

MTGNNe: Una extensión a la arquitectura MTGNN para el pronóstico de series de tiempo multivariantes utilizando Graph Neural Networks (GNN)

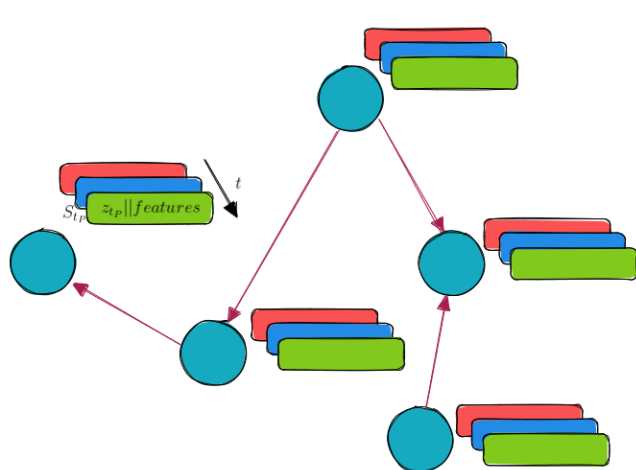
Daniel Alonso

Las series de tiempo como un grafo

- En el artículo *Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks* (GNN) se propone un framework **MTGNN** en donde las **series de tiempo multivariantes se pueden observar como un grafo**, las variables pueden ser consideradas como nodos que están interconectados a través de sus relaciones de dependencia ocultas
- Al propagar la información a través de las estructuras, las GNN permiten que cada nodo sea **consciente del contexto de su entorno**¹
- Expresar los datos temporales como grafos **preservan la trayectoria temporal y explotan la interdependencia entre las series temporales**

GNN espacio-temporales (ST-GNN)²

- Toman como entrada una **serie de tiempo y una estructura de grafos**, y su objetivo es **predecir los valores futuros**
- Las **dependencias espaciales son capturadas por grafos convolucionales (GCN)**, mientras que las **dependencias temporales** entre los estados históricos se conservan con **RNN o convoluciones 1D**



Estructura de grafos desconocida

En el mayor de los casos, las series de tiempo no tiene una estructura de grafo explícita, por lo que, se tiene que descubrir a partir de los datos



Aprendizaje de grafos y GNN

Se centran únicamente en el paso de mensajes y pasan por alto el hecho de que la estructura puede no ser óptima

¿Cómo aprender simultáneamente la estructura de grafos y la GNN para las series de tiempo en un framework end-to-end?

¹ Asumen que el estado de un nodo depende del estado de los nodos vecinos, que es la dependencia espacial

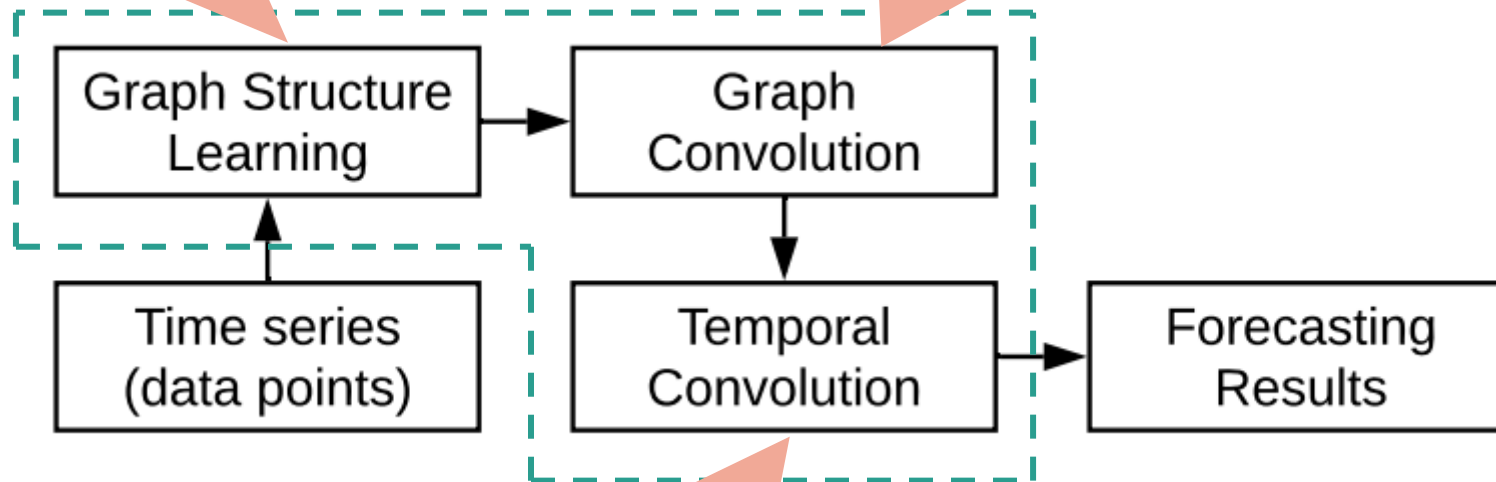
² Nacieron principalmente para resolver problemas de predicción de tráfico y reconocimiento de acciones de esqueletos

Mapa conceptual del framework

● Componentes principales

Obtener una **matriz de adyacencia dispersa adaptativa** del grafo en función de los datos, lo que resuelve **el primer reto**

Con la matriz de adyacencia del paso anterior, obtener las **dependencias espaciales** entre las variables



Capturar **patrones temporales** con múltiples frecuencias y además capturar secuencias muy largas mediante convoluciones 1D modificadas

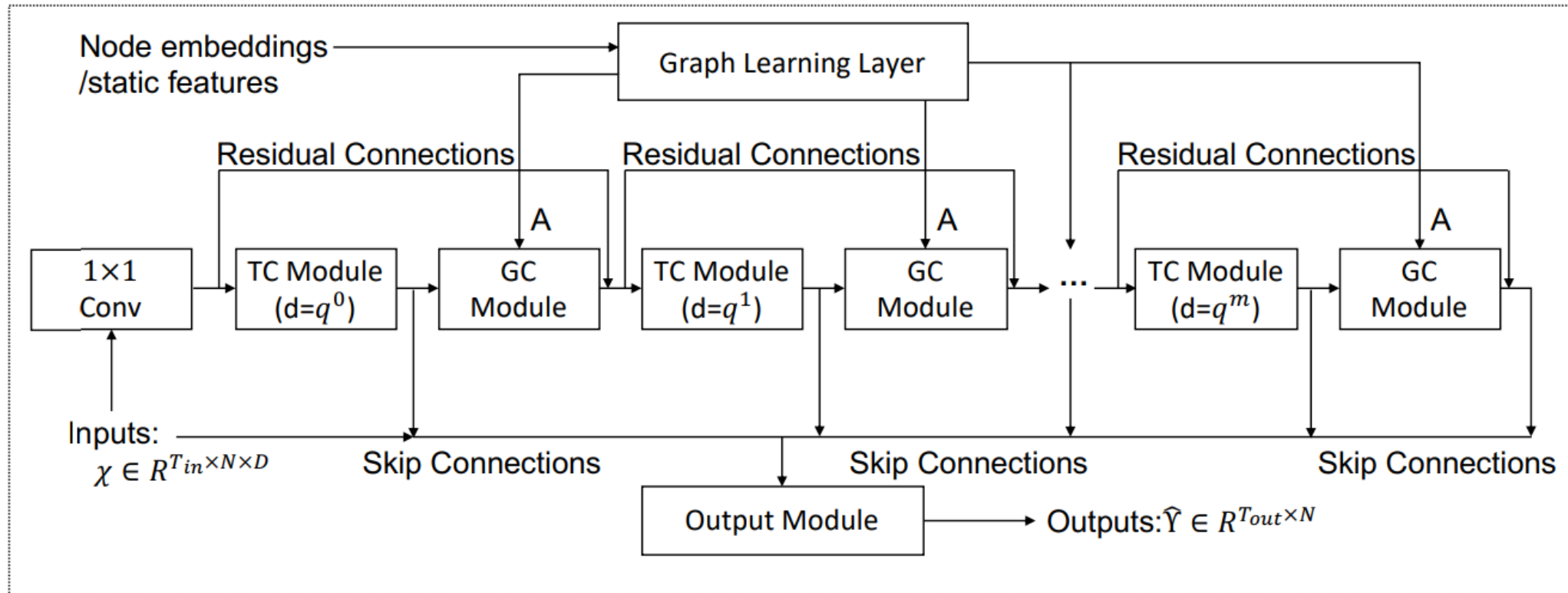
- Todos los parámetros son aprendidos en el entrenamiento, i.e., **modela las series de tiempo mientras que simultáneamente aprende la estructura del grafo, lo que resuelve el segundo reto**
- Aprendizaje por currículum ¹
- Separación de las series de tiempo en subgrafos durante el entrenamiento ²
- El framework entonces es aplicable a **grafos pequeños y grandes, series de tiempo cortas y largas, con y sin una estructura del grafo definida**

¹ El algoritmo comienza por resolver el problema más sencillo, que es predecir el siguiente paso, para que el algoritmo pueda aprender la tarea difícil paso a paso. Dicha estrategia de aprendizaje está motivada por la forma en que los profesores enseñan a los alumnos, donde se da el material con un cierto orden

² Entrenar un grafo requiere guardar la información de los estados intermedios de los nodos, lo que puede ocasionar problemas de desbordamiento de memoria. Para reducir la complejidad en tiempo y memoria se sigue una estrategia que consiste en separar de manera aleatoria los nodos en diferentes grupos, y el algoritmo aprende la estructura de los subgrafos con los nodos muestreados

Arquitectura original

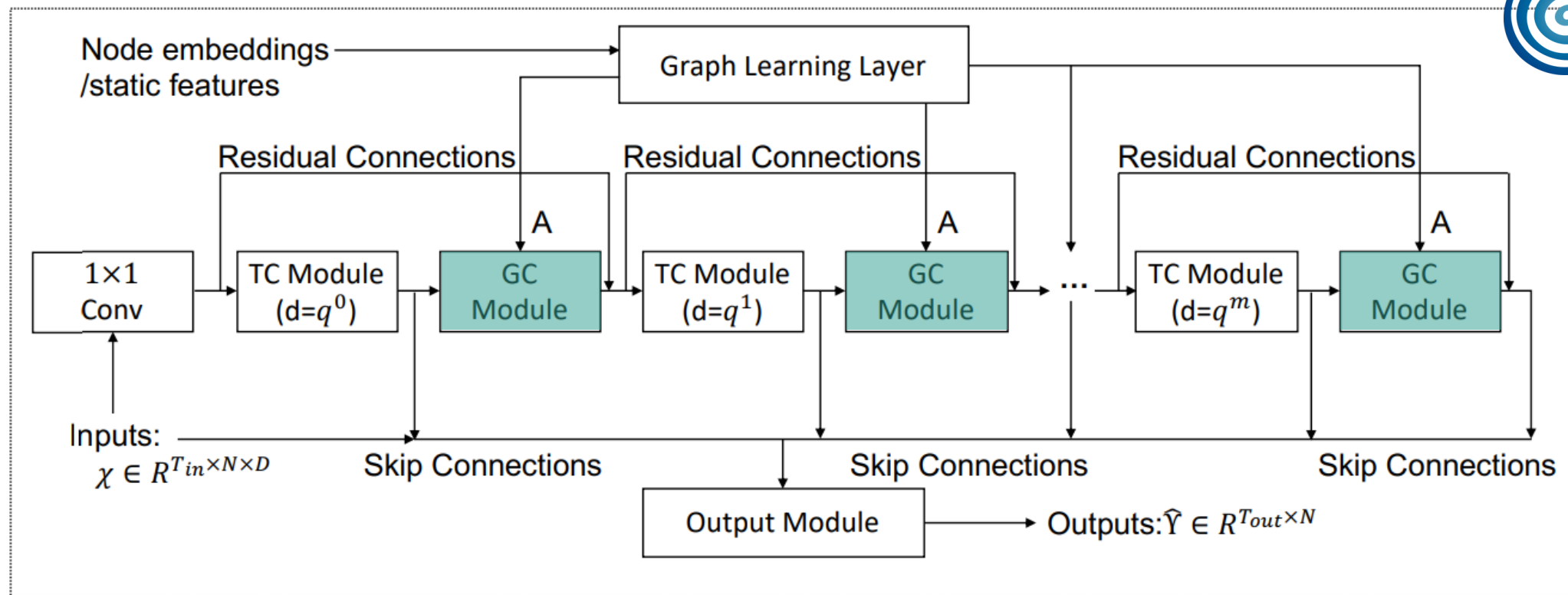
- Primero, una convolución de 1x1 proyecta las entradas a un espacio latente
- Para evitar el problema del desvanecimiento del gradiente se añaden conexiones residuales y skip¹
- Las conexiones skip son convoluciones que estandarizan la información que va al módulo de salida para que tenga la misma longitud de secuencia
- Para obtener los resultados finales, el módulo de salida proyecta las características ocultas a la dimensión de salida deseada



