Universidad de la República

Facultad de Ingeniería

Aprendizaje Automático para Datos en Grafos

GraGOD: Aplicación de modelos de grafos para detección de anomalías en series temporales

Autores:
Federico Bello
Gonzalo Chiarlone

28 de septiembre de 2025

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

1. Introducción

En este informe se exploran técnicas de aprendizaje automático basadas en estructuras de grafos como una herramienta para la detección de anomalías en series temporales. Se discuten distintos modelos basados en recientes estudios los cuales hacen uso de estas técnicas, buscando compararlos y encontrar las ventajas y desventajas de cada uno.

El objetivo final es utilizar este conocimiento para aplicarlo a un conjunto de datos especifico (TELCO), comparando el resultado con enfoques mas clásicos.

2. Marco Teorico

2.1. GNNs

Los grafos son una estructura de modelado matemático los cuales son utilizados en una gran cantidad de aplicaciones, desde el modelado de redes sociales, redes de telecomunicaciones o las interconexiones entre bacterias. Surge natural entonces preguntarse si, dado un problema modelado como un grafo y una gran cantidad de datos sobre este, es posible realizar predicciones y detectar patrones sobre el mismo utilizando técnicas ya conocidas de aprendizaje automático.

Planteando el problema de forma mas concreta, se tiene un grafo G con un conjunto V de nodos ($|V| = N$), otro conjunto E de aristas y una señal en cada nodo. Con esto, se define la matriz de adyacencia $\mathbf{A}$ del grafo y la matriz con la señal $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$. Luego, el objetivo sera predecir alguna propiedad del grafo, es decir, encontrar alguna función $\phi(\mathbf{A}, \mathbf{X})$. que minimice una función de perdida impuesta.

Para entender las *Graph Attention Networks* (GNNs) es primero necesario definir la convolución a nivel de grafo, la cual obtiene una combinación lineal de versiones desplazadas de la señal:

$$\mathbf{X} *_{\mathbf{S}} \mathbf{H} = \sum_{k=0}^{K-1} \mathbf{S}^k \mathbf{X} \mathbf{H}_k = \mathbf{H}(\mathbf{S})\mathbf{X}$$

donde los h_k son coeficientes a aprender y $\mathbf{S}$ es denominado el *Graph Shift Operator* (GSO), donde $\mathbf{S}^k$ tiene la propiedad que la entrada i, j es distinta a 0 sii existe un camino de largo k entre i y j. Por lo tanto, la convolución no es mas que una combinación lineal entre todas las señales a una distancia como máximo k del nodo, ponderados por los coeficientes de $\mathbf{H}_k$. Aquí ya se ve una posible mejora: la importancia que se les da a los nodos solo depende de la distancia y de la cantidad de caminos que haya uniéndolos.

Luego, para formar una GNN no hay que hacer mas que apilar convoluciones de grafo. Sin embargo, observar como las convoluciones son funciones lineales, por lo que para aumentar la expresividad se agrega una no linealidad entre cada convolución. La figura 1 muestra la arquitectura de una GNN de L capas. Por ultimo, la salida de la ultima capa $\mathbf{X}_L$ se compara con la señal observada para calcular el valor de la función de costo y así realizar la actualización de los parámetros del modelo.

2.2. GATs

Como se menciono anteriormente, las modelo como esta presentado tiene la desventaja de darle el mismo peso a todos los vecinos a una cierta distancia. Esto no necesariamente es así, dado que es probable que algunos vecinos sean mas informativos que otros. Este problema es el que apunta a solucionar las *Graph Attention Networks* (GATs), modelo presentado en [8]. Las GATs aplican el ya conocido concepto de *Attention* en las GNNs de la siguiente forma:

$$\mathbf{x}_i = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{H} \mathbf{x}_j)$$

