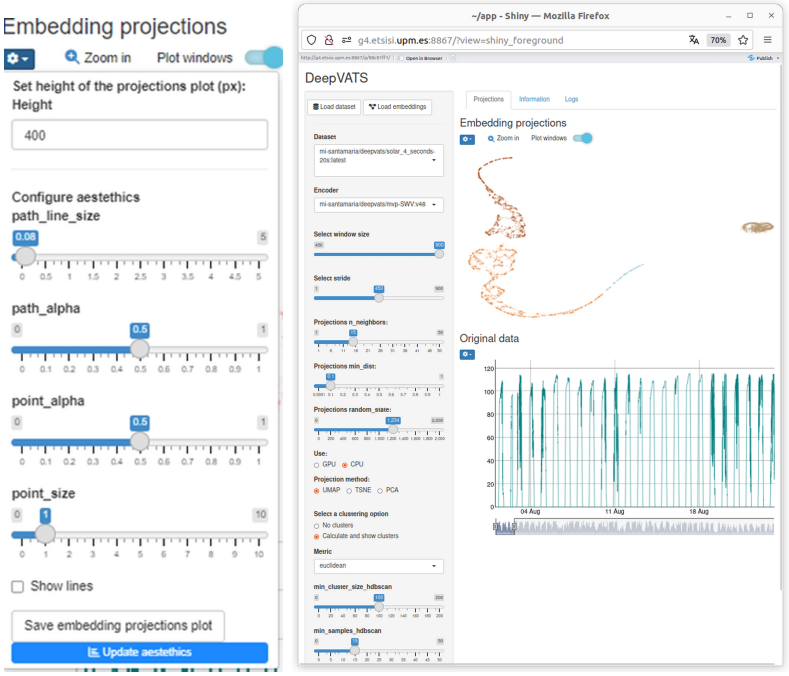



**Figure 3.7:** DeepVATS' Visual Analytics (VA) Module example of projections plot aesthetics configuration


**Figure 3.8:** DeepVATS Visual Analytics (VA) Module projections plot zoom in example


**Figure 3.9:** DeepVATS' Visual Analytics (VA) Module first view example: zoom in the tabs name.

## 3.2 Datasets used for the experimental analysis

The present research is based on both synthetic and real data TSs. For the synthetic dataset, three univariate TSs specifically built in DeepVATS for its analysis and the multivariate synthetic dataset `M-Toy` [272] are used. For the real data TSs, the `Solar Power Dataset (4 Seconds Observations)` dataset [273], and the Kohl's dataset [274] are used.

### 3.2.1 Synthetic data

The DeepVATS repository contains a synthetic dataset generator that is used for creating datasets to test anomaly detection, clustering and segmentation. This generator is found at `_synthetic_dataset` and is based in Time Series decomposition. An important goal in TSs analysis is the decomposition of a TSs into a set of non-observable (latent) components that can be associated to different types of temporal variations [275]. Time Series decomposition is an analytical technique used to split a TSs into different components, typically trend, seasonality and residual (noise) elements. This method allows a better analysis of the underlying patterns and helps in forecasting. The main types of decomposition are the additive (see Def. 3.2.1) and the multiplicative decomposition that uses $X(t) = T(t) \times S(t) \times N(t)$ instead of adding the components. There also exist mixed models that combine both approaches as $X(t) = T(t) \times S(t) + N(t)$ [275].

**Definition 3.2.1: Time Series Additive Decomposition**

Given a TSs $X(t)$, it can be split into three distinct components: trend $T(t)$, seasonality $S(t)$, and noise $N(t)$. The trend component $T(t)$ represents the overall direction or long-term progression of the series along the y-axis over time. Seasonality $S(t)$ captures recurring patterns observed at specific intervals, while noise $N(t)$ accounts for irregular, unpredictable fluctuations that contribute to variability. This decomposition, expressed as $X(t) = T(t) + S(t) + N(t)$, is referred to as the additive decomposition of the TSs.

The additive decomposition works well when seasonal fluctuations are relatively constant over time, making it a common choice when the data shows stable seasonal variation independent of the TSs value. The multiplicative decomposition fits better when seasonal variations are proportional to the trend level, often seen in economic and business data where seasonal effects increase with overall growth. When data exhibits varying seasonal magnitude with trend levels, a log transformation can be applied first to stabilize the variation, allowing the use of an additive model to achieve similar interpretability as a multiplicative approach. The composition techniques can be used in different areas like temporal knowledge graph analysis [253] or air quality prediction [67].

DeepVATS synthetic data generator is based on the Additive decomposition, allowing defining a TSs by giving the following parameters:

- `periods`: define the sequence length. By default, it is set up to seven days ($60 \cdot 24 \cdot 7$ timestamps).
- `start_date`: defines the first timestamp.
- `index_offset`: adds an offset to move the first timestamp in case various segments are going to be joined in the same dataset.
- `hourly_factors`: define the amplitude factor for each hourly frequency group. By default, the 24 hours of a day are grouped in seven groups of two hours.
- `hourly_initial_phase`: sets the initial phase offset for each hourly group, adjusting the starting point of the sine wave in each group.
- `daily_factor`: amplitude factor for the daily (24-hour) component, which scales the intensity of daily variation in the series.
- `daily_init_phase`: initial phase shift for the daily component, altering the starting point of the daily sine wave.
- `weekly_factor`: amplitude factor for the weekly (7-day) component, adjusting the intensity of weekly variation.
- `weekly_init_phase`: initial phase shift for the weekly component, setting the starting point of the weekly sine wave.
- `noise_sd`: standard deviation for Gaussian noise added to the series, introducing random variability.
- `ts_offset`: constant offset applied to the entire TSs, adjusting the baseline level of values.

- `trend`: linear trend factor added to the series, resulting in a constant increase or decrease over time.

The resulting synthetic TSs combines hourly, daily, and weekly seasonal patterns with optional noise, trend, and baseline adjustments, simulating realistic temporal variations over a specified period. The following TSs have already been generated:

- Four synthetic datasets specifically adapted for clustering by forcing the hourly, daily, and weekly factor to get clearly distinct groups.
- For segmentation, a synthetic dataset is specifically created to have four segments.
- For outliers detection, three specific datasets have been defined.
- For trend detection, two TSs are created. The first one, with three different segments with different trends. The second one, with a larger continuous upper trend.

This section introduces segmentation time series (`S1`), outlier detection time series (`S2`), and trend detection time series (`S3`) firstly defined at Rodriguez-Fernandez et al. [7] as well as the multivariate synthetic dataset `M-Toy` [272].

**Segmentation time series (`S1`)**

The `S1` dataset [7] for segmentation analysis is an univariate synthetic TSs spanning 28 days of data, made up of four segments of respective size 7.5, 6.5, 10, and 4 days. Each segment has different patterns constructed by varying the amplitudes of the different seasonal components. More specifically, two of them, the first and the third, maintain the same components with the seasonality of less than a day, and only the components with daily and weekly seasonality vary, maintaining a very similar daily pattern in both segments. Table 3.1 shows the specific duration for each segment and the number of timestamps resulting for building the final TSs (see Fig. 3.10), that is a resampling of the datasets in groups of 20 minutes based on the mean of each batch.

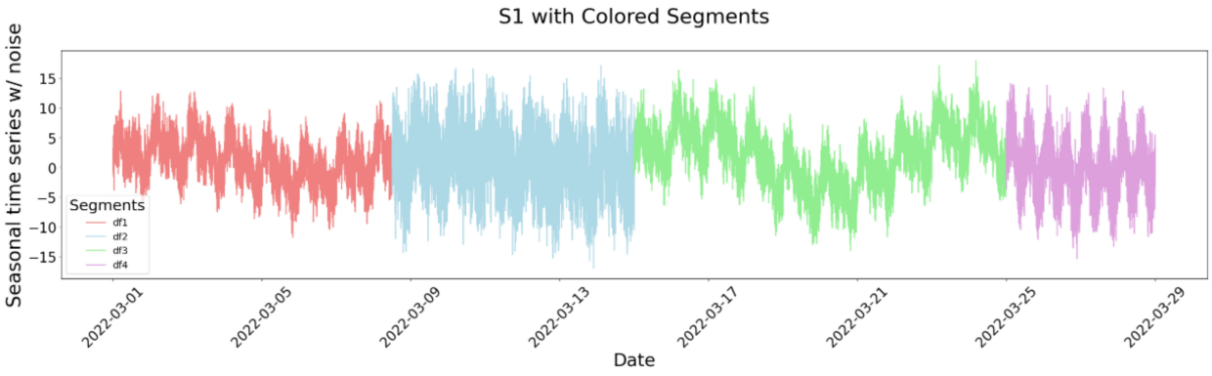



**Figure 3.10:** `S1` TSs with the four segments shown in different colors.

In order to train the MTSAE encoder and choose the best windows for the analysis of the synthetic dataset, it is interesting to note that $540 = 5 \cdot 3^3 \cdot 2^2$, $465 = 31 \cdot 5 \cdot 3$, $720 = 3^2 \cdot 5 \cdot 2^4$ and $288 = 3^2 \cdot 2^5$, resulting in $mcd(540, 720, 288) = 3^2 \cdot 2^2 = 36$ and $mcd(540, 465, 720) = 3 \cdot 15 = 15$, which gives 15 and 36 as good windows to check. Also, for readability, the first and the third datasets 54 and 72 seem to be easy values to check. Thus, following the original analysis, the

**Table 3.1:** The four segments, $df_*$, of synthetic S1 dataset, including the number of timestamps in a 20 minutes sample.

| Segment | Start date | End date | Days | Hours | Minutes | Timestamps |
|---|---|---|---|---|---|---|
| $df_1$ | 2022-03-01 00:00 | 2022-03-08 11:59 | 7.5 | 180 | 10800 | 540 |
| $df_2$ | 2022-03-08 12:00 | 2022-03-14 23:59 | 6.5 | 156 | 9360 | 465 |
| $df_3$ | 2022-03-15 00:00 | 2022-03-24 23:59 | 10 | 240 | 1440 | 720 |
| $df_4$ | 2022-03-25 00:00 | 2022-03-28 23:59 | 4 | 96 | 57600 | 288 |
| **Complete** | **2022-03-01 00:00** | **2022-03-28 23:59** | **28** | **672** | **40320** | **2013** |

dataset is first trained using $w \in [36, 72]$ and looking through 54 and 72 windows with stride 2 for the different segments of the TSs. Figure 3.11 shows that using window 54 leads to a poor analysis (apparently taking note on shorter patterns rather than segmentation) but using 72 yields to a similar result to the original paper [7]. Segment $df4$ is really easy to detect as an appart cluster in both cases. Also, segments $df1$ and $df3$ seem to be easy to detect but clusters are mixed in terms of time, detecting more other type of patterns than the global segments. And $df2$ can be inferred from the third cluster that mixes a little bit with the other two clusters.