¹ Ayudan a disminuir el over-smoothing aún utilizando múltiples capas GNN, la intuición de las conexiones skip es que crean una mezcla de modelos

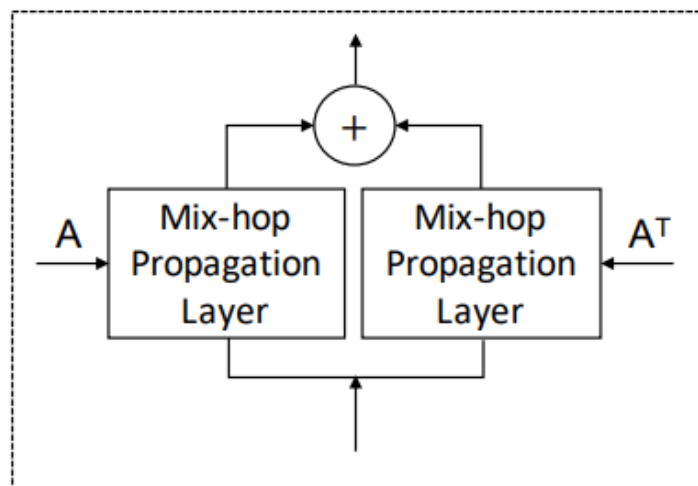
Propuesta y justificación

- La propuesta radica en incorporar un par de mecanismos de atención en el módulo de Graph Convolution (GC), con los objetivos:
 - Obtener la importancia de los nodos vecinos al agregar su información
 - Obtener la importancia de la transmisión de información entre las capas GCN
- Extender el framework para utilizar una librería de configuración automática de algoritmos, específicamente Optuna

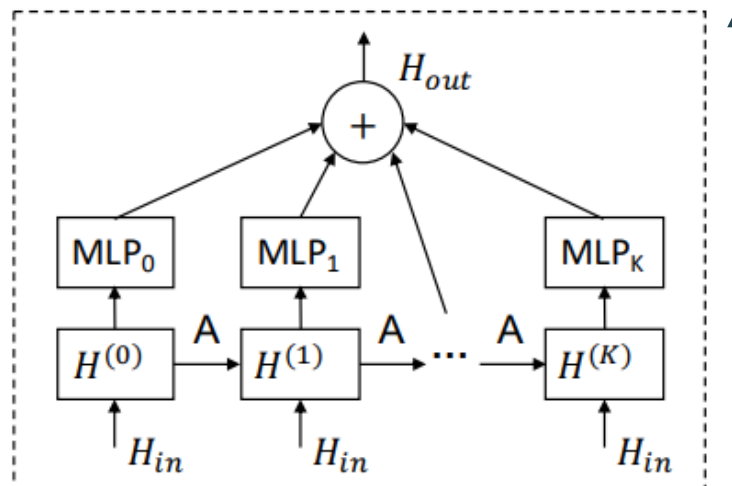


Graph Convolution Module (original)

- Agregar la información de un nodo con la de sus vecinos para manejar las dependencias espaciales de un grafo
- Consta de 2 capas de propagación Mix-hop para procesar la información de entrada y salida que pasa por cada nodo por separado



(a) GC module



(b) Mix-hop propagation layer

2. Selección de la información

$$\mathbf{H}_{out} = \sum_{i=0}^K \mathbf{H}^{(k)} \mathbf{W}^{(k)}$$

- En caso de no existir dependencias espaciales, la agregación de información simplemente **añade ruido**
- Este **paso filtra la información** importante producida en cada paso
- La matriz **W** funciona como un selector de características

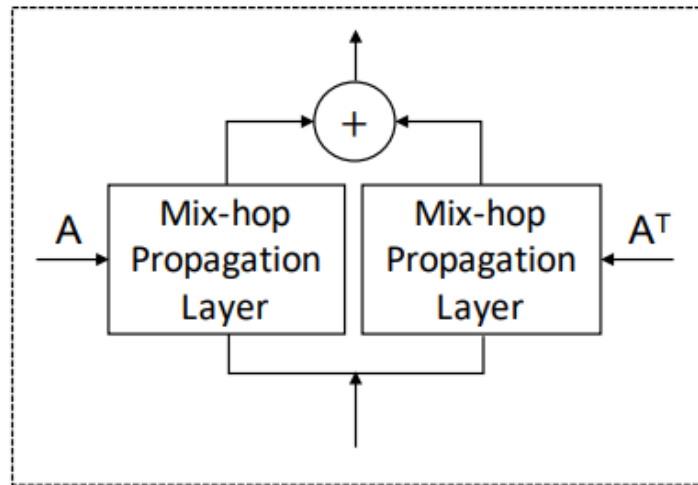
1. Propagación de la información

$$\mathbf{H}^{(k)} = \beta \mathbf{H}_{in} + (1 - \beta) \tilde{\mathbf{A}} \mathbf{H}^{(k-1)}$$

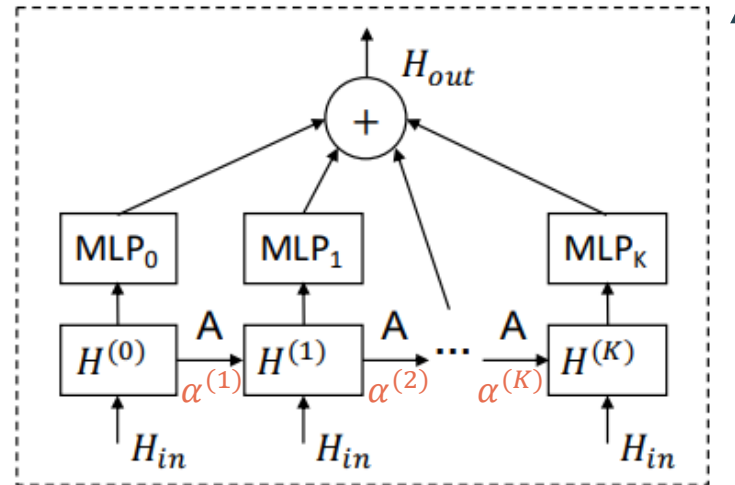
- Propaga la información de los nodos junto con la estructura del grafo de forma **recursiva**
- Al aumentar el número de capas en GCN, el estado de los nodos ocultos convergen a un único punto, **over-smoothing**
- Para solucionarlo, se conserva una **proporción** de los estados originales, tal que el estado de los nodos propagados preservan la localidad y exploran una vecindad profunda

Graph Convolution Module (propuesta)

- La propuesta es incorporar mecanismos de atención en dos partes dentro del GCN, específicamente en el paso de propagación de la capa Mix-hop
- Se mantiene la arquitectura



(a) GC module



(b) Mix-hop propagation layer

2. Selección de la información

$$\mathbf{H}_{out} = \sum_{i=0}^K \mathbf{H}^{(k)} \mathbf{W}^{(k)}$$

- En caso de no existir dependencias espaciales, la agregación de información simplemente **añade ruido**
 - Este **paso filtra la información** importante producida en cada paso
- La matriz **W** funciona como un selector de características

1. Propagación de la información

$$\mathbf{H}^{(k)} = \beta \mathbf{H}_{in} + \alpha^{(k-1)} (1 - \beta) \tilde{\mathbf{A}} \mathbf{H}^{(k-1)}$$

$$\tilde{\mathbf{A}} = \alpha \tilde{\mathbf{A}}$$

- Propaga la información de los nodos junto con la estructura del grafo de forma **recursiva**
- **Mecanismos de atención para agregar la información de un nodo con la de sus vecinos (MTGAN)**
- **Mecanismos de atención al propagar la información por el número de capas de la GCN (MTGNNAH / MTGANAH)**

GCN - MTGAN

- La entrada H_{in} es un tensor \mathbf{X} con dimensiones [B, F, N, W]
- Primera transformación al pasarla por una FCN que mapea a las características de entrada (canales convolucionales) a las características de salida (canales residuales) para cada cabeza k :

$$\overrightarrow{g}_i^k = \mathbf{W}^k \overrightarrow{x}_i$$

- Posteriormente se divide la información entre el número de cabezas \rightarrow [B, W, N, H, F]
- Cálculo de los coeficientes de atención para cada cabeza, e_{ij}^k :

Importancia del
nodo j al nodo i

$$e_{ij} = \alpha(\mathbf{W}\overrightarrow{x}_i, \mathbf{W}\overrightarrow{x}_j) = \alpha(\overrightarrow{g}_i, \overrightarrow{g}_j)$$

Mecanismo de
atención

$$e_{ij} = \text{LeakyReLU}(\mathbf{a}^T [\overrightarrow{g}_i || \overrightarrow{g}_j])$$

- Con la matriz de adyacencia $\mathbf{A}_{N \times N \times H}$ se realiza una máscara, en donde $e_{ij} = -\infty$ si no existe una arista que conecte a los nodos i, j , tal que $e_{ij} = -\infty \rightarrow \exp(e_{ij}) \sim 0$
- Normalización de los coeficientes de atención:

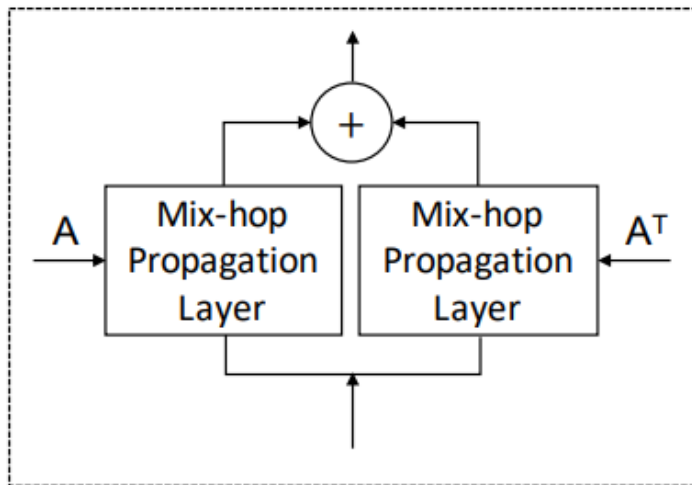
$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})}$$

- Salida final:

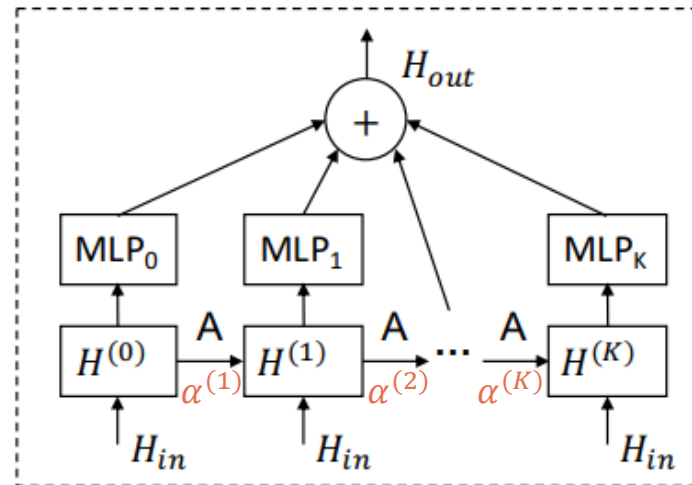
$$\overrightarrow{h}_i'^k = \sum_{j \in N_i} \alpha_{ij}^k \overrightarrow{g}_j^k$$

GCN – MTGNN / MTGANAH

- Agregar mecanismos de atención al propagar la información por el número de capas de la GCN