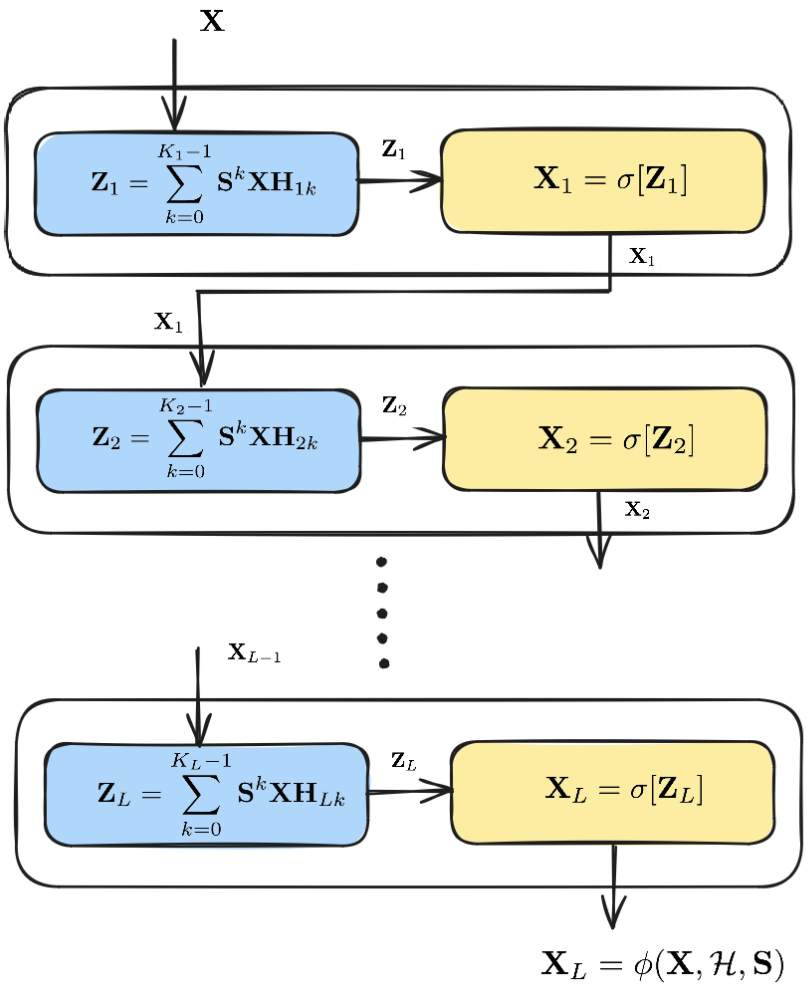

Figura 1: Arquitectura de una GNN de L capas

donde los pesos α_{ij} son los pesos de atención a aprender. El conjunto $\mathcal{N}_i$ es denominado el vecindario de i, y es el conjunto de todos los nodos de hasta distancia k, donde k es un parámetro del modelo.

Para calcular los coeficientes de atención, se utilizan los *embeddings* de los vectores i y j:

$$e(\mathbf{x}_i, \mathbf{x}_j) = \text{LeakyReLU}(\mathbf{a}^T \cdot [\mathbf{H}\mathbf{x}_i || \mathbf{H}\mathbf{x}_j]) \tag{1}$$

donde $||$ representa la operación de concatenación y la matriz $\mathbf{a} \in \mathcal{R}^{2d'}$ es un nuevo parámetro a aprender, con d' la dimensión de salida de cada *embedding*. Luego, para facilitar la comparación de los *embeddings* en los distintos nodos, se normaliza la función de *scoring*, obteniendo así los coeficientes de atención.:

$$\alpha_{ij} = \text{softmax}_j(e(\mathbf{x}_i, \mathbf{x}_j)) = \frac{\exp(e(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{j' \in N_i} \exp(e(\mathbf{x}_i, \mathbf{x}'_j))}$$

El estudio prosigue con una mejora del modelo siguiendo las ideas planteada en la sección 3.2.2 de [7], utilizando M mecanismos de atención independientes: $\boldsymbol{\alpha}_{ij} = (\alpha^1_{ij}, \alpha^2_{ij}, ..., \alpha^M_{ij})$. Finalmente, las salidas de estos mecanismos de atención se concatenan para formar la salida:

$$\mathbf{x}_i = \Big\Vert_{m=1}^{M} \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha^m_{ij} \mathbf{H}^m \mathbf{x}_j\Big)$$

Como un ultimo comentario, observar como en la ecuación 5 si uno de los valores de $\mathbf{a}^T\mathbf{H}\mathbf{x}_j$ es mucho mas grande que el resto podría opacarlos, eventualmente ignorando el nodo original. Esto genera una reducción en la expresividad del modelo en algunos casos, problema que estudian en [1], donde llegan a la conclusión que el simple hecho de multiplicar por $\mathbf{a}^T$ luego de aplicar la no linealidad mejora de forma considerable el resultado, a esto se le denomina GATv2. Por lo tanto, una nueva posible función de scoring para el modelo vendría dada por la forma:

$$e(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{a}^T \cdot \text{LeakyReLU}([\mathbf{H}\mathbf{x}_i || \mathbf{H}\mathbf{x}_j])$$

2.3. GNNs y GATs para detección de anomalías

Debido a los grandes avances en tecnologías de detección y procesamiento de flujos de datos, como pueden ser las comunicaciones inalámbricas, la cantidad de datos en forma de series temporales (TS) a aumentado abundantemente en los últimos años [4]. Con este contexto en mente, las GNNs han surgido como una herramienta poderosa para aprender representaciones de datos temporales con interrelaciones complejas. La estructura de grafo permite la captura de relaciones entre las distintas TS, así también como intertemporales (dependencias entre diferentes puntos en el tiempo).