### Outlier time series (S2)

The S2 time series , saved as `synthetic_outliers(III).csv` has specially been created for the analysis of algorithms capability to detect point anomalies in TSs. It is an univariate TSs spanning 20 days built using three large segments that simulate the complete TSs and two short segments of two hours each, at the beginning of the fifth date and the end of the fourteenth that cause two anomalies (see Fig. 3.12 and Tb. 3.2).

**Table 3.2:** The four segments, $df_*$, of synthetic S1 dataset, including the number of timestamps in a 15 minutes sample.

| Segment | Start date | End date | Days | Hours | Minutes | Timestamps |
|---|---|---|---|---|---|---|
| $df_1$ | 2022-03-01 00:00 | 2022-03-04 23:59 | 4 | 96 | 5760 | 384 |
| $df_2$ | 2022-03-05 00:00 | 2022-03-05 01:59 | ≈0.08 | 2 | 120 | 4 |
| $df_3$ | 2022-03-05 02:00 | 2022-03-14 21:59 | ≈9.83 | 236 | 14160 | 944 |
| $df_4$ | 2022-03-14 22:00 | 2022-03-14 23:59 | ≈0.08 | 2 | 120 | 4 |
| $df_5$ | 2022-03-15 00:00 | 2022-03-20 23:59 | 6 | 144 | 8640 | 576 |
| **Complete** | **2022-03-01 00:00** | **2022-03-20 23:59** | **20** | **480** | **28800** | **1920** |

To analyze S2, the original parameters in [7] are used; that is:

**Masking:** `r`= 0.5, `stateful=false`, `future=false`.

**Sliding window:** `w`= 48, `w_min`= 24, `w_max`= 48.

**Batches** `batch_size`= 32.

**Epochs** `n_epochs`= 200.

Using the window length 28 (which corresponds to 7 hours), MTSAE retains the information of

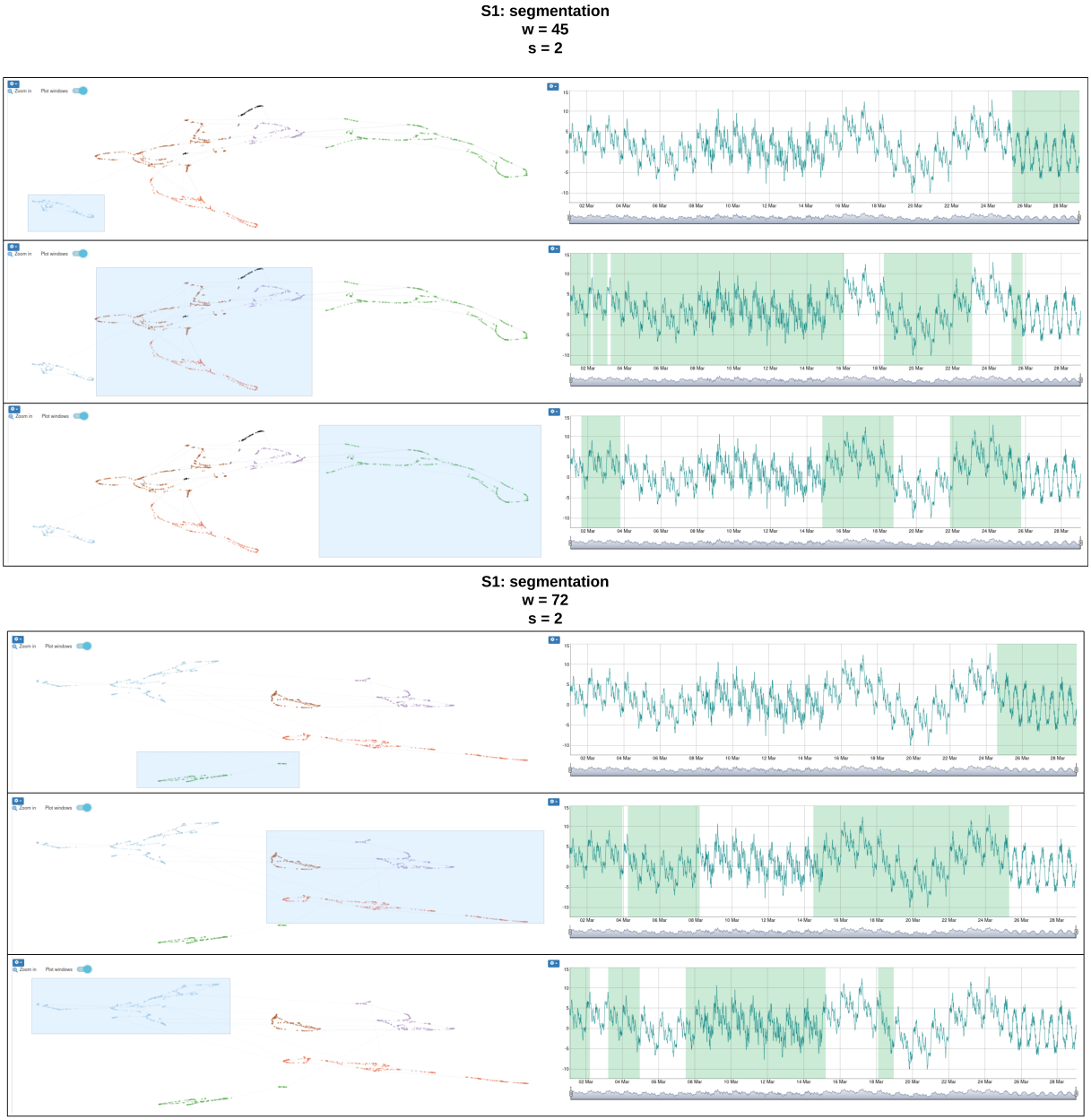



**Figure 3.11:** Segmentation synthetic dataset (S1) analysis using stride = 2 and window lengths 54, 72. The parameters for UMAP reduction and clustering are: `n_neighbors = 21`, `min_dist = 0.0001`, `random_state = 770`, `min_cluster_size_hdbscan = 100`, `min_samples_hdbscan = 15`, `cluster_selection_epsilon = 0.08`.

the time series focused on the related patterns (see Fig. 3.13). It is interesting to note how the patterns are clearly separated with few lines in-between. All together conform approximately one day of the TSs (that is "a larger pattern") and the little patterns are connected ordered in time (see the row of patterns below the PP). It is also clear how it distinguishes the anomalies in a far away zone connecting them with the other clusters just by the initial and final time line between windows.

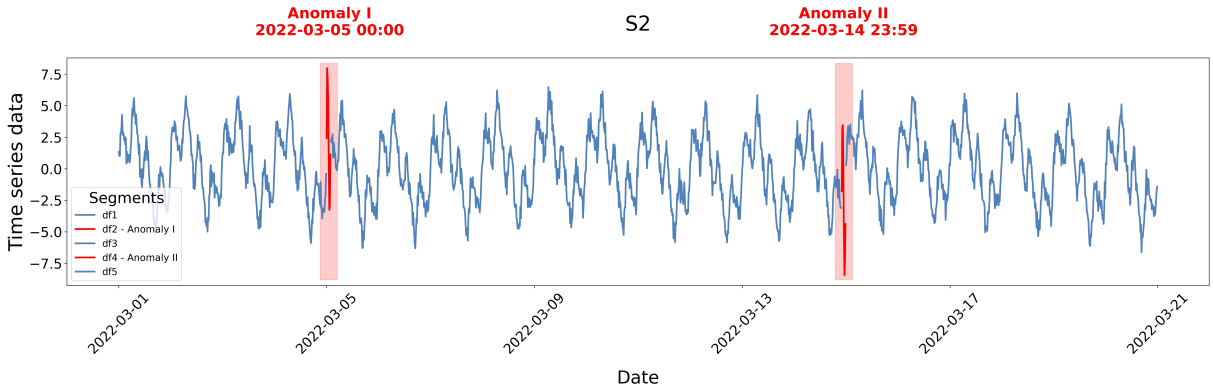



**Figure 3.12:** Resampling of the outlier TSs (S2) taking the mean in batches of 15 minutes. The red segments show the anomalies injected in the synthetic dataset. This anomalies are at the begining of the day 5 and the end of the day 14.