(a) GC module



(b) Mix-hop propagation layer

- Agregar un parámetro entrenable con el tamaño de la profundidad de la GCN k , $e \in \mathbb{R}^k$

- Normalizar: $\alpha = \text{softmax}(e)$

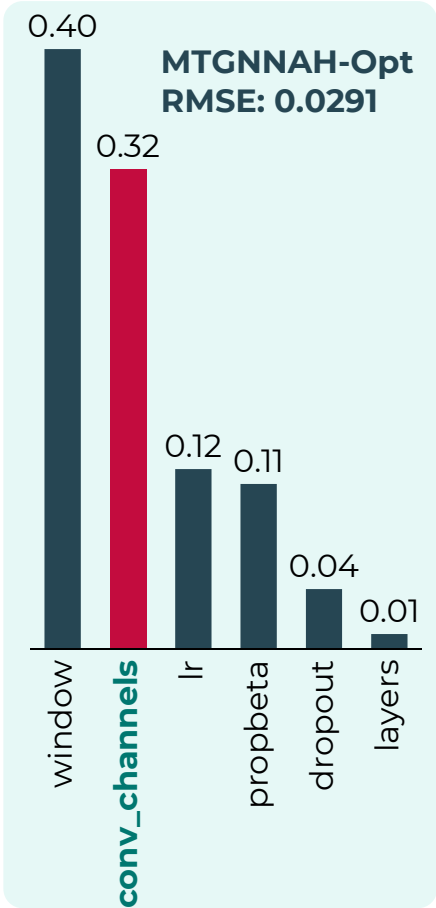
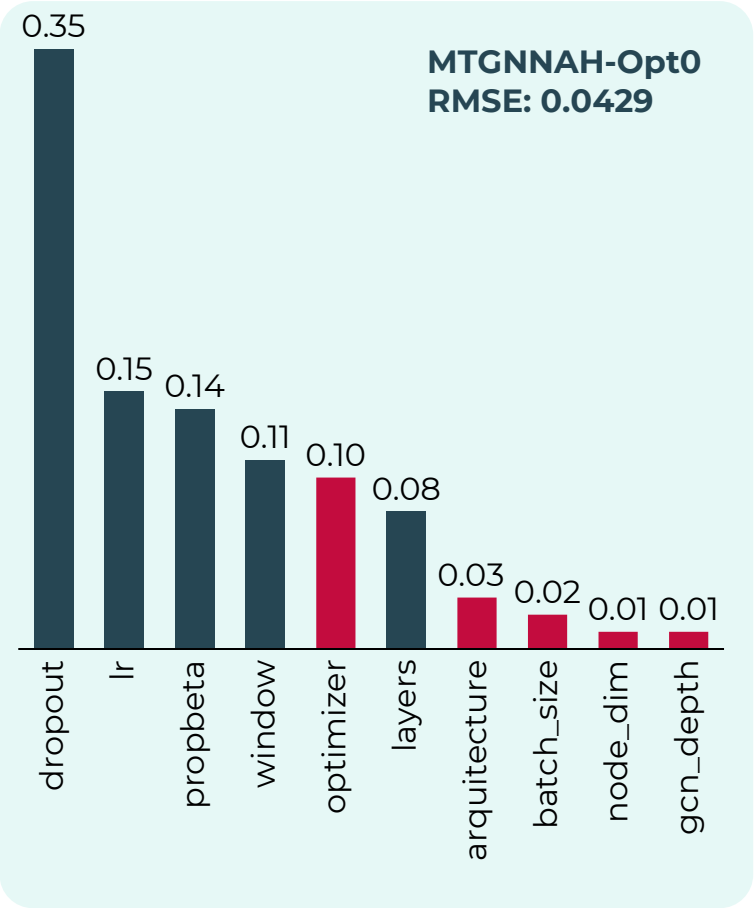
- Tal que cada capa se multiplica por su α^k

$$\mathbf{H}^{(k)} = \beta \mathbf{H}_{in} + \alpha^{(k-1)}(1 - \beta) \tilde{\mathbf{A}} \mathbf{H}^{(k-1)}$$

- **Mecanismos de atención al propagar la información por el número de capas de la GCN (MTGNN / MTGANAH)**

Optimización de hiperparámetros

- La optimización de hiperparámetros se realizó con la paquetería Optuna con optimización bayesiana
- La función objetivo es **minimizar RMSE**
- Los hiperparámetros restantes se dejaron como en el repositorio del artículo
- Al ir quitando los hiperparámetros, se dejaban en sus valores originales



Hiperparámetro	MTGNN	MTGAN	MTGNNAH-Opt
batch_size	4	4	4
lr	1e-4	1e-4	1.6e-4
dropout	0.3	0.3	0.1
window	24*7	24*7	24*7*3
conv_channels	16	16	32
residual_channels	16	16	32
skip_channels	32	32	64
end_channels	64	64	128
gcn_depth	2	2	2
propbeta	0.05	0.05	0.2
layers	5	5	7
n_heads	NA	16	NA
concat	NA	True	NA

Metodología de evaluación

Comparación

La comparación de los modelos se realiza con el problema *single step* a un horizonte de 3 pasos, utilizando la base de datos *Exchange Rate*

Resultados

Promedio de 10 ejecuciones para cada modelo en el conjunto de prueba



Entrenamiento

El entrenamiento se realiza durante 30 épocas con los hiperparámetros utilizados en el trabajo original, modificando aquellos resultantes de la optimización. La metodología de entrenamiento se mantiene del trabajo original.

Métricas de evaluación

MAE, RMSE, MAPE, RRSE, CORR

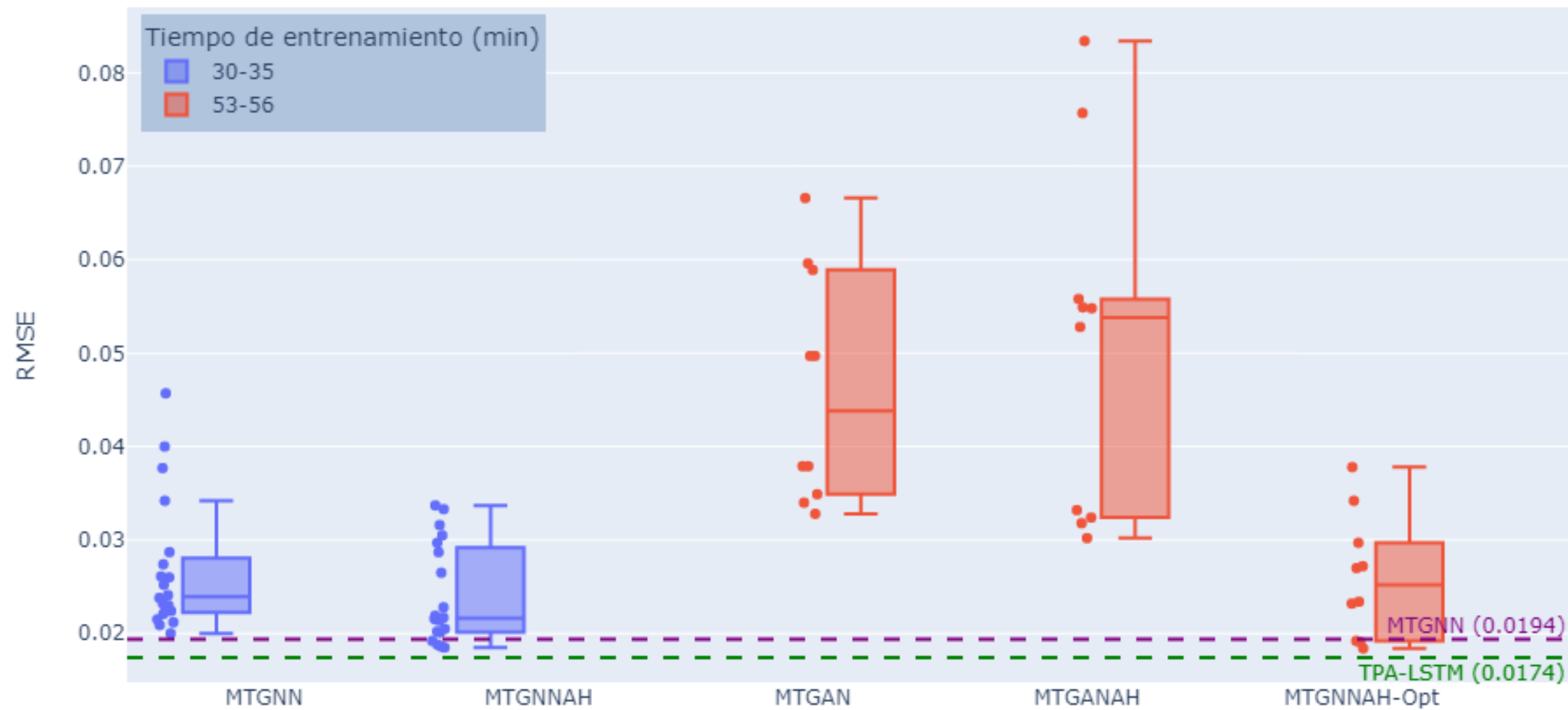
Resultados single-step

Comparación con modelos de series de tiempo multivariantes

● Resultados MTGNN ● Evaluación

Dataset		Solar-Energy				Traffic				Electricity				Exchange-Rate			
		Horizon				Horizon				Horizon				Horizon			
Methods	Metrics	3	6	12	24	3	6	12	24	3	6	12	24	3	6	12	24
AR	RSE	0.2435	0.3790	0.5911	0.8699	0.5991	0.6218	0.6252	0.63	0.0995	0.1035	0.1050	0.1054	0.0228	0.0279	0.0353	0.0445
	CORR	0.9710	0.9263	0.8107	0.5314	0.7752	0.7568	0.7544	0.7519	0.8845	0.8632	0.8591	0.8595	0.9734	0.9656	0.9526	0.9357
VARMLP	RSE	0.1922	0.2679	0.4244	0.6841	0.5582	0.6579	0.6023	0.6146	0.1393	0.1620	0.1557	0.1274	0.0265	0.0394	0.0407	0.0578
	CORR	0.9829	0.9655	0.9058	0.7149	0.8245	0.7695	0.7929	0.7891	0.8708	0.8389	0.8192	0.8679	0.8609	0.8725	0.8280	0.7675
GP	RSE	0.2259	0.3286	0.5200	0.7973	0.6082	0.6772	0.6406	0.5995	0.1500	0.1907	0.1621	0.1273	0.0239	0.0272	0.0394	0.0580
	CORR	0.9751	0.9448	0.8518	0.5971	0.7831	0.7406	0.7671	0.7909	0.8670	0.8334	0.8394	0.8818	0.8713	0.8193	0.8484	0.8278
RNN-GRU	RSE	0.1932	0.2628	0.4163	0.4852	0.5358	0.5522	0.5562	0.5633	0.1102	0.1144	0.1183	0.1295	0.0192	0.0264	0.0408	0.0626
	CORR	0.9823	0.9675	0.9150	0.8823	0.8511	0.8405	0.8345	0.8300	0.8597	0.8623	0.8472	0.8651	0.9786	0.9712	0.9531	0.9223
LSTNet-skip	RSE	0.1843	0.2559	0.3254	0.4643	0.4777	0.4893	0.4950	0.4973	0.0864	0.0931	0.1007	0.1007	0.0226	0.0280	0.0356	0.0449
	CORR	0.9843	0.9690	0.9467	0.8870	0.8721	0.8690	0.8614	0.8588	0.9283	0.9135	0.9077	0.9119	0.9735	0.9658	0.9511	0.9354
TPA-LSTM	RSE	0.1803	0.2347	0.3234	0.4389	0.4487	0.4658	0.4641	0.4765	0.0823	0.0916	0.0964	0.1006	0.0174	0.0241	0.0341	0.0444
	CORR	0.9850	0.9742	0.9487	0.9081	0.8812	0.8717	0.8717	0.8629	0.9439	0.9337	0.9250	0.9133	0.9790	0.9709	0.9564	0.9381
MTGNN	RSE	0.1778	0.2348	0.3109	0.4270	0.4162	0.4754	0.4461	0.4535	0.0745	0.0878	0.0916	0.0953	0.0194	0.0259	0.0349	0.0456
	CORR	0.9852	0.9726	0.9509	0.9031	0.8963	0.8667	0.8794	0.8810	0.9474	0.9316	0.9278	0.9234	0.9786	0.9708	0.9551	0.9372
MTGNN+sampling	RSE	0.1875	0.2521	0.3347	0.4386	0.4170	0.4435	0.4469	0.4537	0.0762	0.0862	0.0938	0.0976	0.0212	0.0271	0.0350	0.0454
	CORR	0.9834	0.9687	0.9440	0.8990	0.8960	0.8815	0.8793	0.8758	0.9467	0.9354	0.9261	0.9219	0.9788	0.9704	0.9574	0.9382