A continuación, se utilizan las clasificaciones realizadas en [4]. De forma general, en un problema de detección de anomalías en TS se tiene un conjunto de series temporales $s = [s_1, s_2, ..., s_N]$, donde $s_i \in \mathbb{R}^T$, es decir, se tienen N series de T muestras. Normalmente, se busca uno de los siguientes objetivos:

- Predecir anomalías en cada serie: En este caso, el algoritmo tiene como salida un conjunto $y \in \mathbb{R}^{N \times T}$ de etiquetas binarias donde $y_{i,t} = 1$ si la i-esima serie presenta un valor anómalo en el instante t y $y_{i,t} = 0$ en el caso contrario.
- Predecir anomalías en una de las series: Este caso es similar al anterior, solo que tiene como objetivo una única serie. En este caso, la salida es $y \in \mathbb{R}^T$.
- Predecir anomalías en un sistema: Por ultimo, se puede buscar predecir anomalías en un sistema formado por múltiples series temporales (por ejemplo, predecir fallas en un dispositivo electrónico dadas medidas de múltiples componentes del mismo). En este caso la salida también es $y \in \mathbb{R}^T$.

Los modelos de detección de anomalías se basan en medidas de discrepancia entre los datos reales y algún tipo de estimación de la misma, si la discrepancia es alta se da el dato como anómalo. Las distintas formas de estimar una se pueden dividir en tres grandes grupos (no disjuntos):

- **Discrepancia de reconstrucción:** se basa en la idea de que los errores de reconstrucción deberían ser bajos durante períodos normales y altos durante períodos anómalos. Se emplean marcos de trabajo, como *autoencoders*, que buscan replicar las entradas como salidas. Se asume que el modelo base es lo suficientemente expresivo para modelar y reconstruir bien la distribución de datos de entrenamiento, pero no datos fuera de muestra.
- **Discrepancia de predicción:** similar al modelo de reconstrucción, solo que se en vez de reconstruir los valores se intenta predecir el siguiente, es decir, utilizando los valores entre t_1 y t_2, con $t_2 > t_1$, se intenta predecir los valores de las TS en $t_2 + 1$. Igual que en el caso anterior, se espera un error de predicción mas grande en muestras anómalas.
- **Discrepancia relacional:** se basa en la idea de que las relaciones entre variables deben experimentar cambios significativos durante los períodos anómalos en comparación con los normales. En este enfoque,

se utiliza un modelo de aprendizaje de grafos para construir la evolución de las relaciones entre variables. Luego, se analizan los cambios en estas relaciones y asigna una puntuación de anomalía o discrepancia.

Se ve claramente como las tres categorías se basan en la misma idea de realizar una estimación para luego calcular el error en la estimación, solo que difieren en como realizan esta estimación. Además, como se menciono anteriormente, las categorías no son disjuntas. Existen modelos híbridos los cuales tienen componentes de dos o mas de las categorías, los cuales tienen el objetivo de aprovechar al máximo la información de los datos.

Finalmente, un detalle no menor. Todos los modelos se basan en que el conjunto de entrenamiento tiene o pocas o ninguna anomalía. Si esto no fuera así, se aprendería el comportamiento de las anomalías como un comportamiento 'normal', impidiendo la detección de las mismas. Es por esto que es de suma importancia preprocesar correctamente los datos, eventualmente sustituyendo los datos que estén clasificados como anómalos.

3. Modelos

A continuación, se incluye una explicación de los modelos explorados.