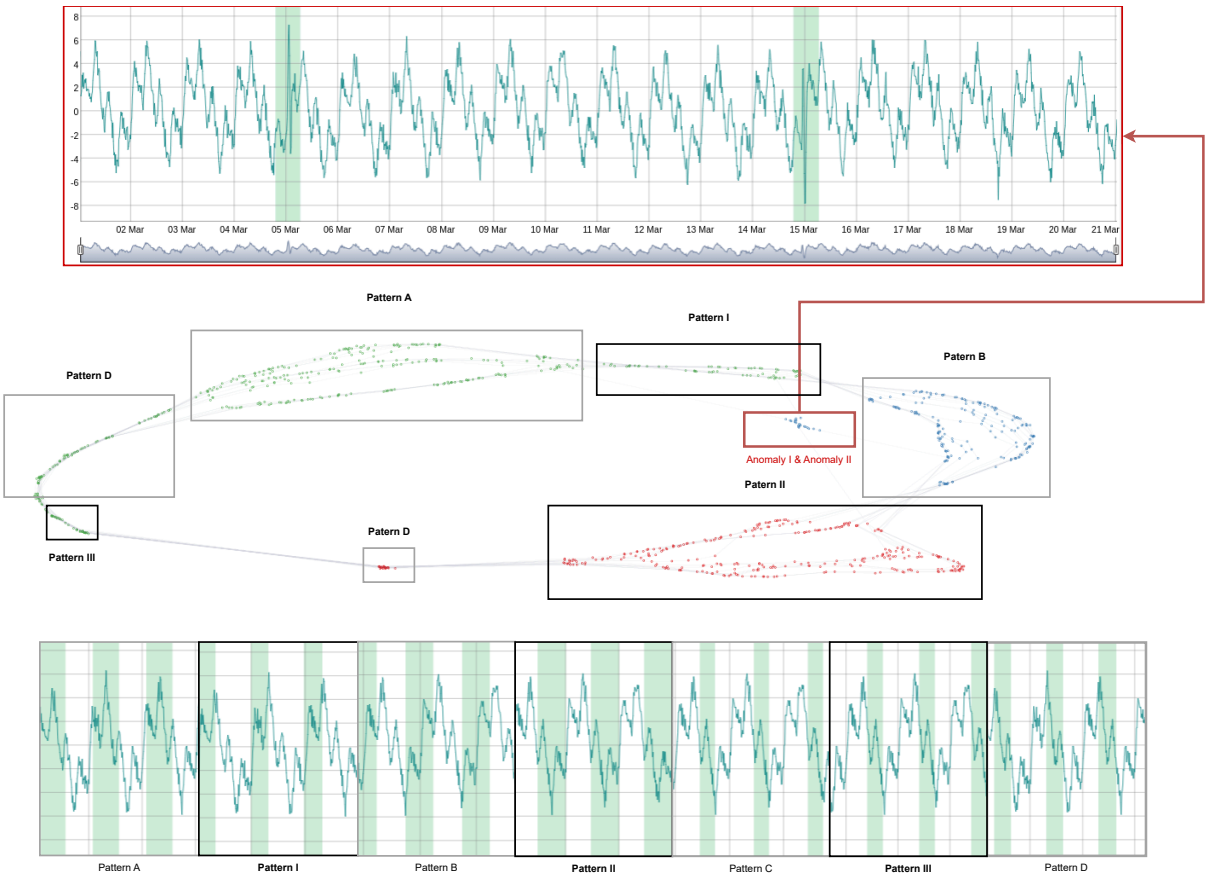



**Figure 3.13:** Analysis of S2 . The analysis is done for window length 28 and stride 2. The parameters for dimensionality reduction (using PCA followed by UMAP with GPU) are `n_neighbors`= 15, `min_dist`= 0.1, `random_state`= 1394. The parameters for cluster computation are `metric=euclidean`, `min_size`= 40, `min_samples`= 15,`epsilon`= 0.08.

### Trend TSs (S3)

The S3 time series, saved as `synthetic_trends(II).csv` within the synthetic data generator is generated with a length of 20 days starting at 2022/03/01. It has a trend of 0.002 to ensure

a low but constant trend from the first index to the latest one (see Fig. 3.14. This TSs is used to analyze the ability of DeepVATS to explain the TSs data trend.

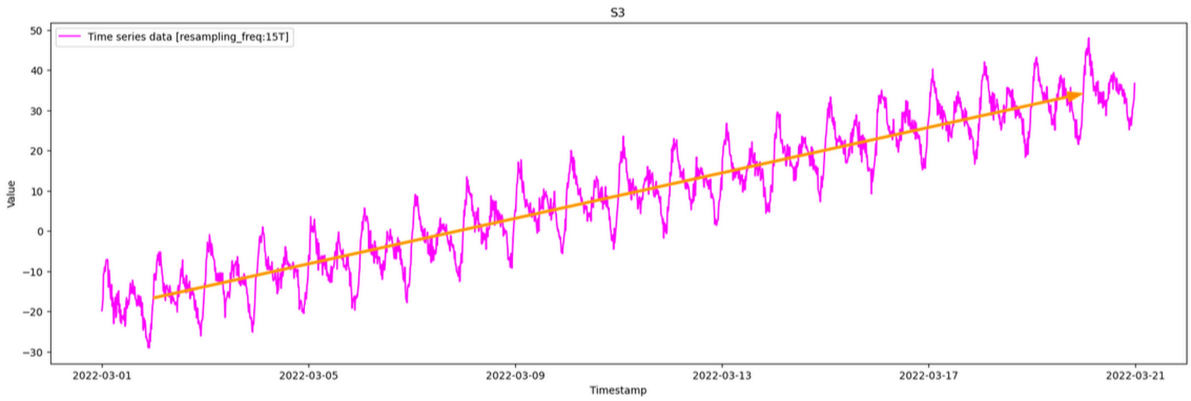



**Figure 3.14:** Resampling of S3 taking the mean in segments of 15 minutes. The arrow shows its upper trend.

In the case of S3, the TSs has a of length $60 \times 24 \times 20 = 28800$ timestamps and 1 minute frequency. Once resampled to 15 minutes batches by taking the mean, it results in 1920 timestamps with 15 minutes frequency. By training with windows 32 and 96 the focus is on subsequences of 8 and 24 hours respectively. By taking 48 length, the analysis comprises 12h. However, following the original analysis, the results are reproduced with a length of 50 (12.5h). Figure 3.15 reproduces (simplified) the figure 12 in [7]. The image shows how it seems that the trend can be captured by following the connected clusters, which seems to lack many times in common (only one/two lines connecting one cluster with the next one) while distanced-time clusters do not have any cluster in common, showing that something is changing across the time that makes the model separate time-separated subsequences in the embedding space.

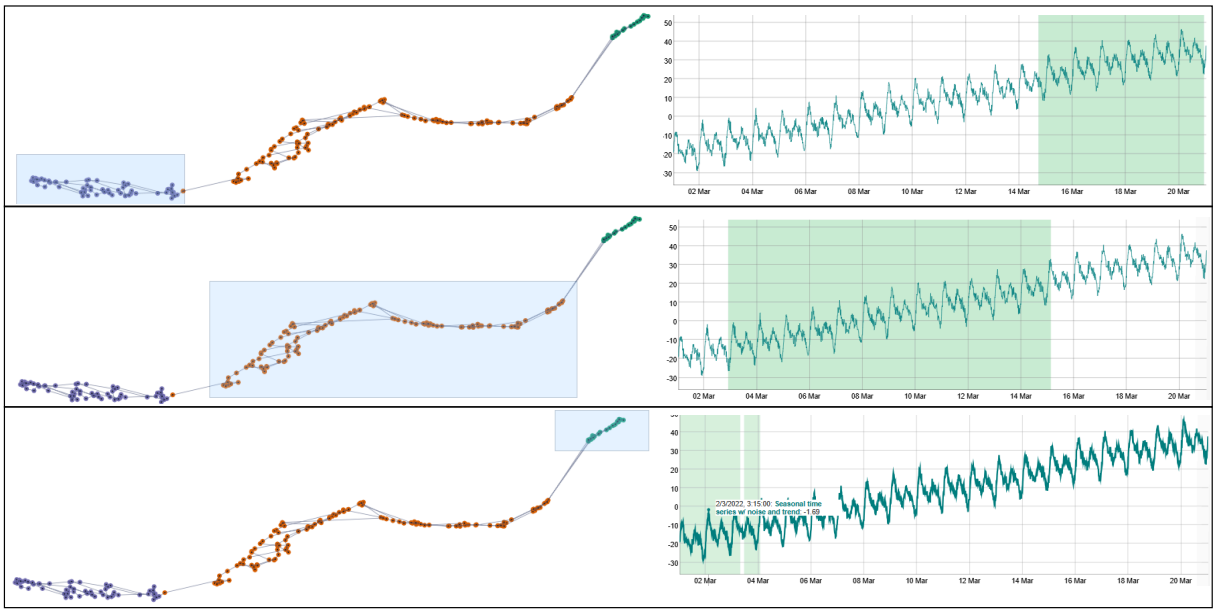


**Figure 3.15:** S3 analysis in DeepVATS using the MTSAE encoder trained for window lengths from 32 to 96 and infered with window length 50 and stride 9 as in [7]

.