Resultados de Exchange-Rate single step 3



RMSE	0.027±0.007	0.024±0.005	0.046±0.012	0.051±0.019	0.026±0.007
CORR	0.979±0.001	0.980±0.001	0.965±0.003	0.966±0.002	0.979±0.000
Tiempo (min)	33.74±0.156	34.99±0.025	53.80±0.502	56.47±0.993	54.06±1.153
Parámetros	337,345	337,365	340,095	340,115	8,082,957

Conclusiones

- La propuesta original **MTGNN** es un framework que pronostique series de tiempo multivariantes con o sin una estructura de grafos definidas con un enfoque de GNN
- Los resultados, en la arquitectura original muestran que superan el SOAT en 3 de los 4 conjuntos de datos de referencia
- La arquitectura propuesta **MTGNN_e** radica en extender el framework para:
 - Agregar mecanismos de atención para obtener la importancia de los nodos vecinos al agregar la información (**MTGAN**)
 - Agregar mecanismos de atención al propagar la información al pasar por el número de capas de la GCN (**MTGNN_{NAH}** / **MTGAN_{NAH}**)
 - Utilizar una librería para realizar una optimización de hiperparámetros (**MTGNN_{NAH}-Opt**)
- La optimización bayesiana se emplea para optimizar hiperparámetros cuando evaluar la función objetivo es costosa
- En la primera optimización realizada no se obtuvieron mejores resultados que los experimentos con los hiperparámetros empleados por los autores, la hipótesis es que el número de hiperparámetros es alto y no se le dio el número de intentos adecuados al optimizador
- En la optimización, específicamente con la librería de Optuna se puede obtener información relevante, como la importancia de los hiperparámetros en el proceso de optimización
- La arquitectura **MTGNN_{NAH}** muestra mejores resultados que la arquitectura **MTGNN** en un tiempo similar de entrenamiento ya que no se agregan muchos más parámetros
- La arquitectura **MTGAN** en general empeora con respecto a **MTGNN**, además de que toma el doble de tiempo de entrenamiento
- La arquitectura **MTGNN_{NAH}-Opt** tiene un rendimiento un poco mejor que **MTGNN** pero no es mejor que **MTGNN_{NAH}**, aunque por el aumento en el número de canales convoluciones, residuales, skip y del módulo de salida, el número de parámetros aumenta 23.96x, duplicando el tiempo de entrenamiento
- **La extensión al framework, específicamente al introducir mecanismos de atención al propagar la información por el número de capas de la GCN, al menos con los datos Exchange rate, prediciendo single step a 3 pasos, da mejor resultados que el framework original**

Trabajo a futuro

- Sustituir el módulo GC de la arquitectura original por una ST-GNN ya implementada, p.ej., A3T-GCN
 - Para realizarlo es necesario transformar la matriz de adyacencia en una lista de aristas junto con sus pesos
- En la arquitectura **MTGAN** implementar una función de activación a la salida de la capa Graph Attention
- Extender los experimentos para utilizar los horizontes de pronósticos single step de 6, 12 y 24 y validar si la arquitectura **MTGNNAH** mejora los resultados en cada caso
- Extender los experimentos a todos los conjuntos de base de datos, al menos con las que no tienen una estructura de grafo definida, para validar si la arquitectura **MTGNNAH** mejora los resultados en cada caso
- Estudiar la importancia de los hiperparámetros para ejecutar de manera adecuada la optimización de hiperparámetros tal que se obtengan los beneficios de utilizar configuración automática de algoritmos

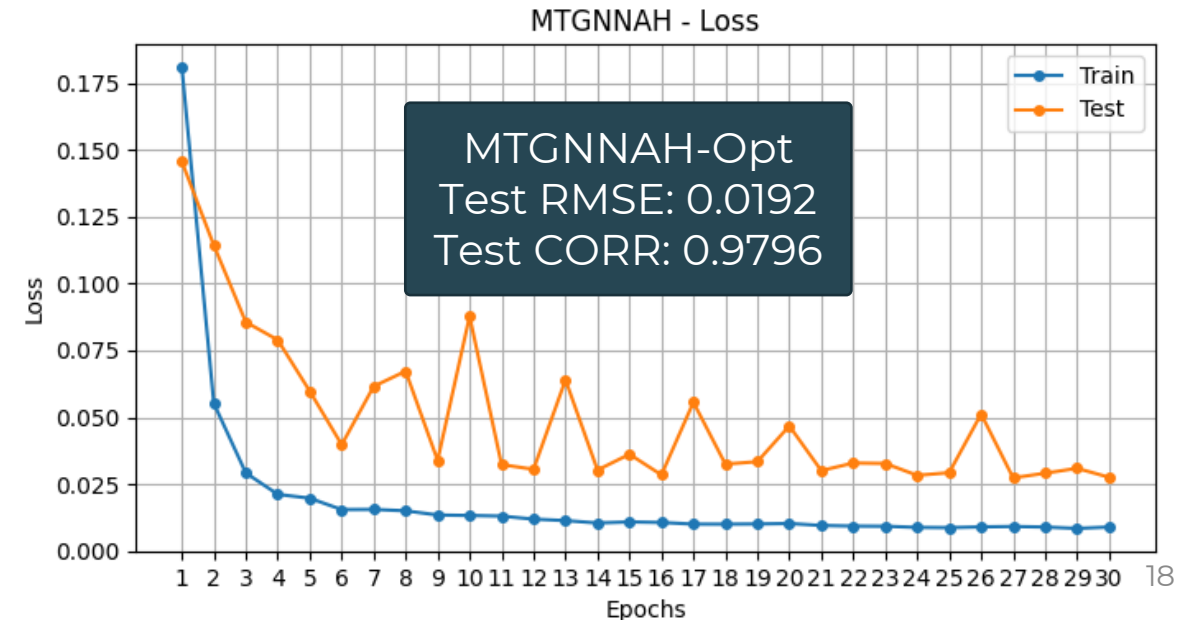
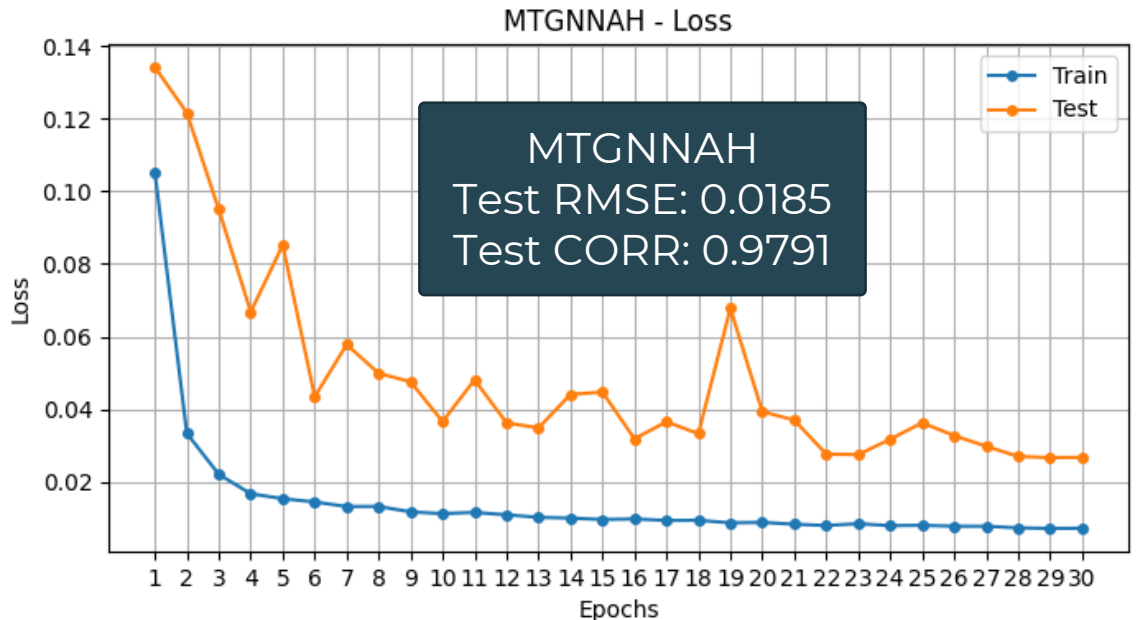
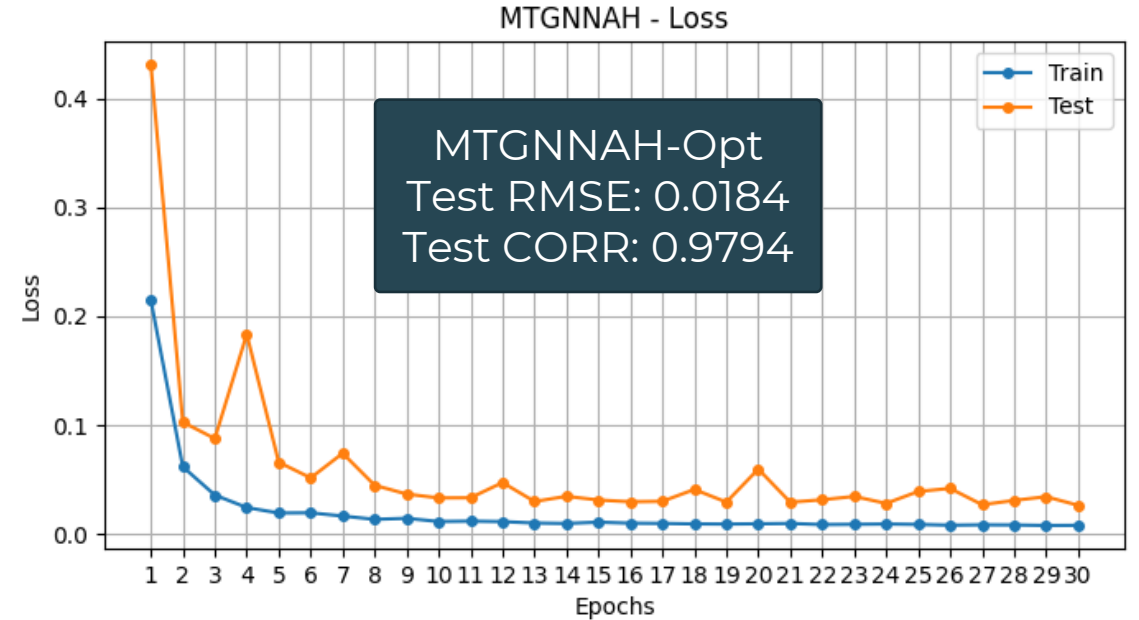
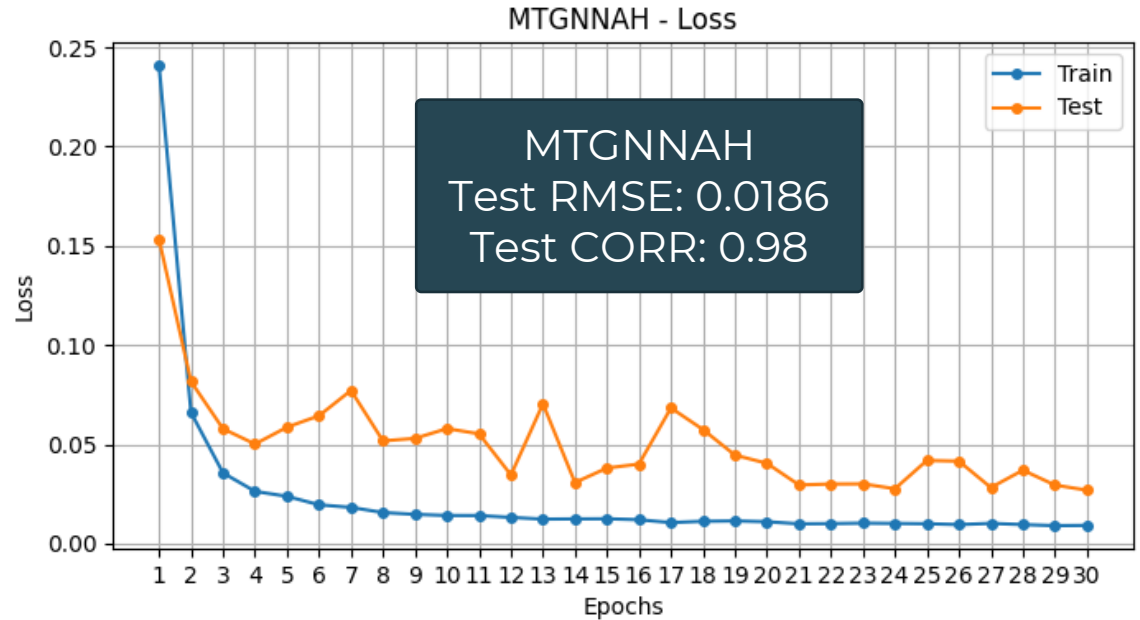