3.1. MTAD-GAT

El primer modelo a explorar es MTAD-GAT (*Multivariate Time-series Anomaly Detection via Graph Attention Network*), presentado en [9]. Este modelo usa tanto la discrepancia de reconstruccion como la de prediccion, generando un *score* en el cual se tienen presentes ambas cosas. El modelo toma como entrada una matriz $V \in R^{w \times n}$ donde w es el tamaño de la *sliding window* (ya que no se entrena con todas las muestras temporales al mismo tiempo) y n es la cantidad de series temporales. La arquitectura del modelo, representada en 2 (obtenida de [9]), incluye las siguientes componentes:

- *1-D convolution*: Convoluciones de una dimensión sobre cada una de las señales, para extraer características a alto nivel de cada una. Esta técnica es muy común en cualquier modelo de detección de anomalías.
- *Feature-oriented GAT*: Por un lado, se busca captar las relaciones entre series temporales, utilizando una GAT. En este caso, cada nodo esta formado por cada una de las series temporales, es decir, el grafo esta formado por los nodos $V = [v_1, v_2, ..., v_n]$, donde $v_i \in R^w$. De esta manera, las layers de *attention* se encargaran de captar las relaciones entre nodos. En este caso, se tomo el grafo completo, por lo que este modelo resulta muy parecido a aplicar un Transformer entre series temporales.
- *Time-oriented GAT*: Por otro lado, se busca captar las dependencias temporales de cada serie, otra vez, con una GAT. En este caso, se toma un nodo por cada tiempo en w, donde cada nodo esta formado por los valores en las distintas series temporales. Esto quiere decir: $V = [v_1, v_2, ..., v_w]$, donde $v_i \in R^n$, siendo n el numero de series temporales. De vuelta, dado que se supone un grafo completo, este modelado resulta muy parecido al de un Transformer.
- *GRU*: Se concatenan los tres resultados anteriores y se usan como entrada de una *Gated Recurrent Unit* (GRU) de dimensión oculta d_1. Este modelo es un tipo de RNN muy popular, basado en LSTM, el cual en este caso se utiliza para capturar mejor las relaciones temporales de las series.
- *Forecasting model*: El modelo encargado de realizar la predicción a futuro, para el cual se utiliza solamente una *fully-connected*.

- *Reconstruction model*: El modelo encargado de realizar una reconstruccion, el cual se implemnta con un VAE (*Variational AutoEncoder)*. En nuestro caso, este modelo fue reemplazado por una GRU.

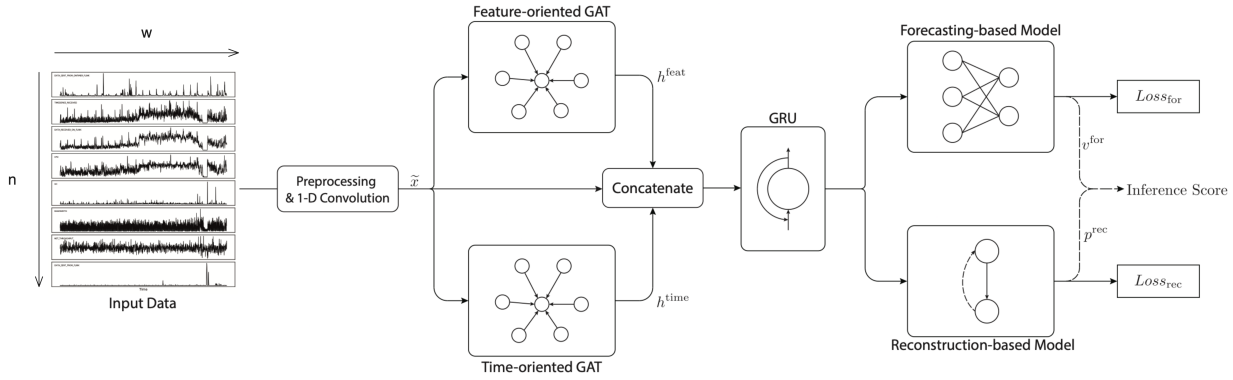

Figura 2: Arquitectura de MTAD-GAT

Como se puede ver en la figura 2 (obtenida de [**?**], la matriz de series temporales de tamaño $w \times n$ es pasada por la *1d-convolution*, cuya salida es procesada en paralelo tanto por la *feature-oriented GAT* como por la *time-oriented GAT*. Luego, estas tres salidas son concatenadas, formando una matriz de $w \times 3n$. Esta matriz es luego pasada por la GRU, dando como salida una matriz de $w \times d_1$. Por ultimo, esta matriz es la que ingresa tanto al modelo de *forecasting* como al de *reconstruction*.

Este modelo se entrena para optimizar una *Loss* compuesta, la cual tiene la siguiente forma:

$$Loss = Loss_{forecasting} + Loss_{reconstruction}$$

1) *Forecasting loss*: Cuando el modelo intenta predecir el siguiente valor de una serie temporal, simplemente se utiliza una *Root Mean Square Error (RMSE)*:

$$Loss_{forecasting} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

2) *Reconstruction loss*: Si bien el paper original utiliza otra loss, debido a que en la implementacion que utilizamos se reemplazo el VAE con una GRU, simplemente se vuelve a utilizas RMSE.