#### `M-Toy` (for anomaly detection)

The `M-Toy` dataset is a synthetic multivariate TSs available through the `stumpy` library [272], originally inspired by an example dataset used in [276]. It consists of three features, denoted $T1$, $T2$, and $T3$, each with 550 samples. Among these features, $T1$ and $T2$ contain two artificially embedded motifs (identical subsequences of length 30) inserted into the timestamps 150 and 350. These motifs represent recurrent patterns. In contrast, $T3$ is a random walk (a signal where each value is the sum of the previous one and some random noise) intentionally included to simulate an irrelevant or noisy dimension.

Figure 3.16 shows the repeated patterns. Originaly added as motif (repeated patterns); as they are only two appartitions, they can also be considered as segment anomalies. The charasteristics of the time series makes it perfect for testing the capabilities of VA tools in detecting TSs anomalies in presence of uncorrelated features.

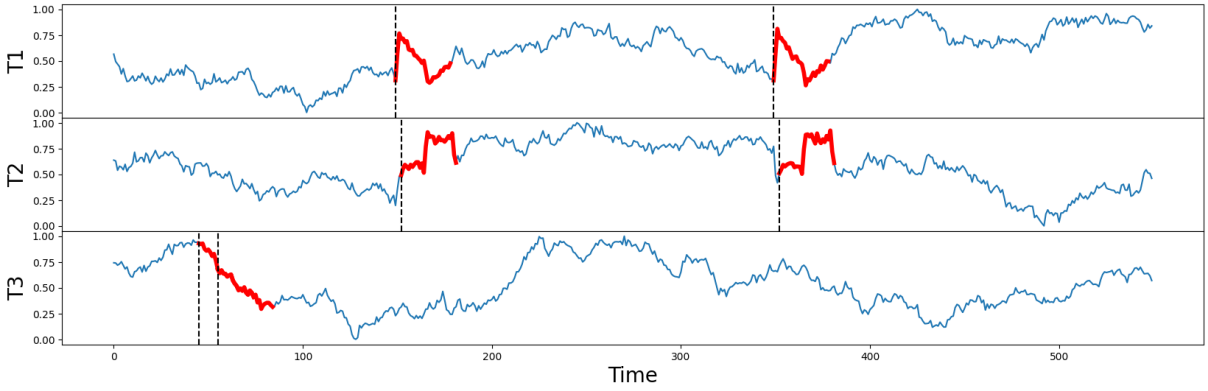



**Figure 3.16:** `M-Toy` dataset. In red, the detected motifs or anomalies, showing the relation between $T1$ and $T2$ and their independence from $T3$. Figure obtained from stumpy.readthedocs.io/Motif_Discovery.

### 3.2.2 Real data

This section introduces the real data TSs used for the research: the `Solar 4 seconds` dataset (from the Monash benchmark [277]) and the Kohl's dataset [274].

#### `Solar 4 seconds`

The Monash benchmark is a repository for TSs forecasting containing 30 TSs [277]. The largest TSs within the benchmark is the `Solar Power dataset (4 Seconds Observations)` dataset, hereinafter referred to as `Solar 4 seconds`. This dataset contains a very long univariate TSs representing the solar power production in MegaWatts recorded every 4 seconds starting from 01/08/2019. It was downloaded from the Australian Energy Market Operator (https://aemo.com.au/) online platform. It contains a total of $7397222 \approx 7.4$M elements.

The size of the `Solar 4 seconds` datasets makes it ideal to test the scalability of the different algorithms and functionalities of DVA tools, allowing its resampling through different frequency factors to test the application in different sized datasets (See Section. 3.4).

### Kohl's

The `Kohl's` dataset was presented by Matsubara et al. [274]. It summarizes the number of searches for the keyword "Kohl's" on Google. This term refers to an American retail chain (Koh'ls). The samples were collected weekly over a decade starting in January of 2004. The TSs shows a growing trend, with significant but unsurprising peaks during the end-of-year holiday season. These bumps generally increase over time (see Fig. 3.17). Its characteristics make it ideal for testing the capabilities of visual analytics tools in detecting trends and patterns.

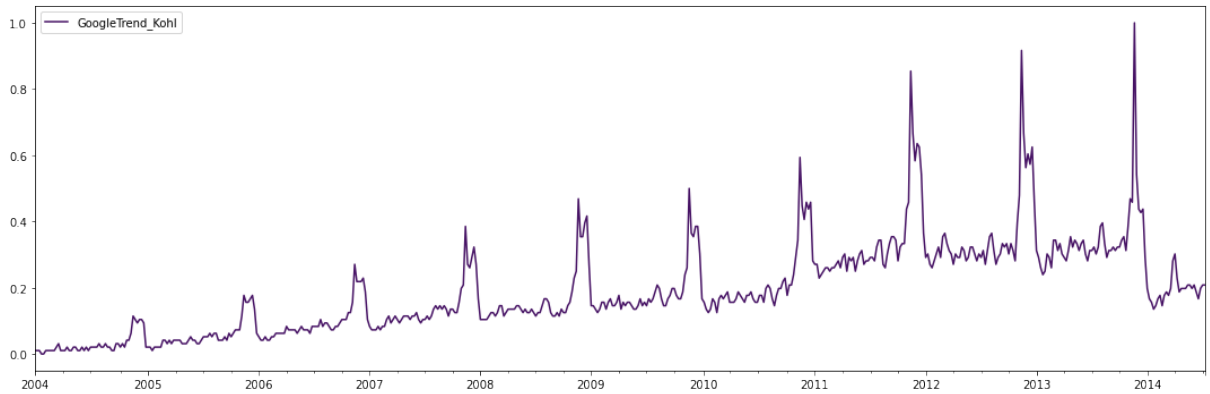



**Figure 3.17:** Kohl's dataset. Figure obtained from [7].

### Pulsus Paradoxus

Pulsus paradoxus, also known as paradoxic pulse or paradoxical pulse, refers to an abnormal decrease in systolic blood pressure - exceeding 10 mmHg - and pulse wave amplitude during inspiration. This condition can be indicative of serious health issues, such as cardiac tamponade [143].

The `PulsusParadoxus`[2] dataset is a real-world dataset previously analyzed by Shahcheraghi et al. [11]. It contains an univariate TSs that records the oxygen saturation ($SPO_2$)[3] of a patient.

The timeseries was annotated by a clinician present during the recording section, who may have had access to contextual information beyond the raw signal. Although specific algorithms tailored to detect Pulsus Paradoxus exist, the dataset aims to evaluate general-purpose TSs models that do not rely on domain-specific assumptions. While Electrocardiogram (ECG) - the recording of the heart's electrical signals - is commonly used in clinical settings to assess conditions like Pulsus Paradoxus, it is not part of this dataset.

The dataset contains a total of 17521 time steps, 54 being the length associated with 1 heartbeat (1 pulsus; see Fig. 3.18).

The characteristics of the pulsus paradoxus make this dataset specially useful to analyze the

[2]Dataset available at `https://drive.google.com/file/d/1mJs_FSjSnffw2xPJhu3S1ES2kXSCMoNy/view`. Although it is also believed to be recorded in the UCR Time Series Classification Archive [278, 279] (check https://www.slideserve.com/old_UCR_timeseries_semantic_segmentation_archive_presentation), finding the corresponding entries in the different repositories has proved difficult. Thus, to ensure reproducibility, the dataset is hosted in the thesis repository at github.com/misantamaria/thesis/Datasets/PulsusParadoxus.txt

[3]Measures the percentage of hemogoblin capable of carrying oxygen in the blood.

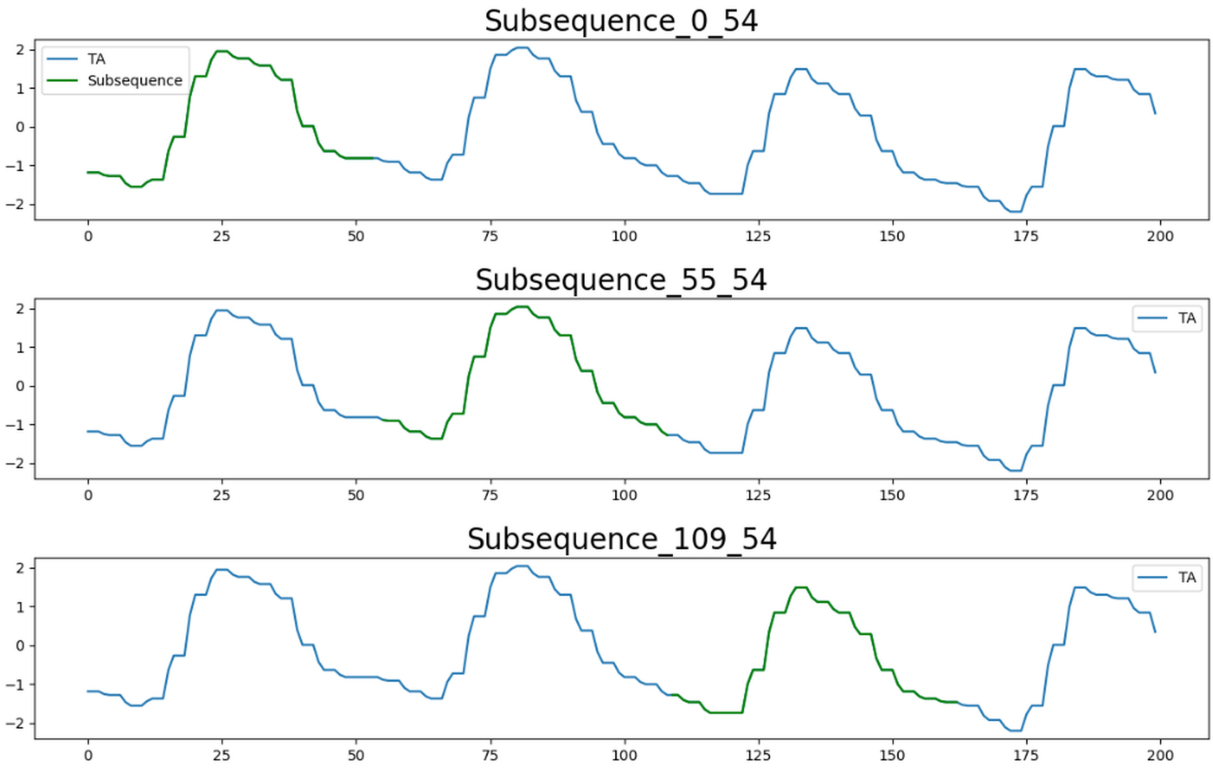



**Figure 3.18:** First Pulsus Paradoxus data subsequences of size 54.

capabilities of models on detecting patterns that appear at an abnormal point in the TSs but may repeat shifted and with different "heights" in future values.