Preguntas



Anexo

Ejemplos de resultados para MTGNNAH y MTGNNAH-Opt



Hiperparámetros


Hiperparámetro	Descripción	MTGNN	MTGAN	MTGNNAH-Opt
batch_size	Tamaño del lote	4	4	4
lr	Tasa de aprendizaje	1e-4	1e-4	1.6e-4
dropout	Probabilidad de dropout	0.3	0.3	0.1
window	Tamaño de ventana	24*7	24*7	24*7*3
conv_channels	Canales de convolución	16	16	32
residual_channels	Canales residuales	16	16	32
skip_channels	Canales en las conexiones skip	32	32	64
end_channels	Canales en el módulo de salida	64	64	128
gcn_depth	Profundidad de la GCN	2	2	2
propbeta	Proporción de retención del estado original	0.05	0.05	0.2
layers	Número de capas mtggn	5	5	7
n_heads	Número de cabezas GAT	NA	16	NA
concat	Concatenar o promediar las cabezas	NA	True	NA



Introducción

- Los métodos de pronósticos en series de tiempo multivariantes **asumen intrínsecamente interdependencias entre las variables**, i.e., cada variable depende no sólo de sus valores históricos, sino también de otras variables
- Sin embargo, **los métodos existentes no explotan las interdependencias latentes** entre las variables de manera eficiente y eficaz

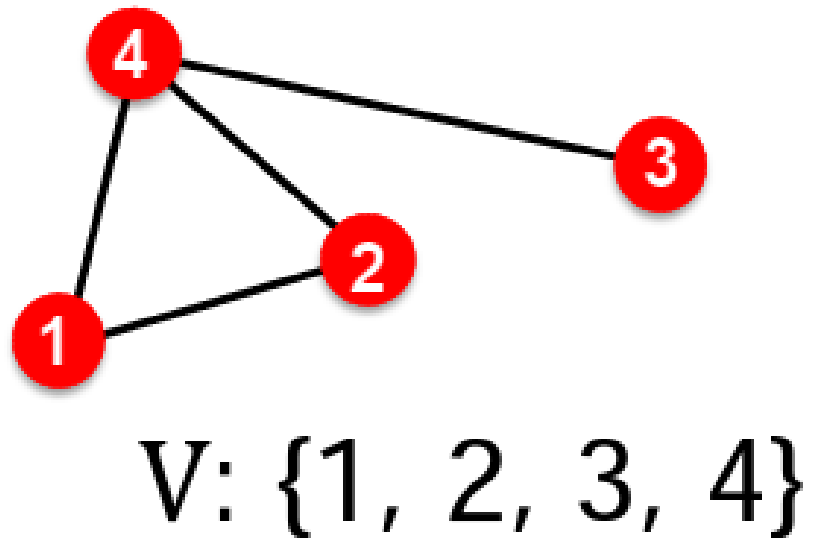
Métodos estadísticos	Aprendizaje profundo
<ul style="list-style-type: none">• Algunos modelos asumen una dependencia lineal entre variables• La complejidad de los modelos crece de manera cuadrática con el número de variables• Problema de overfitting con un número grande de variables• Por lo tanto, no escalan bien a series de tiempo multivariantes• Simplicidad e interpretabilidad	<ul style="list-style-type: none">• Buenos al capturar patrones no lineales• Dos trabajos que fueron los primeros modelos diseñados para pronóstico de series de tiempo multivariantes (LSTNet, TPA-LSTM)• Utilizan CNN para capturar dependencias locales entre las variables y RNN para preservar las dependencias temporales a largo plazo• No modelan explícitamente las dependencias entre pares de variables• Debilitan la interpretabilidad del modelo



Por lo tanto, no pueden explotar totalmente las dependencias latentes

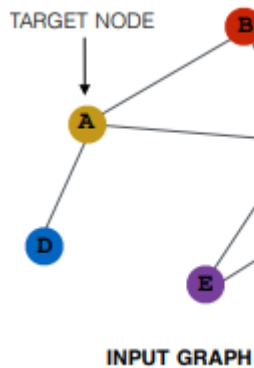
Introducción a los grafos

- Los grafos describen las relaciones entre las entidades en una red
- Grafo: $G = (V, E)$
- Vecindario de un nodo: $N(v) = \{u \in V \mid (u, v) \in E\}$
- Matriz de adyacencia A



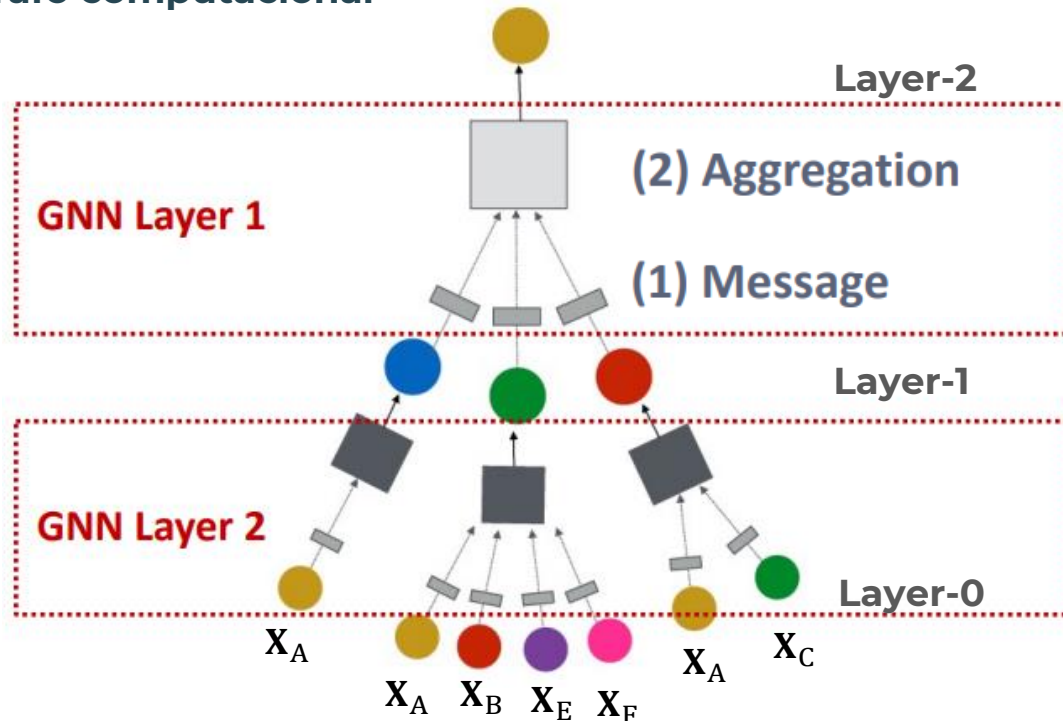
$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Introducción a Graph Neural Networks (GNN)



- Generar embeddings de los nodos basados en la red local del vecindario
- Nodos agregan información de sus vecinos usando redes neuronales
- Los nodos tienen un embedding en cada capa
- Layer-0 embedding del nodo v es su vector de entrada x_v
- Layer- k embedding toma la información de los nodos que están a k saltos de distancia

Grafo computacional



Initial 0-th layer embeddings are equal to node features

embedding of v at layer k

Total number of layers

Average of neighbor's previous layer embeddings

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Notice summation is a permutation invariant pooling/aggregation.

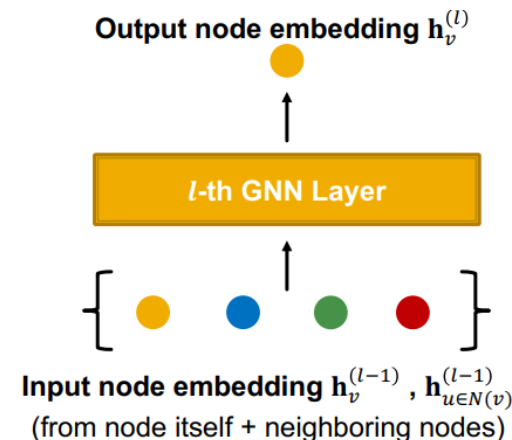
$$h_v^0 = x_v$$

$$h_v^{(k+1)} = \sigma \left(W_k \sum_{u \in N(v)} \frac{h_u^{(k)}}{|N(v)|} + B_k h_v^{(k)} \right), \forall k \in \{0, \dots, K-1\}$$

$$z_v = h_v^{(K)}$$

Idea de una capa GNN

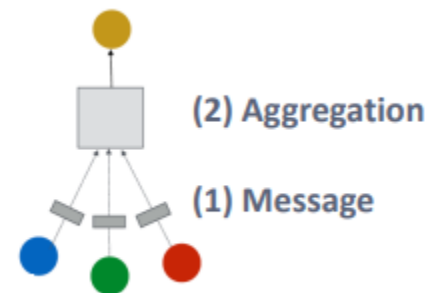
- Comprimir un conjunto de vectores en uno solo
 - **Mensaje:** Cada nodo creará un mensaje que será enviado a otros nodos
 - **Agregación:** Cada nodo agregará los mensajes de los nodos vecinos
- Output node embedding $h_v^{(l)}$



$$h_v^{(l)} = \text{AGG}^{(l)} \left(\left\{ \mathbf{m}_u^{(l)}, u \in N(v) \right\} \right)$$

Introducción a Graph Convolutional Networks (GCN)

$$\mathbf{h}_v^{(l)} = \sigma \left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|} \right)$$



- **Mensaje:**

- Para cada vecino: $\mathbf{m}_u^{(l)} = \frac{1}{|N(v)|} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}$