Por ultimo, se define una funcion de Scoring S. Esta funcion da un puntaje de que tan probable es que un dato sea una anomalia. Se utiliza la siguiente formula:

$$S_i = \frac{(\hat{x_i} - x_i)^2 + \gamma \times (1 - p_i)}{1 + \gamma}$$

done x_i es el valor real de la i-esima serie, $\hat{x_i}$ el valor predecido p_i representa la probabilidad de haber realizado una buena construccion. En nuestro caso, no se tiene esta probabilidad, por lo que se cambia la funcion por la siguiente:

$$S_i = \frac{\sqrt{(\hat{x_i} - x_i)^2} + \gamma\sqrt{(\tilde{x_i} - x_i)^2}}{1 + \gamma}$$

donde $\tilde{x_i}$ es la reconstrucción de la i-esima serie. Por ultimo, gamma es solo un hiperparametro a optimizar.

3.2. GDN

El siguiente modelo estudiado es *Graph Deviation Network* (GDN) presentado en [2]. Este modelo cae dentro de una categoría híbrida, donde se aprende la estructura del grafo mediante las relaciones entre las TS y se realiza la estimación de la señal mediante una predicción temporal. El modelo se basa en cuatro componentes:

- Un *Embedding* que representa a cada señal, los cuales se inicializan como un vector dimensión d de forma aleatoria. El mismo es aprendido durante el entre entrenamiento, y el objetivo es que nodos con comportamientos similares tengan *embeddings* similares.

- Mediante relaciones entre las TS, aprende la estructura del grafo, donde para cada nodo i se tiene su vector de *embedding* $\mathbf{v}_i$ y un conjunto de nodos candidatos: $C_i \subseteq \{1, 2, ..., N\} \backslash \{i\}$. Si no se tiene información previa sobre el grafo, ese conjunto es el conjunto de todos los nodos. Luego, se calcula: $e_{ji} = \frac{\mathbf{v}_i^T \mathbf{v}_j}{||\mathbf{v}_i|| ||\mathbf{v}_j||} \quad \forall\ j \in C_j$, eligiendo los K valores mas altos para formar la matriz de adyacencia. Observar que el grafo formado es dirigido, ya que los conjuntos C_i no tienen por que ser simétricos, así como K puede ser un numero impar.

- Predicción temporal para realizar la estimación, la cual utiliza la señal en instantes anteriores y las señales de los vecinos en el grafo aprendido.

  Formalmente, la entrada al modelo es una ventana de tiempo $\mathbf{x}^{(t)} \in \mathbb{R}^{N \times w}$ la cual tiene los datos de las N series en los w instantes de tiempo anteriores, y el objetivo es predecir el instante actual.

  Luego, se utiliza un extractor de características basado *graph attention*, donde las únicas diferencias con el explicado en la sección 2.2 es que a la entrada de la función de scoring e se le concatena el *embedding* del primer paso: $(\mathbf{v}_i \,\|\, \mathbf{H}\mathbf{x}_i)$. Además, se calcula la representación del nodo i como:

$$\mathbf{z}_i^{(t)} = \text{ReLU}\left(\alpha_{i,i}\mathbf{W}\mathbf{x}_i^{(t)} + \sum_{j \in N(i)} \alpha_{i,j}\mathbf{W}\mathbf{x}_j^{(t)}\right)$$

  Con esto y con los vectores de *embedding* se calcula la salida en el tiempo t $\hat{\mathbf{s}}^{(t)}$ como:

$$\hat{\mathbf{s}} = f_\theta(\mathbf{v}_1 \circ \mathbf{z}_1^{(t)}, \ldots, \mathbf{v}_N \circ \mathbf{z}_N^{(t)})$$

  Por ultimo, como función de costo a minimizar se utiliza el error cuadrático medio entre la salida predicha $\hat{\mathbf{s}}^{(t)}$ y los datos observados $\mathbf{s}^{(t)}$:

$$L_{MSE} = \frac{1}{T_{\text{train}} - w} \sum_{t=w+1}^{T_{\text{train}}} ||\hat{\mathbf{s}}^{(t)} - \mathbf{s}^{(t)}||_2^2$$

  donde T_{train} es el número de muestras de entrenamiento.

- *Scorer*, el cual identifica las desviaciones entre las predicciones y los datos. El puntaje de anomalía compara el comportamiento esperado en el tiempo t con el comportamiento observado, calculando la diferencia entre la salida observada y la predicha.