## 3.3 MTSAE architecture description

This section introduces the MTSAE architecture. It uses the MVP callback [266], which sometimes leads to referring to the resulting model MVP. The model is is built for each timeseries, $X \in \mathbb{R}^{d \times n}$ through the following steps.

Once the dataloaders are defined as described in Section 3.1, `tsai.learner.ts_learner` is used to build a `fastai.Learner`. The main characteristics of this learner are:

**dls** The dataloaders' parameter is assigned to the computed *dls*.

**InceptionTimePlus** The model is assigned to `tsai.models.cnns.InceptionTimePlus`.

**cbs** The callbacks include a `WandbCallback(log_preds = False)` to log the experiment into W&B, a `ShowGraphCallback2()` to update and display a graph of the training and validation loss during the training process; and the Masked Value Predictor callback `tsai.training.Callback.MVP` used to predict TSs step values after a binary mask has been applied.

The next subsections introduce the MVP callback and the InceptionTimePlus model.

### 3.3.1 Masked Value Predictor callback

The Masked Value Predictor (MVP or TSBERT) callback receives the following main parameters, configured for each experiment:

**r** Probability (ratio) of masking.

**ws** Window size tuple `(window_len_min, window_len_max)`.

**future_mask** If `True`, the model will be trained as a forecasting model. Thus, the generated mask will mask the values at the end of the sequence. If `False`, the values will be created directly for all timestamps.

**sync** If *True*, all features have the same masks.

**stateful** If *True*, a geometric distribution is used to define the mask sequences with an average mask length of 3 elements.

The *ws* tuple is used to define the window length limits. These values are used in the `before_batch` function to select a random value, $w\_rand$, between them both. Then $w\_start$ is defined as a random value between 0 and $w - w\_rand$ to build the batch input, target, and mask (by selecting $tensor[..., w\_start : w\_start + w\_random]$ for their original values, using only those timestamps for the next training step). See that $window\_len\_max$ must be less than or equal to $w$ to avoid computation errors. Also, Fig. 3.19 shows examples of masking for the different available configurations.

### 3.3.2 Encoder-Decoder architecture

The encoder-decoder architecture was introduced by Cho et al. [280] to get semantically and syntactically meaningful representations of linguistic phrases. It consists of two RNN: the encoder (first RNN) processes a sequence of symbols and compresses it into a fixed-length latent vector while the decoder transforms this representation into another sequence of symbols (see Fig. 3.20). Both components are jointly trained to maximize the conditional probability of a target sequence given an input sequence. The encoder-decoder architecture has demonstrated robust efficacy in sequence-to-sequence tasks such as machine translation. Consequently, it has been across diverse domains. Among the many uses of encoder-decoder architectures, AutoEncoders represent a particular case in which the target output is identical to the input, and the goal is to reconstruct the original sequence rather than transform it.

In the field of TSs, encoder-decoder has been successfully applied to a variety of tasks such as forecasting, prediction, imputation, or classification. For forecasting, variants such as Long-Term Short-Term (LSTM)-based encoder-decoder architectures have been employed to capture complex temporal dependencies and generate predictions [281, 282, 283]. In particular, attention mechanisms and hybrid architectures that combine convolutional and recurrent layers have been shown to enhance predictive accuracy in domains such as hydrology [282] or oil prediction [283]. For data imputation, encoder-decoder structures have been integrated with temporal attention to reconstruct missing values in multivariate series [284, 285]. In classification, attention-augmented bidirectional LSTM-based encoder has been applied to biomedical signals such as Electrocardiograms (ECGs) [286]. These examples illustrate the versatility of encoder-decoder models in handling diverse temporal learning tasks.

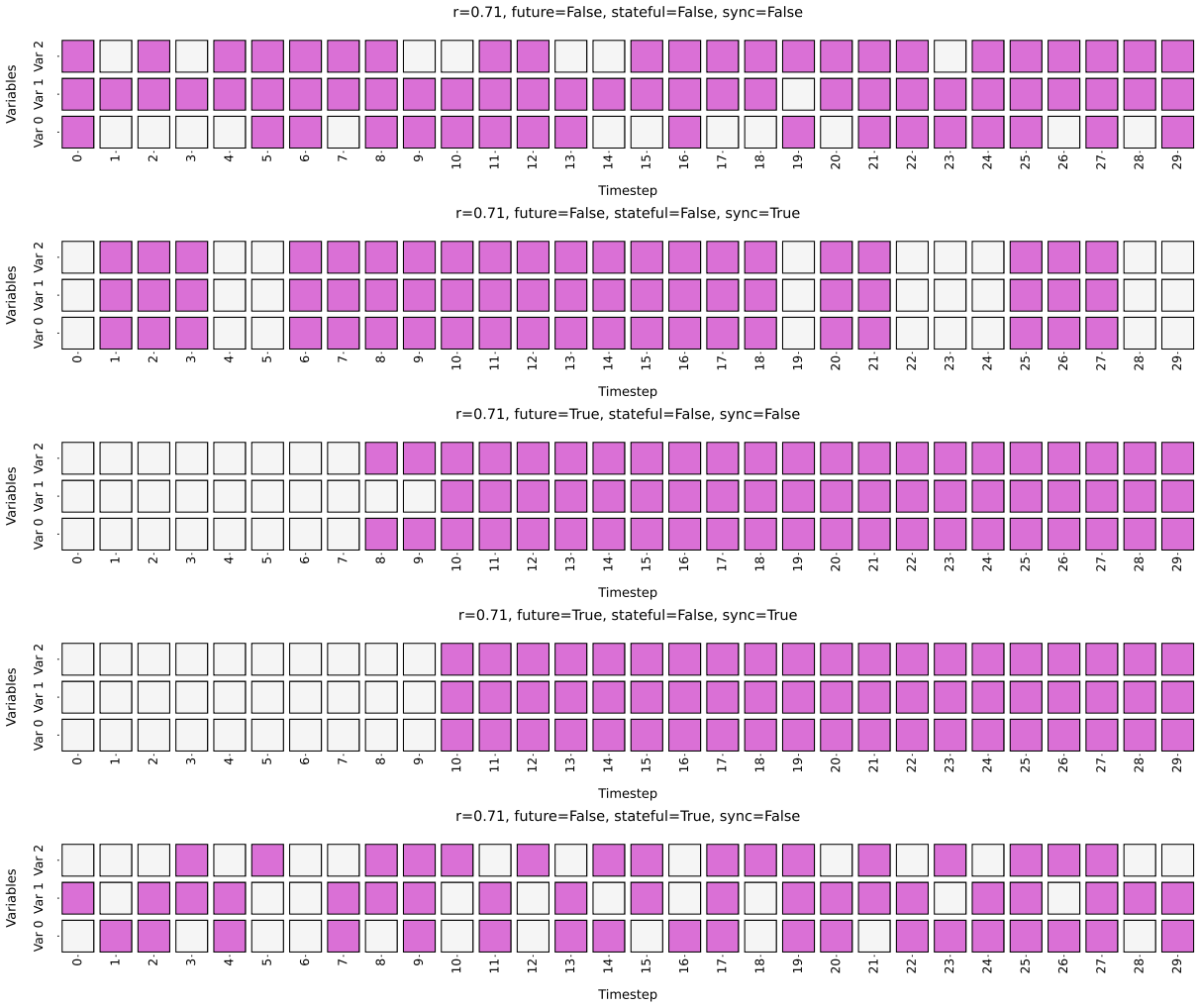



**Figure 3.19:** Examples of mask for M-Toy dataset used in MTSAE modifing the diferent configurable parameters. Take into account that it is just the first element of a batch and the masking percentage (r) is aplied to the full batch, so the number of masked elements may vary between the examples.