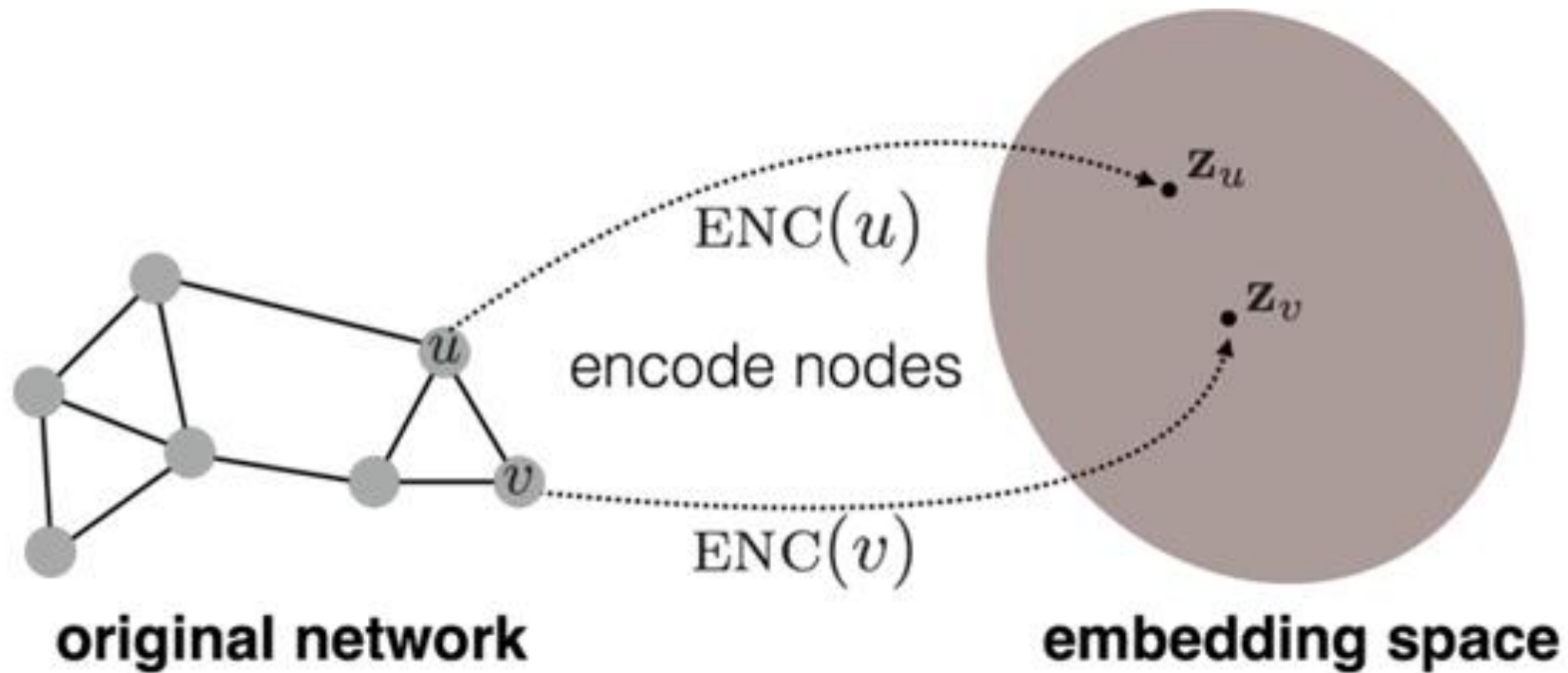
- **Agregación:**

- **Suma** sobre los mensajes de los vecinos, y después se aplica una función de activación

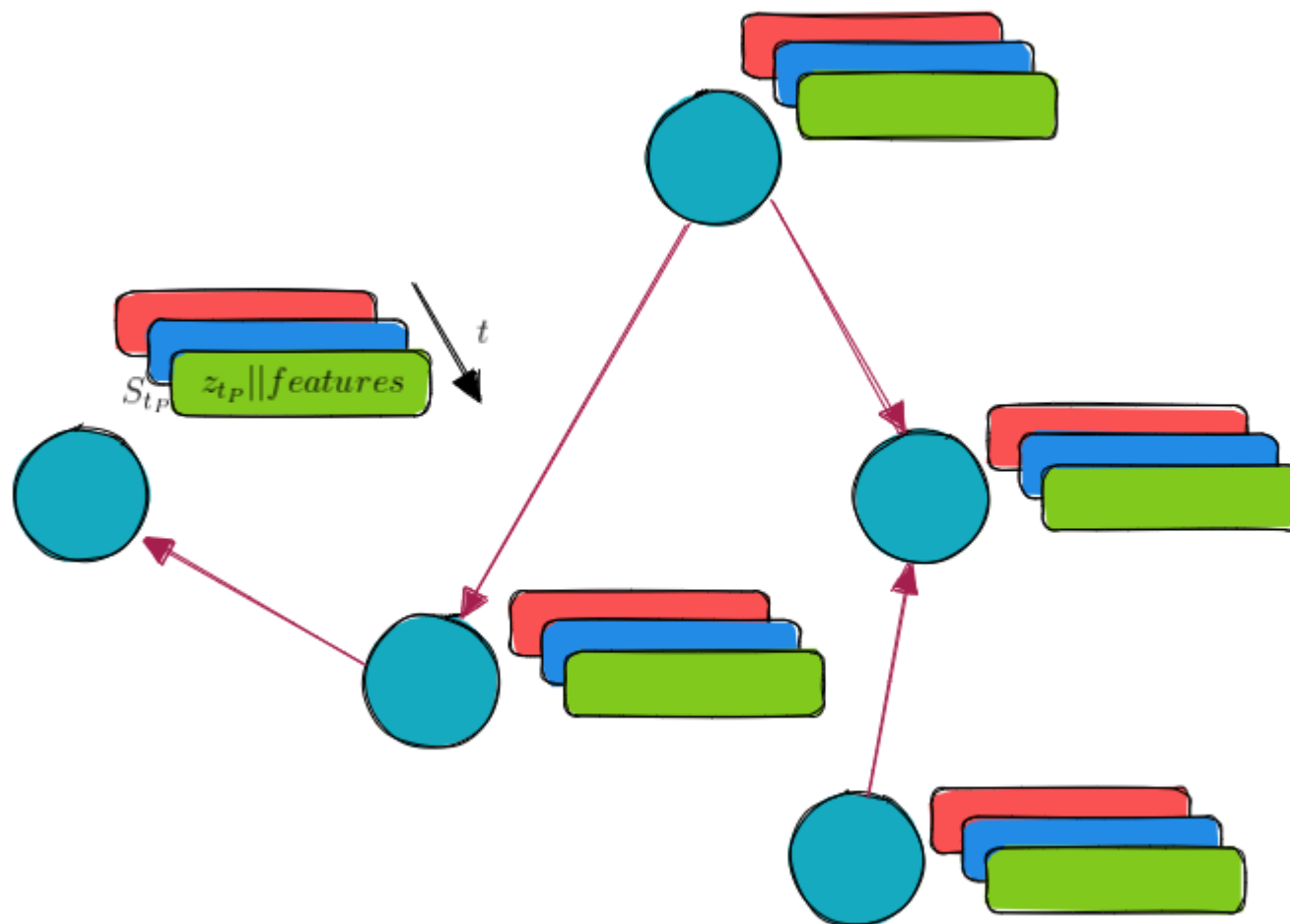
$$\mathbf{h}_v^{(l)} = \sigma \left(\text{Sum} \left(\left\{ \mathbf{m}_u^{(l)}, u \in N(v) \right\} \right) \right)$$

Embedding nodes

- El objetivo es codificar (*encode*) los nodos para que la similitud en espacio *embedding* se aproxime a la similitud en el grafo

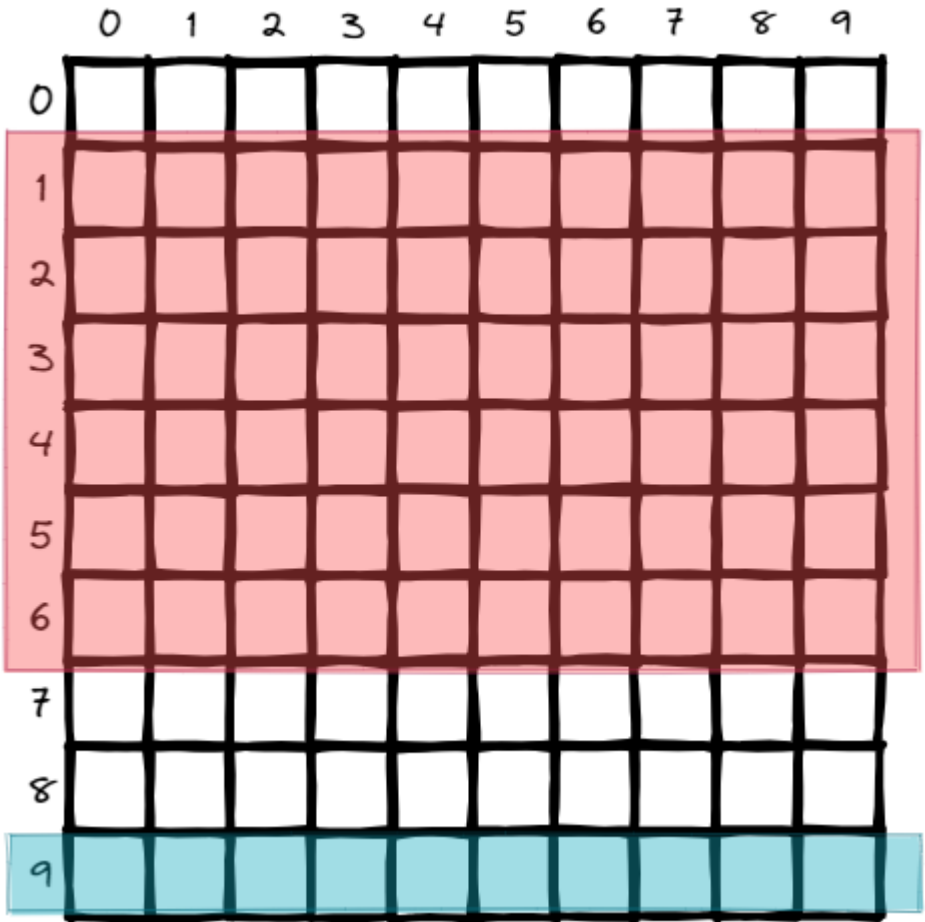
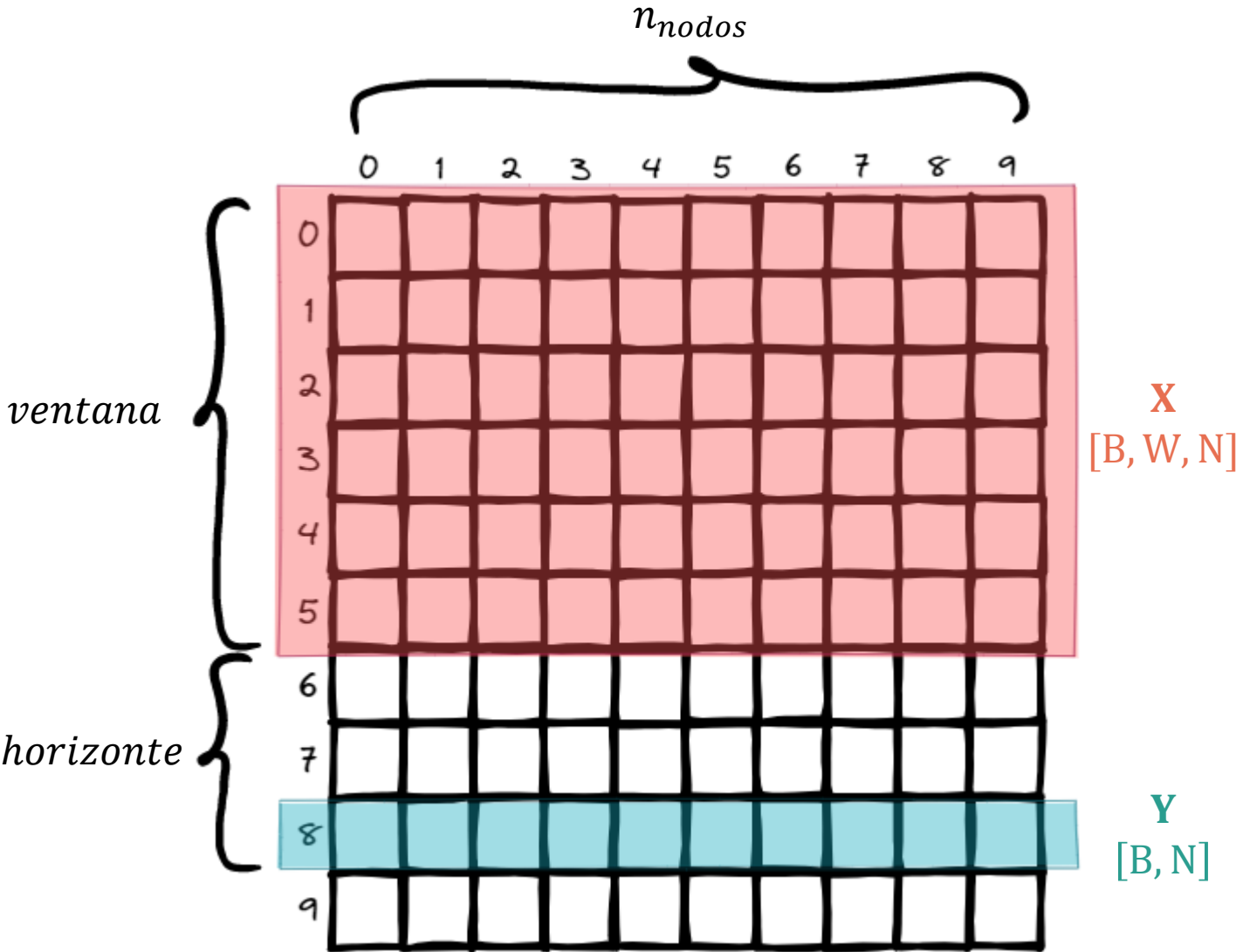