  Con esto, se obtienen distintos valores de *score* para cada TS. El estudio de referencia prosigue con una estrategia para inferir si hay una anomalía en todo el sistema. Sin embargo, en este caso es de interés detectar cada serie por separado, por lo que para cumplir con el objetivo es suficiente fijar un *threshold* para cada TS, si el valor de *Err* es superior a este se da como anomalía.

4. Dataset

Este informe utiliza el TELCO dataset, el cual se basa en mediciones reales recopiladas por un ISP móvil. Se compone de doce TS que abarcan siete meses, proporcionando una perspectiva única para el análisis de comportamientos estacionales a largo plazo.

Cada serie temporal refleja datos agregados de diversas fuentes, como tarifas de transferencia de datos prepago, llamadas y costos, volumen de tráfico de datos, número de SMS, entre otros. Para preservar la confidencialidad empresarial, no se especifica el tipo exacto de datos representados por cada serie temporal. El conjunto de datos es separado; como es usual; en tres subconjuntos: entrenamiento, validación y prueba, los cuales están ordenados temporalmente y tienen muestras correspondientes a 3 meses, 1 mes y 3 meses respectivamente.

La figura 3 muestra el fuerte desbalance del conjunto de datos: la cantidad de muestras anómalas no supera el 3 % de los datos totales, situación usual en escenarios de detección de anomalías. Esto no es un problema al momento del entrenamiento, ya que para atacar este tipo de problemas se suelen utilizar algoritmos no supervisados. Sin embargo, si es necesario ser inteligentes con la métrica de desempeño a utilizar; un *naive classifier* el cual nunca predice una anomalía tendrá cerca de un 98 % de *accuracy*.

	training			**validation**			**testing**			**total**		
TS ID	normal	anomalous	%	normal	anomalous	%	normal	anomalous	%	normal	anomalous	%
TS_1	24,731	1,183	4,6%	8,628	12	0,14%	26,084	412	1,6%	59443	1607	**2,6%**
TS_2	25,713	201	0,8%	8,629	11	0,13%	25,995	501	1,9%	60,337	713	**1,2%**
TS_3	25,784	130	0,5%	8,636	4	0,05%	26,358	138	0,5%	60,778	272	**0,4%**
TS_4	24,464	1,450	5,6%	8,636	4	0,05%	26,317	179	0,7%	59,417	1,633	**2,7%**
TS_5	25,840	74	0,3%	8,637	3	0,03%	26,390	106	0,4%	60,867	183	**0,3%**
TS_6	25,850	64	0,2%	8,639	1	0,01%	26,390	107	0,4%	60,879	172	**0,3%**
TS_7	25,793	127	0,5%	8,638	2	0,02%	26,227	269	1,0%	60658	398	**0,7%**
TS_8	25,787	127	0,5%	8,640	0	–	26,229	267	1,0%	60,656	394	**0,6%**
TS_9	25,287	627	2,4%	8,508	132	1,53%	25,932	564	2,1%	59,727	1,323	**2,2%**
TS_{10}	24,558	1,356	5,2%	8,463	177	2,05%	25,995	501	1,9%	59016	2,034	**3,3%**
TS_{11}	25,725	189	0,7%	8,601	39	0,45%	26,475	21	0,1%	60,801	249	**0,4%**
TS_{12}	25,770	144	0,6%	8,640	0	–	26,481	15	0,1%	60,891	159	**0,3%**

Figura 3: Proporciones de datos normales/anómalos en los distintos conjuntos de datos, imagen extraída de [3]

5. Experimentos

El cuadro 1 muestra los resultados de los experimentos realizados (modelo GDN y MTAD-GAT) en comparación con DC-VAE, modelo presentado en [3], y con el metodo de *Seasonal Exponential Smoothing* (S-EXPS). Los resultados correspondientes a estos utlimos dos experimentos fue extraido de [3].

En todos los experimentos se realizo un preprocesamiento básico de los datos, donde las anomalías en el conjunto de entrenamiento fueron eliminadas para luego realizar una interpolación de los datos faltantes. Este cambio mostró un impacto grande en los resultados, el cual suponemos que se debe a que de no sustituir los datos anómalos, el modelo logra a aprender a predecir los mismos. Es importante notar que este preprocesamiento, por mas simple que sea, convierte los modelos en modelos supervisado en vez de no supervisado, por lo que reduce las posibles aplicaciones del mismo. Sin embargo, se ve como el resultado básico de la arquitectura GDN alcanza buenos resultados.