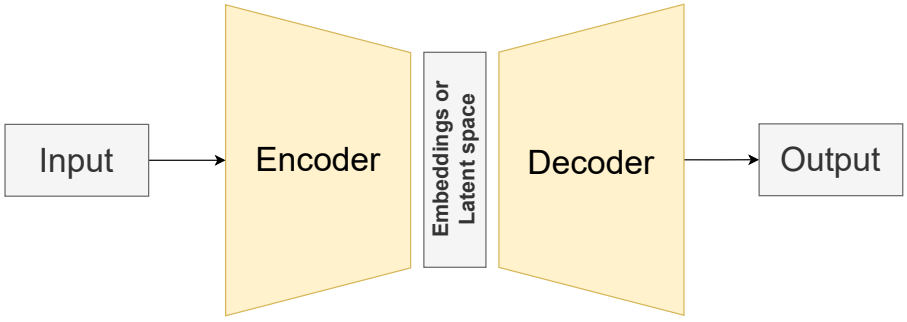



**Figure 3.20:** Visual schema of the encoder-decoder architecture.

### 3.3.3 InceptionTimePlus model

The InceptionTimePlus model is based on InceptionTime, an AlexNet adaptation for time series. [287, 166]. The model is an encoder-decoder with two main parts: the backbone (encoder) and

the head (decoder). The input is the TSs $X \in \mathbb{R}^{d \times n}$. The series is passed through five CNN InceptionBlockPlus blocks. Figure 3.21 shows the associated block diagram while Figs. 3.22, 3.23 remark the execution pipeline including the different vector sizes.

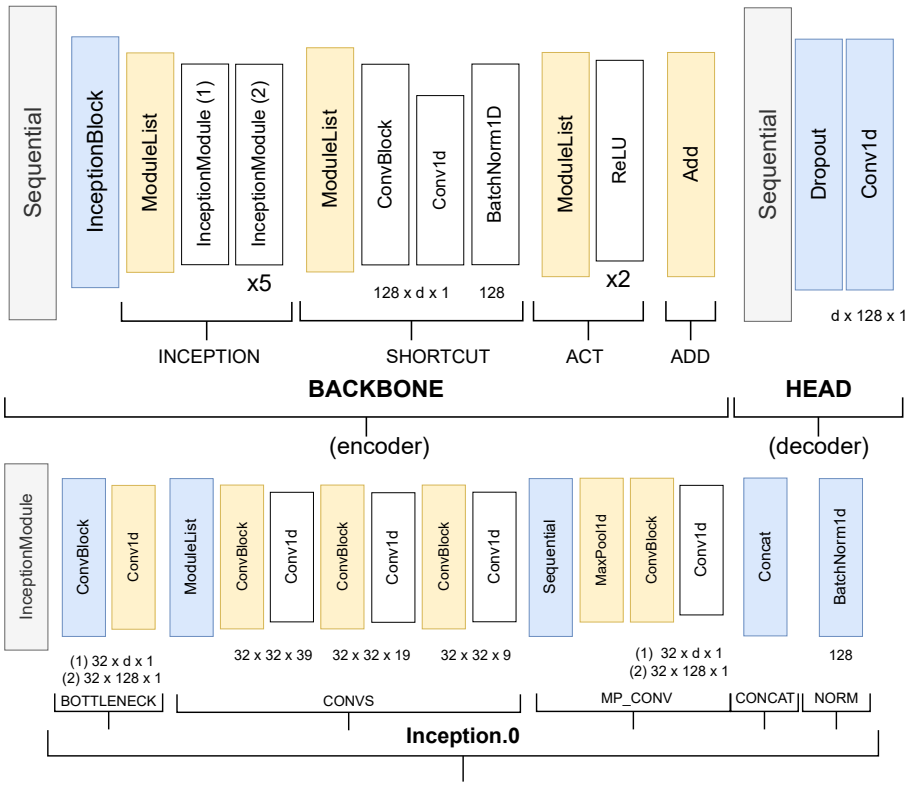



**Figure 3.21:** InceptionTimePlus block diagram. In the left, the global view of the model. In the right, the Inception.0 layer.

The InceptionTimePlus model is built by the following layers (see Figs. 3.21, 3.22, 3.23):

- `InceptionModulePlus`
  - The `bottleneck` layer transforms the input tensor $Y \in \mathbb{R}^{bs \times D \times ws}$ into $Z \in \mathbb{R}^{bs \times 32 \times ws}$, being $bs$ the batch size and $ws$ the sequence length, using a convolutional layer with kernel size 1, modifying the number of channels to 32 feature maps. Initially, $D = d$ (number of input features), as the input is the original TSs. In subsequent iterations $D = 128$, reflecting the number of channels in the previous layers.
  - The `convs` layer simultaneously applies to $Z$ three convolutional layers with kernel sizes of 39, 19 and 9 (to analyze different time window length). Each convolution produces and output of shape $(bs \times 32 \times ws)$, resulting in a total of $32 \times 3 = 96$ channels.
  - The `mp_comp` layer has a MaxPool1d with a kernel of size 3 and stride 1, followed by a convolutional layer with parameters $(32 \times D \times 1)$. The `MaxPool1d` operation reduces the time dimension of $Z$ in 2 timestamps while preserving its $D$ channels. This output is then passed to the convolutional layer, which transforms it into a representation of 32 channels.
  - The `concat` layer merges the outputs of `convs` and `mp_conv` l, forming a final sequence

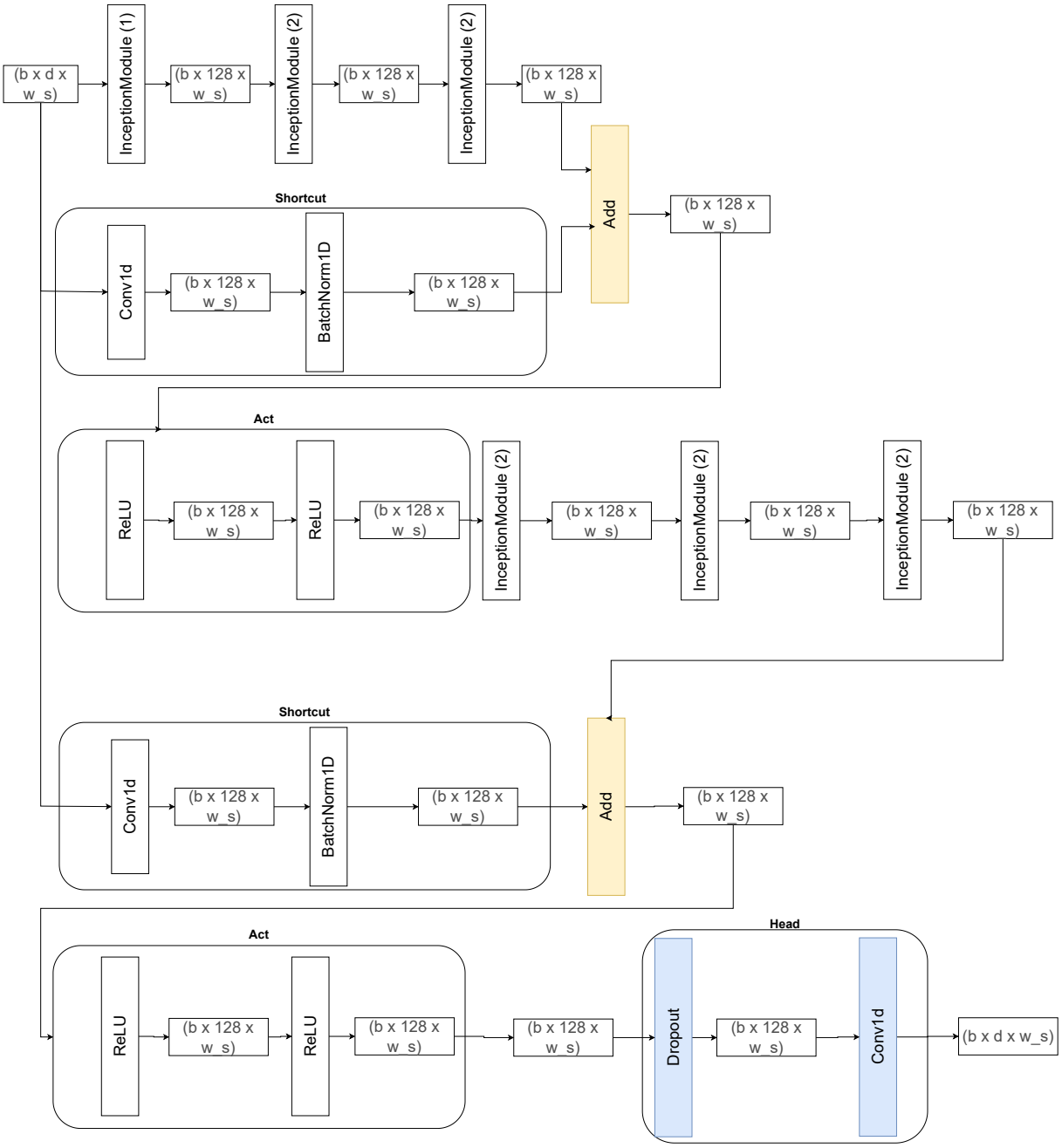



**Figure 3.22:** InceptionTimePlus global execution pipeline.

of shape a $(bs \times 128 \times ws)$ subsequently normalized using BatchNorm1d applied across the channel dimension.