Formulación del problema



- Las variables son consideradas como nodos del grafo
- $\mathbf{z}_t \in \mathbb{R}^N$
- $\mathbf{X} = \{z_{t_1}, z_{t_2}, \dots, z_{t_P}\}$
- $\mathcal{X} = \{S_{t_1}, S_{t_2}, \dots, S_{t_P}\}$
- $\mathbf{S}_t \in \mathbb{R}^{N \times D}$
- El objetivo es predecir Q -pasos adelante: $\mathbf{Y} = \{\mathbf{z}_{t_{P+Q}}\}$
- O una secuencia de valores futuros: $\mathbf{Y} = \{z_{t_{P+1}}, z_{t_{P+2}}, \dots, z_{t_{P+Q}}\}$
- Se propone construir una función $f(\cdot): \mathcal{X} \rightarrow Y$ minimizando una pérdida absoluta con regularización $L2$

Cargador de datos *single-step*



Graph Learning Layer

- **Aprender la matriz de adyacencia adaptativamente** para capturar las **relaciones ocultas** entre los datos de la serie temporal
- Algunos estudios miden la **similitud entre parejas de nodos con una métrica de distancia, que toma $O(n^2)$** , por lo tanto restringe el uso de grafos muy grandes
- Para abordar el problema, utilizan un **enfoque de muestreo** que sólo calcula las relaciones de pareja entre un subconjunto de nodos
- Otro problema es que las **métricas de distancia suelen ser simétricas o bidireccionales**. En las series temporales se espera que el cambio de la condición de un nodo provoque el cambio de la condición de otro nodo, i.e., **se espera un grafo dirigido**
- Por lo tanto, **la relación aprendida debe ser unidireccional**. La capa de aprendizaje de grafos está diseñada específicamente para extraer relaciones unidireccionales

$$\mathbf{M}_1 = \tanh(\alpha \mathbf{E}_1 \Theta_1)$$

$$\mathbf{M}_2 = \tanh(\alpha \mathbf{E}_2 \Theta_2)$$

$$\mathbf{A} = \text{ReLU}(\tanh(\alpha(\mathbf{M}_1 \mathbf{M}_2^T - \mathbf{M}_2 \mathbf{M}_1^T)))$$

for $i = 1, 2, \dots, N$

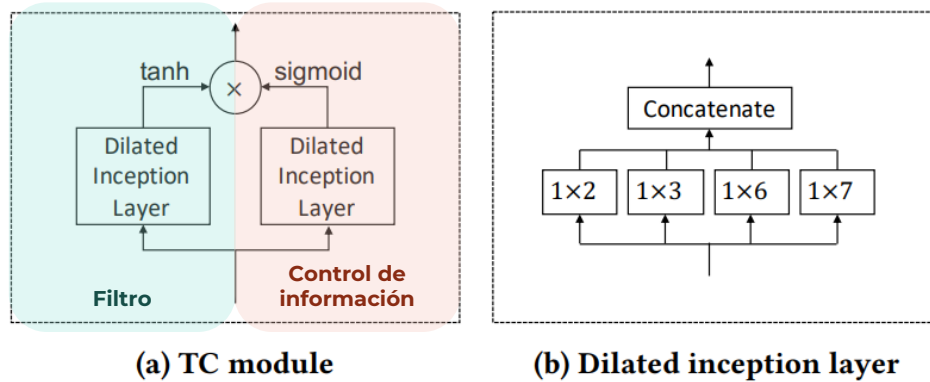
$\text{idx} = \text{argtopk}(\mathbf{A}[i, :])$

$\mathbf{A}[i, -\text{idx}] = 0,$

Con la diferencia y ReLU regularizan a \mathbf{A} tal que la diagonal superior sea positiva y la inferior sea 0, por lo tanto, se modelan las relaciones unidireccionales

Hacer la matriz de adyacencia dispersa, mientras se reduce el costo computacional
Para cada nodo, se selecciona el top- k nodos más cercanos en su vecindario

Temporal Convolution Module



- Aplicar un conjunto de filtros de convolución 1D dilatadas para **extraer características temporales**
- La capa de inception dilatada¹ se propone para encontrar patrones temporales con **varios rangos y que maneje secuencias muy largas**

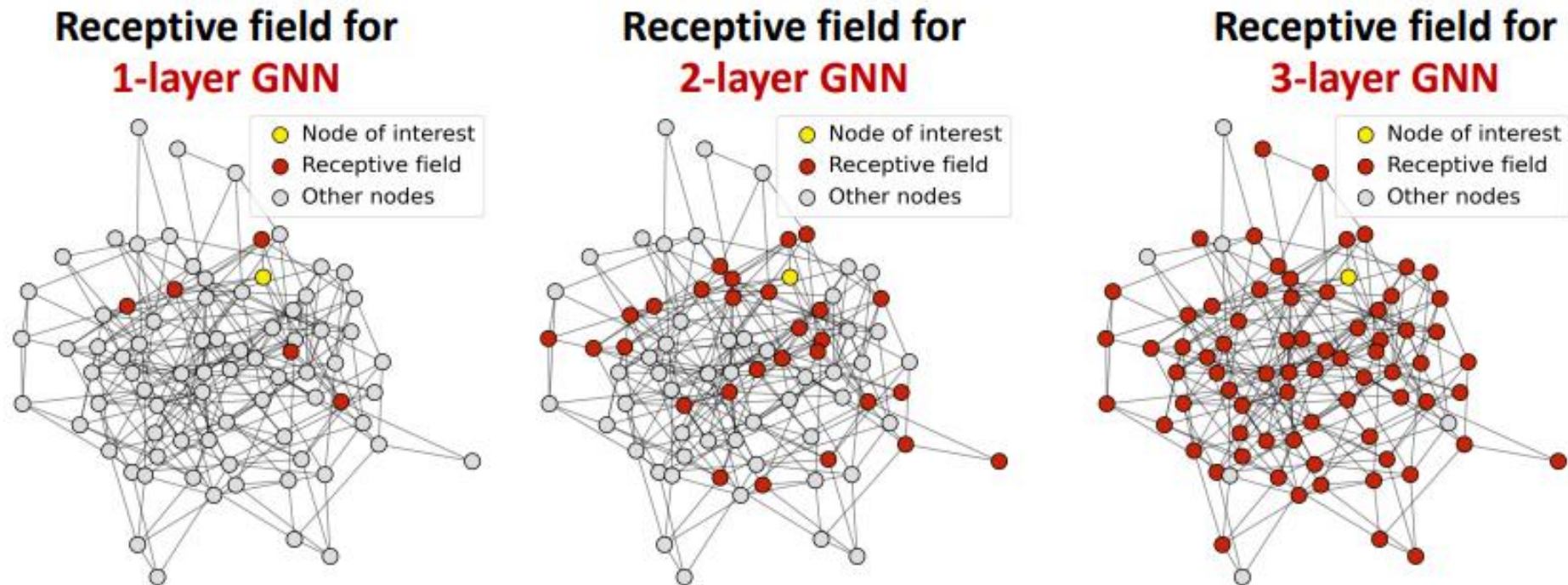
Incepción	Dilatación
<ul style="list-style-type: none">- Escoger el tamaño de kernel en CNN es complicado- El tamaño del filtro puede ser muy grande para encontrar los patrones en el corto plazo, o demasiado pequeño para descubrir suficientes patrones en el largo plazo- Se usan unos tamaños propuestos de kernel para capturar la naturaleza de las series de tiempo	<ul style="list-style-type: none">- El campo receptivo de una CNN crece de manera lineal con la profundidad de la red y el tamaño del kernel- Procesar secuencias muy largas requiere de un red muy profunda o filtros muy grandes- La convolución dilatada reduce la complejidad del modelo al realizar convolución sobre entradas submuestreadas con una cierta frecuencia²- Factor de dilatación d que aumenta exponencialmente en cada capa para que el campo receptivo de la red aumente, permitiendo capturar secuencias mucho más largas

¹ Emplear filtros de distintos tamaños

² P.ej., un factor de dilatación $d = 2$ aplica una convolución en las entradas submuestreadas cada dos pasos

Campo receptivo de una GNN y over-smoothing

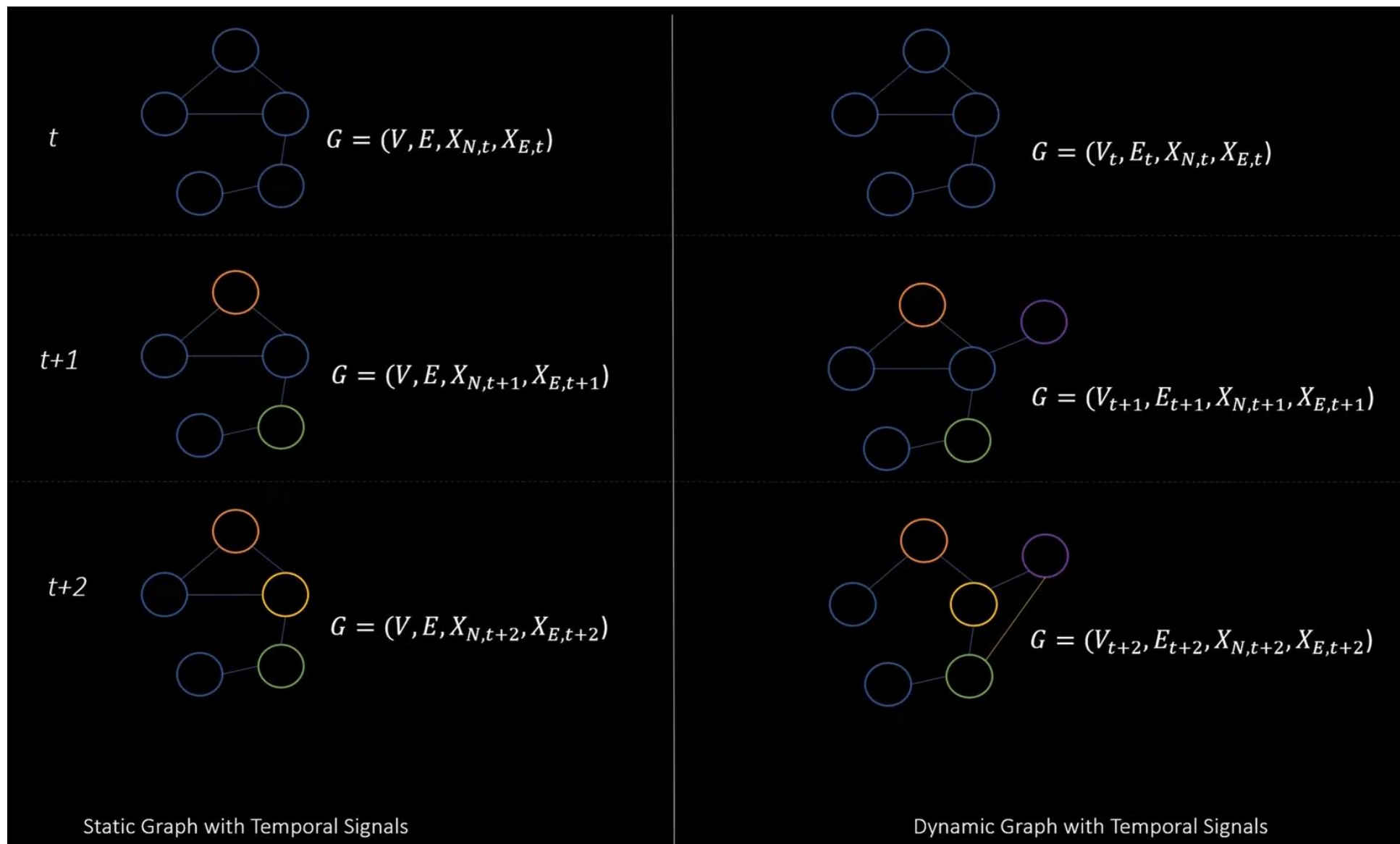
- El campo receptivo es el conjunto de nodos que determinan el *embedding* de un nodo de interés
- En una GNN de k capas, cada nodo tiene un campo receptivo de vecindad de k saltos
 - El número de vecinos compartidos crece rápidamente al incrementar el número de saltos