Los experimentos fueron realizados con las implementaciones provistas en [6] y [5]. Luego de obtener un primer resultado con cada uno de los modelos, se opto por modificar los modelos levemente, buscando una mejora en el resultado aplicando conceptos teóricos.

	S-EXPS			DC-VAE			GDN			MTAD-GAT		
TS ID	R_r	P_r	$F1_r$	R_r	P_r	$F1_r$	R_r	P_r	$F1_r$	R_r	P_r	$F1_r$
TS1	45 %	88 %	60 %	58 %	71 %	64 %	25 %	38 %	30 %	52 %	80 %	63 %
TS2	70 %	96 %	81 %	74 %	20 %	67 %	20 %	75 %	32 %	34 %	5 %	9 %
TS3	78 %	58 %	67 %	86 %	47 %	60 %	75 %	94 %	83 %	31 %	2 %	4 %
TS4	67 %	41 %	51 %	63 %	21 %	32 %	70 %	97 %	81 %	55 %	56 %	55 %
TS5	73 %	63 %	53 %	75 %	50 %	60 %	68 %	99 %	80 %	42 %	1 %	3 %
TS6	63 %	63 %	63 %	57 %	83 %	68 %	62 %	99 %	76 %	50 %	2 %	3 %
TS7	69 %	53 %	60 %	72 %	90 %	80 %	82 %	100 %	90 %	58 %	47 %	52 %
TS8	56 %	36 %	43 %	44 %	80 %	57 %	81 %	91 %	86 %	72 %	72 %	72 %
TS9	5 %	5 %	5 %	17 %	4 %	11 %	13 %	27 %	17 %	37 %	5 %	9 %
TS10	48 %	44 %	46 %	52 %	59 %	55 %	79 %	92 %	87 %	43 %	26 %	32 %
TS11	50 %	32 %	39 %	100 %	25 %	40 %	0 %	0 %	0 %	25 %	5 %	8 %
TS12	100 %	67 %	80 %	100 %	11 %	22 %	0 %	0 %	0 %	22 %	9 %	17 %
Media	58 %	54 %	54 %	67 %	47 %	52 %	52 %	67 %	**56** %	43 %	18 %	27 %
Mediana	60 %	55 %	57 %	68 %	49 %	59 %	65 %	92 %	**78** %	47 %	22 %	32 %

Cuadro 1: *Recall* (R_r), *precision* (P_r) y *F1-score* ($F1_r$) de distintos modelos en TELCO

La primer modificación a probar fue utilizar una celda GATv2 en lugar de una GAT en el modelo de GDN. Con esto se esperaba una mejora en la expresividad del modelo, al igual que afirma en [1]. Sin embargo, la modificación no produce ningún impacto positivo, incluso disminuyendo levemente el valor medio del *F1-score*.

La siguiente modificación fue aplicar una inferencia de un grafo en el modelo de MTAD-GAT. El grafo utilizado fue el inferido por el modelo GDN, el cual puede observarse en la figura 4. Para inferir el mismo, se utilizo una máxima cantidad de vecinos por nodo de 6, una ventana de largo 300 $time - stamps$ y $learningrate$ igual a $1 * 10^{-4}$. Observar como, a diferencia del estudio de referencia, la implementación del método elige las K aristas mas significativas por nodo, en lugar de por el grafo entero. Es decir, la cantidad de aristas total sera $N * K$. Este cambio tuvo un impacto negativo en el modelo, empeorando el F1 score.

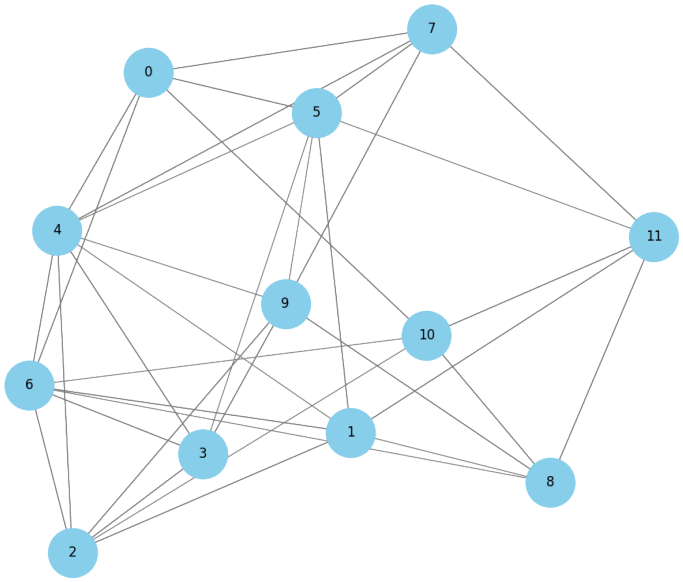

Figura 4: Grafo inferido por modelo de GDN

Como ultimo experimento, se decidió realizar un ensamble de 12 modelos de MTAD-GAT, donde en