- To add a residual connection, a `shortcut` and `add` layers are defined. Every three `InceptionModulePlus` blocks, the shortcut output is added to the output of `InceptionModulePlus` using the `add` layer. The result of this addition is then passed through the activation layer(`act`), that uses *ReLU* to enhance nonlinearity. This decision allows for the modelation of complex patterns and helps with gradient propagation by preventing

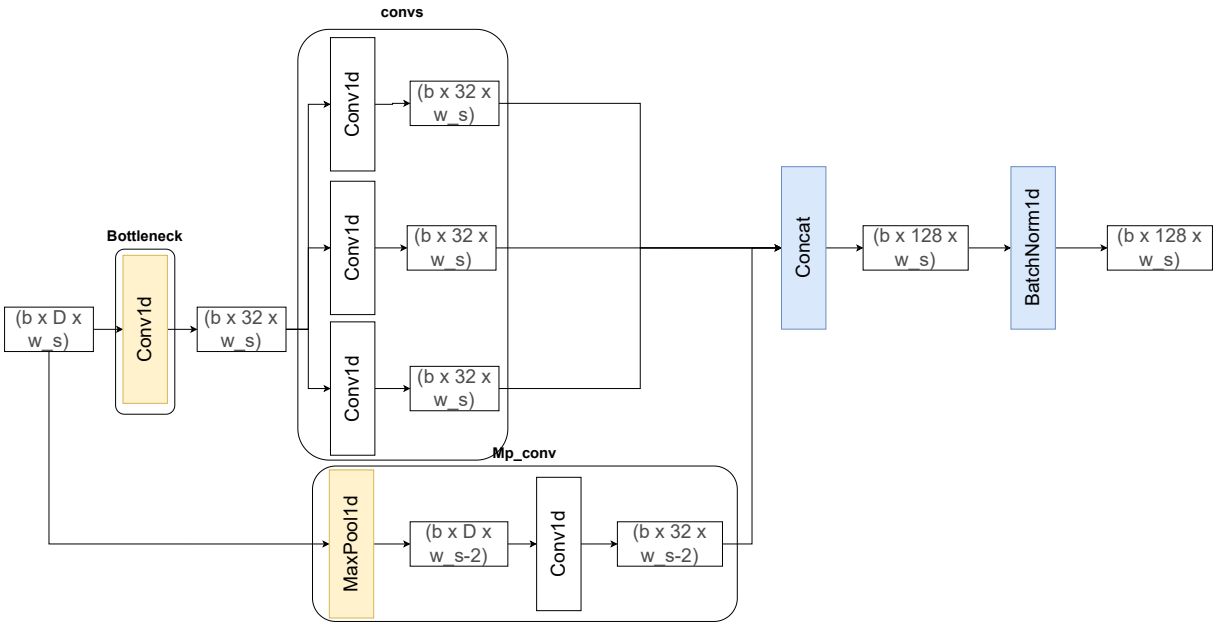



**Figure 3.23:** InceptionModule pipeline.

saturation (gradient near to 0), and avoiding extreme values, ensuring stable updates and more consistent learning. The `shortcut` connection is applied directly to the input tensor $Y$, ensuring an alternative path for the gradient flow. This shortcut consists of

  - A `Conv1d` layer with kernel size 1 that adjusts the number of channels from $d$ (the number of input features) to 128.
  - A `BatchNorm1d` layer to stabilize training.

- Head
  - As the output is not defined and its default value is 0.0, the `dropout` layer that is used to randomly deactivate some neurones to prevent overfitting, does nothing by default.
  - `Conv1d`. The final layer is a convolution that adjusts the number of channels from 128 (the output of the backbone) to $d$ (the original number of features of the TSs).

It is important to note that although the architecture is formally categorized as an encoder-decoder model, the decoder component is minimal: it consists of a simple 1D convolutional layer that maps the encoder's 128 output channels back to $d$, the original number of features in the input time series.

## 3.4 Scalability analysis of DeepVATS

The main objective of the present research is to have an improved DVA tool for analyzing TSs. For this, the DeepVATS application is selected as base. The main goal of DeepVATS is to analyze long TSs. As a first approach, an scalability analysis is done to know its efficiency, and then new DM and DL tools are added to enhance the application. This section examines the scalability

of DeepVATS on long Time Series, based on the analysis presented in Santamaria-Valenzuela et al. [89].

**Table 3.3:** Resamples of `Solar Power Dataset (4 Seconds Observations)` used for the scalability analysis. Seconds is the frequency generated by using the proposed frequency factor. The last column shows the number of elements of each dataset.

| Seconds | Frequency Factor | Number of Elements |
|---|---|---|
| 4 | 1 | 7,397,222 |
| 20 | 5 | 1,479,444 |
| 60 | 15 | 493,149 |
| 300 (5m) | 75 | 98,630 |
| 600 (10m) | 150 | 49,315 |

The Monash benchmark [277] presents more than 20 datasets with a different number of series and lengths, thus providing good data for testing the app's performance. To check the app's scalability and ensure its performance in large time series, a scalability analysis has been performed using the Monash benchmark [277] to analyse future improvements that can be carried out. To test the most challenging case, the dataset with the longest time series has been used: "Solar Power Dataset (4 Seconds Observations)" [273] (see Section 3.2).

To get representative subsets for scalability analysis, this dataset has been resampled using a frequency factor. Defining the frequency factor ($f_{factor} = 150$) as the number the frequency is multiplied by ($f = 4s$) so that the desired frequency is obtained ($f = 4s \cdot 150 = 600s = 10m$). For simplicity, after checking that the 10-minute frequency had also been previously used in the Monash benchmark [288], factors 5, 15, 75 and 150 have been selected to obtain divisions of frequencies of 4 seconds to 10 minutes. Table 3.3 shows the evolution in terms of the number of elements. To analyze the efficiency through execution times and logs, the following methodology has been carried out within the different datasets:

1. **Load the dataset from the feather file.** Select the desired dataset with the last associated encoder logged and check the time required to load the dataset from the binary file. The app will automatically get the embeddings from the encoder and apply UMAP using GPU for dimensionally reduction (DR) and generate the associated projections and time series plots.

2. **Change to PCA.** As will be detailed in the following sections, it is interesting to check the use of PCA and other DR algorithms.

3. **Change to CPU.** It is interesting to check the performance of the app according to this selection.

4. **Clustering.** To gain insight into the structure of the embeddings.

5. **Selection of projection points.** The TS points associated to the selected points will be shaded in the TS plot.

6. **Plots interactions.** To check the plots rendering after interactions, basic steps have been added: PP zoom in and zoom out, select a point on the time series plot so its associated projections points are remarked.

7. **PP aesthetics updates.** To check rendering times for aesthetic updates, minimal changes have been selected: update point alpha aesthetic in PP and removing lines in PP for clearer visualization.

This analysis has shown some performance problems. First, there is an issue of stability related to the use of `cuml`'s UMAP implementation [267] that is unstable due to an internal issue [3]. Two alternatives are proposed for future analysis. Second, the use of `reticulate` may add some time due to the communication between `R` and `Python`. Thus, some operations have been joined or moved to be used from a pickled file instead of R variables to get better performance. Finally, some extra execution steps have been detected due to `Shiny`'s reactiveness. This can be easily managed by a better use of the cache in the `R Shiny` app (the Vsual Module), thus some alternatives are proposed within the analysis.

In the following, the different steps that the application executes when applied are analyzed: load dataset, get encoder embeddings, compute projections, compute and visualize plots, and compute clusters.

**Load dataset.** TSP is generated using `dygraph` [289]. This function expects to receive a dataframe with timestamps as rownames. Thus, a rownames conversion was initially added to move the time index column to rownames. This step rapidly increases the execution time when increasing the TS length. After taking insights on the execution progress, a dataframe to `xts` conversion was detected when calling dygraph. Fig. 3.24 shows how explicitly making the conversion via `xts(TS_dataframe,order.by=TS_dataframe$timeindex)` reduces the execution time up to 27 seconds for the largest case, thus notably enhancing the app performance.

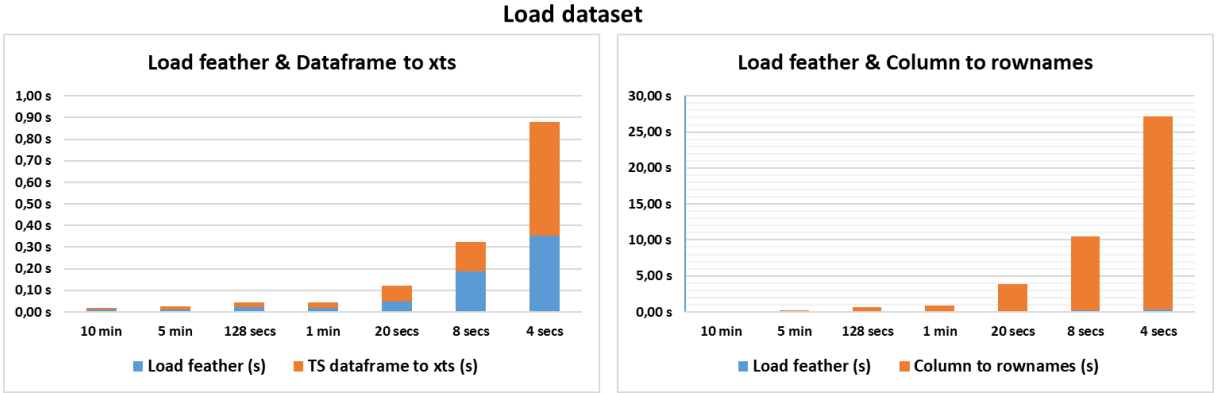



**Figure 3.24:** Load dataset aggregated time plot. In blue, read feather operation. In orange, on the left, time series dataframe conversion to xts. In orange, on the right, move the timeindex column to rownames operation.

**Get embeddings.** The time and memory consumption increase rapidly when the embeddings are obtained from the trained models (see 3.25). For memory handling and better `R-Python` communication, the implementation has been modified. Now, the dataset input is stored as a `feather` file, enabling data loading and manipulation within `Python`. Furthermore, obtaining embeddings is chunked so that GPU memory errors are avoided.