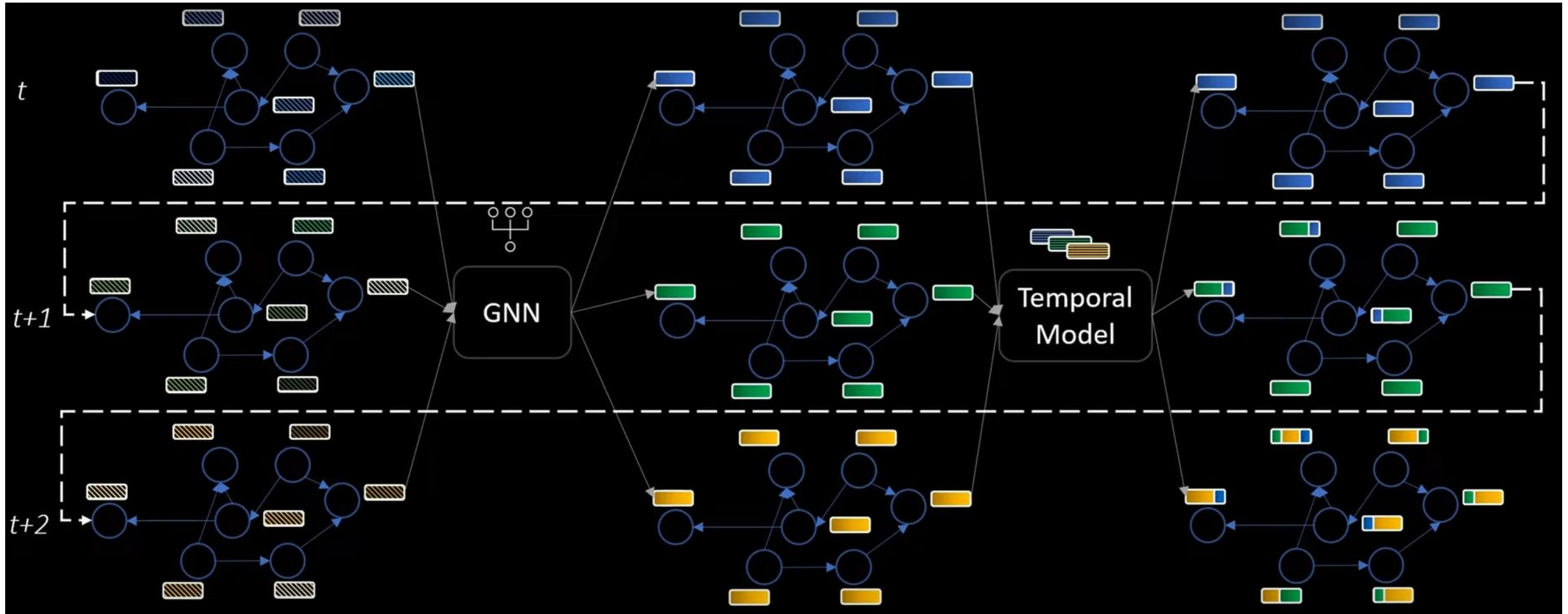
Over-smoothing

- Si apilas muchas capas GNN → los nodos tendrán sus campos receptivos altamente superpuestos → por lo que sus *embeddings* serán similares → over-smoothing

Tipos de ST-GNN

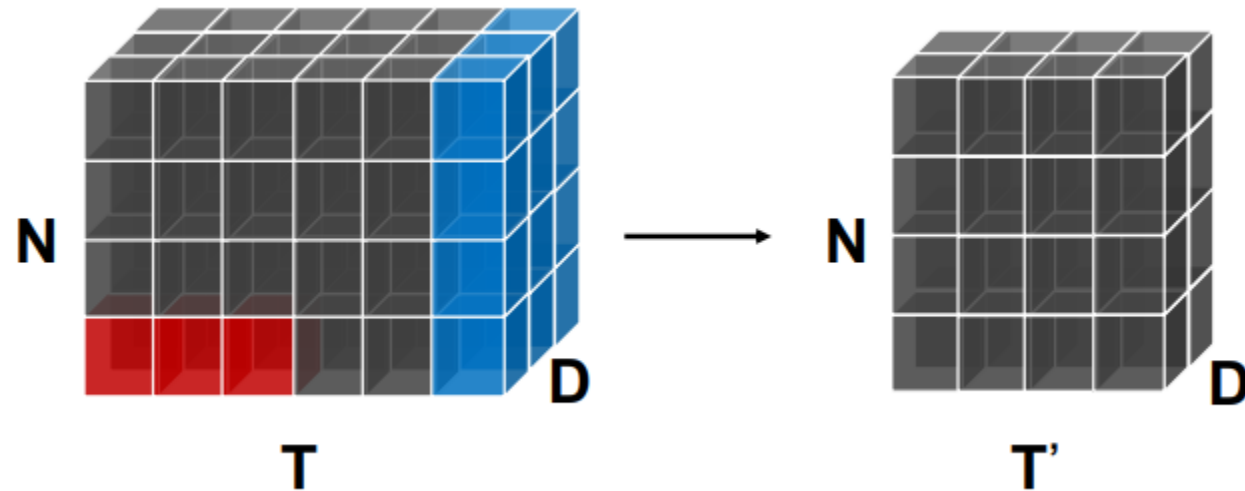


ST-GNN



Colaboración del módulo temporal con el módulo de GCN

- El módulo de convolución temporal filtra las entradas al deslizar una ventana 1D sobre el tiempo y los nodos
- El módulo del GCN filtra las entradas en cada paso



Experimentos

- Se utilizaron 5 métricas de evaluación: MAE, RMSE, MAPE, RRSE, CORR
- Se evaluaron 2 modelos:
 - MTGNN¹
 - MTGNN+sampling²

- Un paso (single-step)
- Múltiples pasos (multi-step)

Bases de datos empleadas para los experimentos

Datasets	# Samples	# Nodes	Sample Rate	Input Length	Output Length
traffic	17,544	862	1 hour	168	1
solar-energy	52,560	137	10 minutes	168	1
electricity	26,304	321	1 hour	168	1
exchange-rate	7,588	8	1 day	168	1
metr-la	34272	207	5 minutes	12	12
pems-bay	52116	325	5 minutes	12	12

Benchmark de series de tiempo multivariantes

Benchmark de estructura de grafos predefinidas para ST-GNN


¹ Modelo propuesto

² Modelo con entrenamiento en un subconjunto submuestreado de un grafo en cada iteración

Resultados multi-step

Comparación con otros ST-GNN modelos

	Horizon 3			Horizon 6			Horizon 12		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
METR-LA									
DCRNN	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.60	10.50%
STGCN	2.88	5.74	7.62%	3.47	7.24	9.57%	4.59	9.40	12.70%
Graph WaveNet	2.69	5.15	6.90%	3.07	6.22	8.37%	3.53	7.37	10.01%
ST-MetaNet	2.69	5.17	6.91%	3.10	6.28	8.57%	3.59	7.52	10.63%
MRA-BGCN	2.67	5.12	6.80%	3.06	6.17	8.30%	3.49	7.30	10.00%
GMAN	2.77	5.48	7.25%	3.07	6.34	8.35%	3.40	7.21	9.72%
MTGNN	2.69	5.18	6.86%	3.05	6.17	8.19%	3.49	7.23	9.87%
MTGNN+sampling	2.76	5.34	5.18%	3.11	6.32	8.47%	3.54	7.38	10.05%
PEMS-BAY									
DCRNN	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
STGCN	1.36	2.96	2.90%	1.81	4.27	4.17%	2.49	5.69	5.79%
Graph WaveNet	1.30	2.74	2.73%	1.63	3.70	3.67%	1.95	4.52	4.63%
ST-MetaNet	1.36	2.90	2.82%	1.76	4.02	4.00%	2.20	5.06	5.45%
MRA-BGCN	1.29	2.72	2.90%	1.61	3.67	3.80%	1.91	4.46	4.60%
GMAN	1.34	2.82	2.81%	1.62	3.72	3.63%	1.86	4.32	4.31%
MTGNN	1.32	2.79	2.77%	1.65	3.74	3.69%	1.94	4.49	4.53%
MTGNN+sampling	1.34	2.83	2.83%	1.67	3.79	3.78%	1.95	4.49	4.62%

- **Rendimiento similar** a las ST-GNN en el SOAT
 - **DCRNN, STGCN, MRA-BGCN dependen totalmente de grafos definidos**
 - **Graph WaveNet propone una matriz de adyacencia auto adaptativa, pero necesita combinarse con un grafo predefinido**
 - **ST-MetaNet emplea mecanismos de atención para los ajustar los pesos de las aristas de un grafo definido**
- 
- **Cuando la estructura del grafo no está definida, estos métodos no modelan eficazmente los datos de las series de tiempo multivariantes**

Estudios particulares

Ablation Study

- Estudio sobre el conjunto de datos METR-LA para entender la **contribución de los componentes en el modelo**
- Cada experimento se repitió **10 veces por 50 épocas**. Reportando el MAE, RMSE, MAPE con la desviación estándar sobre 10 ejecuciones en el conjunto de prueba.
 - w/o GC:** Remplazo de la GC con una capa lineal
 - w/o Mix-Hop:** Omisión del paso de selección de información, i.e., la salida del paso de propagación pasa directamente al siguiente módulo
 - w/o Inception:** Se mantiene el mismo número de canales de salida pero utilizando un filtro único de 1x7
 - w/ CL:** Entrenamiento sin aprendizaje por currículum

Methods	MTGNN	w/o GC	w/o Mix-hop	w/o Inception	w/o CL
MAE	2.7715±0.0119	2.8953±0.0054	2.7975±0.0089	2.7772±0.0100	2.7828±0.0105
RMSE	5.8070±0.0512	6.1276±0.0339	5.8549±0.0474	5.8251±0.0429	5.8248±0.0366
MAPE	0.0778±0.0009	0.0831±0.0009	0.0779±0.0009	0.0778±0.0010	0.0784±0.0009

Estudio sobre la capa de aprendizaje de grafos (Graph Learning Layer)

- Estudio para comparar la eficiencia de la capa de aprendizaje de grafos en comparación con otras formas en la literatura

Methods	Equation	MAE	RMSE	MAPE
Pre-defined-A	-	2.9017±0.0078	6.1288±0.0345	0.0836±0.0009
Global-A	$\mathbf{A} = \text{ReLU}(\mathbf{W})$	2.8457±0.0107	5.9900±0.0390	0.0805±0.0009
Undirected-A	$\mathbf{A} = \text{ReLU}(\tanh(\alpha(\mathbf{M}_1\mathbf{M}_1^T)))$	2.7736±0.0185	5.8411±0.0523	0.0783±0.0012
Directed-A	$\mathbf{A} = \text{ReLU}(\tanh(\alpha(\mathbf{M}_1\mathbf{M}_2^T)))$	2.7758±0.0088	5.8217±0.0451	0.0783±0.0006
Dynamic-A	$\mathbf{A}_t = \text{SoftMax}(\tanh(\mathbf{X}_t\mathbf{W}_1)\tanh(\mathbf{W}_2^T\mathbf{X}_t^T))$	2.8124±0.0102	5.9189±0.0281	0.0794±0.0008
Uni-directed-A (ours)	$\mathbf{A} = \text{ReLU}(\tanh(\alpha(\mathbf{M}_1\mathbf{M}_2^T - \mathbf{M}_2\mathbf{M}_1^T)))$	2.7715±0.0119	5.8070±0.0512	0.0778±0.0009