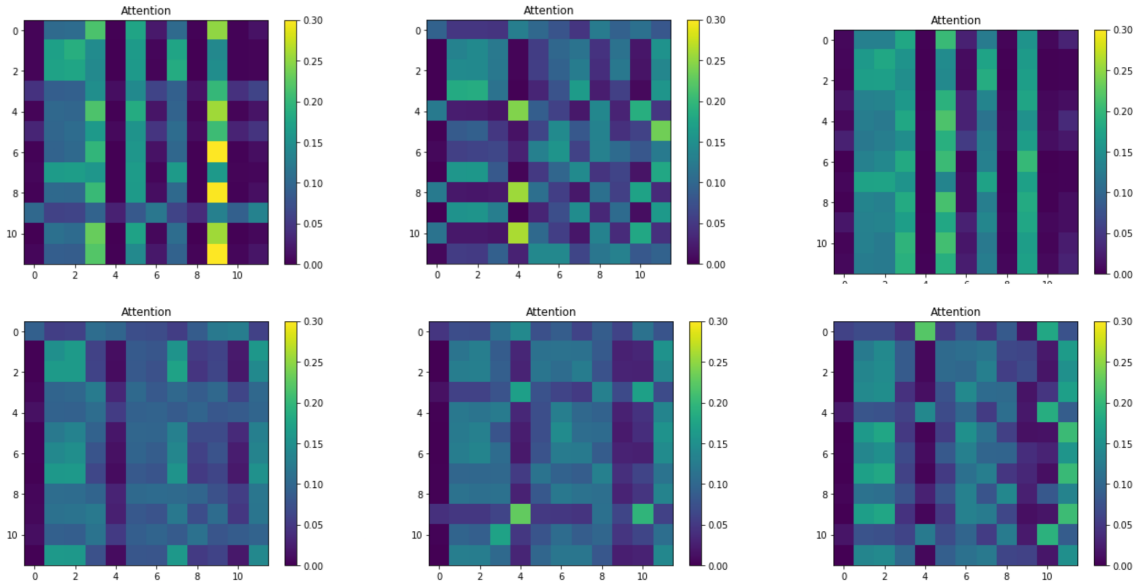

Figura 5: Ejemplos de las matrices de attention entre las series temporales de MTAD-GAT.

cada uno de los modelos se tomo una serie distinta como serie objetivo. Esto permite ajustar la función de *loss* para sobreajustarse a solo una serie en lugar de las 12, mejorando el resultado considerablemente. Sin embargo, tiene la contra de tener que entrenar tantos modelos como series se tengan, aumentando el costo computacional y disminuyendo la escalabilidad de la solución.

6. Conclusiones

Los resultados experimentales revelan que los modelos de grafos ofrecen mejoras significativas en la detección de anomalías en series temporales. No obstante, es importante destacar que estas mejoras no son uniformes entre todos los modelos de grafos. Por ejemplo, se observa que el modelo GDN presenta resultados notablemente superiores a los del modelo MTAD-GAT. Esta disparidad podría estar relacionada con el enfoque de inferencia de grafos utilizado por cada modelo: mientras que el primero emplea un grafo completo, el segundo infiere un grafo a partir de las series temporales.

Además, se requiere llevar a cabo más investigaciones para analizar los casos de uso y las limitaciones en la aplicación de este tipo de modelos. Esto incluye explorar una variedad más amplia de conjuntos de datos y probar una gama más amplia de modelos.

Referencias

[1] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv*, 2021.

[2] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. *arXiv*, 2021.

[3] G. García González, S. Martinez Tagliafico, A. Fernández, G. Gómez, J. Acu na, and P. Casas. One model to find them all deep learning for multivariate time-series anomaly detection in mobile network data. *IEEE Transactions on Network and Service Management*, page 1–1, 2024.

[4] Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I. Webb, Irwin King, and Shirui Pan. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *arXiv*, 2023.

[5] ML4ITS. mtad-gat-pytorch. `https://github.com/ML4ITS/mtad-gat-pytorch`, 2023.

[6] Jorge Pessoa. pytorch-gdn. https://github.com/jorge-pessoa/pytorch-gdn, 2021.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv*, 2017.

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv*, 2017.

[9] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, November 2020.