**Compute projections.** The computation of projections is known to be much faster on a GPU than on a CPU. However, there exists a known bug in the GPU version that makes UMAP

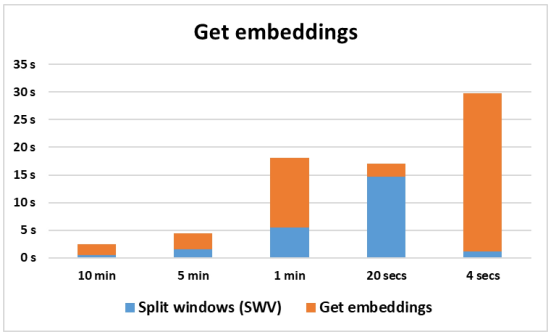



**Figure 3.25:** Aggregated plot for `get embeddings` step. In blue, the execution time consumed by the splitting window operation using Sliding Window View (in seconds). In orange, the execution time consumption for the get encoder embeddings operation (in seconds).

fail when using cuml UMAP GPU function that makes projections unstable and with low-quality cluster distribution [3, 290]. This quality can be checked in the app logs, where `cluster_score(prjs, clusters_labels) = silhouette_score(prjs, clusters_labels)` is used to obtain a value in $[-1,1]$, with $-1$ being the worst score. To avoid this problem, the CPU version can be selected, but this results in an excessive execution time for a visual dynamic application. Thus, there exist two possible work lines. The first option is to change the GPU UMAP library, to use, for example, Peter Eisenmann's parallel UMAP function [2]. The second option is to explore the solution proposed by Corey J. Nolet: Execute UMAP followed by PCA to obtain better results [3]. As Fig. 3.26 shows, the time would still be much better with this option than with the CPU. The final option selected has been using UMAP followed by PCA, faccilitating posterior analysis.

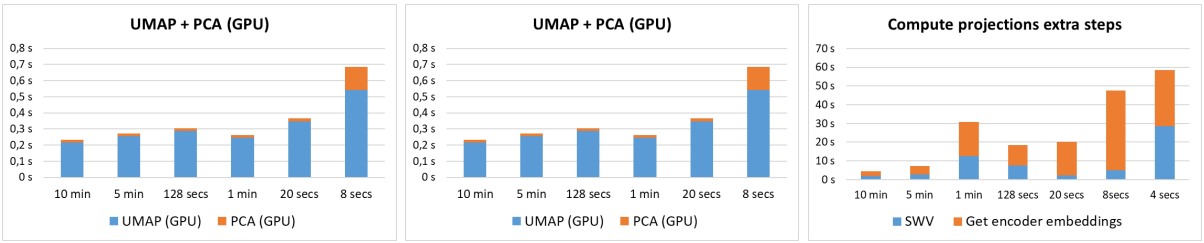



**Figure 3.26:** Compute projections. On the left UMAP followed by PCA (GPU) execution time aggregated plot. On the middle, the comparison with UMAP (CPU) executions. On the right, the execution time associated to extra executed steps.

**Compute and visualize plots.** A great reduction in performance was detected when trying to load the `Solar 4 Seconds` dataset and interact with the plots. This is reasonable as it is the largest one. The problem did not appear for small datasets, but it causes the app to crash with large datasets. Thus, this scalability analysis was conducted, where some reactive operations were detected to be executed without need (see Table 3.4). This adds a lot of time to the rendering of the plots. Also, a notable difference was noticed between execution and rendering time (see Table 3.5), allowing nearly 4 minutes to render the plots for the 4-second frequency dataset. To avoid this problem, shiny can cache plots to avoid repeated calculations. Also, the reduction of the number of shown points has reduced the renderization time. Therefore, the next step of the work is to analyze the use of more `reactiveVal`, and cache plots [291], and to check more options for the reduction of the plotted points.

**Table 3.4:** TSP and PP computation time media and extra detected steps execution time in seconds.

| | Time (seconds) | | | | | Time (% Relative to total) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Computation time of** | **10m** | **5m** | **1m** | **20s** | **4s** | **%10m** | **%5m** | **%1m** | **%20s** | **%4s** |
| TSP | 0.03 | 0.03 | 0.08 | 5.45 | 25.37 | 0.63 | 0.39 | 0.26 | 23.23 | 46.27 |
| PP | 2.16 | 3.28 | 12.46 | 0.93 | 0.05 | 45.92 | 42.65 | 40.71 | 3.97 | 0.09 |
| Extra steps | 2.51 | 4.38 | 17.06 | 29.4 | 29.83 | 53.45 | 56.95 | 59.03 | 72.78 | 53.63 |

**Table 3.5:** TSP and PP plot rendering time after computing projections.

| | Time (seconds) | | | | |
|---|---|---|---|---|---|
| **Rendering time of** | **10m** | **5m** | **1m** | **20s** | **4s** |
| TSP | 4.11 | 5.25 | 20.48 | 79.19 | 228.05 ≈ 4m |
| PP | 5.55 | 7.24 | 21.16 | 64.80 | 228.10 ≈ 4m |

**Compute clusters.** The execution of the embedding space clusters is very fast even for large time series. However, some reactive functions have also been detected that affect the performance of the clustering step. Thus, cache analysis is also useful to improve this step.

### 3.4.1 Improving the scalability of DeepVATS

The scalability analysis performed has shown that DeepVATS has excellent performance for small datasets (ranging from 49.3K to 98.6K elements) and medium datasets (up to 493.1K elements). However, performance issues appeared when processing larger datasets, with noticeable degradation at 3.7M elements and application crashes at 7.4M elements.

To enhance the application's usability, different possible solutions are detected:

- Enhancing the code using cached values and batching computations in large datasets, supposing a great change in the interaction times.
- Stability of the embedding projection plot. To ensure high-quality dimensionality reduction, two strategies are suggested: adopt an alternative implementation of UMAP [2], and explore the application of PCA followed by UMAP, rather than UMAP alone [3]. The second option, using PCA followed by UMAP, has been taken, which facilitates reproducibility.
- Adding other state-of-the-art tools that may add more explainability in less execution time.

In the next sections, the focus is on the third line. Two different tools are analyzed. First, MPlot, an already optimized data mining tool that adds a fast preview of different properties of the time series. Second, the Time Series Foundation models, which has been adapted to time series in recent years and has shown good performance in the different tasks (classification, anomaly detection, etc.), leading to its widespread adoption in the field. In this research, the backbone model (MTSAE) is changed to test if the fast application of a generally pre-trained model to new cases is real for our datasets in terms of the explainability of the projections plot

as it would suppose a great time saving as the analyst would not need to train the model for each case or just train it for a really small part of the new dataset (fine-tuning).

## 3.5 Introducing Distance Matrix Plot

The work presented in this thesis aims to add an interactive exploration of MPlot integrated into DeepVATS. This integration allows to analyze the MPlot for a dataset while waiting for the MTSAE model to be trained with it and so get two different perspectives for the analysis. MPlot is a plot based on the distances between the subsequences of a TSs. Next sections introduce the different definitions necessary for the comprehension and implementation of the plot, following [11] and analyze the `PulsusParadoxus` dataset.

### 3.5.1 Basic definitions

Definition 3.5.1 formalizes the concept of **subsequence**. See the example subsequence starts at the position $k$ with a length of $m$ time steps. This value ($m$) is called **subsequence length**. This will be notated as $X^n \sim n$, $X^{(k,m)} \sim m$. The subsequences will be called **window** and so, subsequence length will also be called **window length** following the analogous definition of MTSAE.

**Definition 3.5.1: Time Series subsequence**

Given a TSs $X^n = \{x_0, x_1, \ldots, x_n\} \subset \mathbb{R}^{d \times n}$. A **subsequence** is an ordered subset of $X$: $X^{(k,m)} = \{x_i\}_{i=k}^{k+m-1} \subset \mathbb{R}^{d \times m}$, following the same order as the original TSs.

Now, the distance between two subsequences of $X$ of the same length can be defined by using them as vectors: $d(X^{(k,m)}, X^{(l,m)}) = d((x_i)_{i=k}^{k+m-1}, (x_i)_{i=l}^{l+m-1})$. The most widely used distance is the **z-normalized Euclidean** distance [292].

The **distance profile** (Def. 3.5.2) and the **distance matrix**( 3.5.3) or **similarity matrix** are also defined based on the distances of the subsequences of size "window length".

**Definition 3.5.2: Distance Profile**

Given $X_A \sim n_A$, $X_B$, the **distance profile** is the vector of distances between each subsequence in $X_A$ and $X_B^{(k,m)}$:

$$DP_{A,B}^{(k,m)} = (d(X_A^{(i,m)}, X_B^{(k,m)}))_{i=0}^{n_A-m+1}.$$

**Definition 3.5.3: Similarity Matrix**

Given $X_A \sim n_A$, $X_B \sim n_B$. The **similarity matrix** or **distance matrix** is the matrix $DM_{A,B} \in \mathbb{R}^{n\_rows \times n\_cols}$ where each row is a distance profile with the corresponding $X_B$ subsequence. That is,

$$DM_{A_B} = (DP_{A,B}^{(k,m)})_{k=0}^{n_B-m+1} = ((d(X_A^{(i,m)}, X_B^{(k,m)}))_{i=0}^{n_A-m+1})_{k=0}^{n_B-m+1},$$

with $n\_rows = n_A - m + 1$ and $n\_cols = n_B - m + 1$.

The Similarity Matrix plot or MPlot is the plot resulting from visualizing in a heatmap mode that matrix (see fig. 3.27). This matrix computation can incur a very large memory and time cost. Three ways can be followed to make it more efficient. One possibility is to reduce the