Использование графовых нейронных сетей для анализа динамических экономических систем

Васильев Максим

Москва, 2025

План презентации

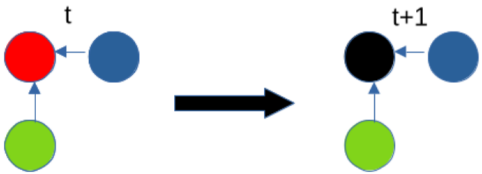
Актуальность

- Современные экономические системы обладают сложной динамической структурой.
- Традиционные (линейные) методы анализа и не учитывают сетевую природу взаимодействий и плохо работают с нелинейными зависимостями.
- Графовые нейронные сети позволяют моделировать связи между экономическими агентами и отслеживать их эволюцию во времени.

Графовые нейронные сети

Определение

Графовая нейронная сеть (GNN) — это модель, извлекающая признаки из данных, представленных в виде графа.



Графовые нейронные сети

Общее правило обновления признаков узла:

$$h_{t+1}(v) = \tau_t\big(h_t(v),\ \mathrm{AGGR}_t\big(\{h_t(u) : u \in \mathcal{N}(v)\}\big)\big)$$

- $h_t(v)$ — вектор признаков узла v на слое t.
- $\mathcal{N}(v)$ — множество соседей узла v, т.е. всех узлов, связанных с ним ребром.
- $\mathrm{AGGR}_t(\cdot)$ — агрегирующая функция, объединяющая информацию от соседей:
- $\tau_t(\cdot)$ — функция обновления признаков узла:
- $h_{t+1}(v)$ — новое представление узла, использующее информацию как самого узла, так и его соседей.

Графовые сверточные нейронные сети (GCN)

Основная идея

Графовая сверточная сеть обобщает идею обычной свёртки на структуру графа. Каждая вершина обновляет своё представление, усредняя признаки своих соседей.

$$H_{l+1} = \sigma\big(D^{-1/2} A D^{-1/2} H_l W_l\big), \tag{1}$$

где

- $H_l \in \mathbb{R}^{n \times d_l}$ — матрица признаков на слое l, n — число узлов, d_l — размерность признаков;
- $\sigma(\cdot)$ — функция активации (например, ReLU), действует поэлементно: $\sigma : \mathbb{R} \to \mathbb{R}$;
- $D = \mathrm{diag}(d_1, \ldots, d_n)$ — диагональная матрица степеней вершин, $d_i = \sum_j A_{ij}$;
- $A \in \mathbb{R}^{n \times n}$ — матрица смежности с петлями, элементы $A_{ij} \in \{0, 1\}$;
- $W_l \in \mathbb{R}^{d_l \times d_{l+1}}$ — матрица весов слоя, действительные значения.

Сложность добавления нового ребра в GCN

При появлении нового ребра изменяется нормализованная матрица $\hat{A} \in \mathbb{R}^{N \times N}$, поэтому необходимо пересчитать всю операцию:

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l)}).$$

Размеры матриц:

$$\hat{A} \in \mathbb{R}^{N \times N}, \quad H^{(l)} \in \mathbb{R}^{N \times d}, \quad W^{(l)} \in \mathbb{R}^{d \times d}.$$

1. **Пересчёт** $\hat{A}H$: $O(N^2 d)$
2. **Пересчёт** $(\hat{A}H)W$:$O(Nd^2)$

Итоговая сложность пересчёта всего слоя:

$$\boxed{O(N^2 d + Nd^2)}$$

Графовые сети внимания (GAT)

Основная идея

Каждая вершина обновляет своё представление, взвешивая вклад соседей через механизм внимания.

$$h_i' = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \, W h_j, \quad e_{ij} = a^\top [W h_i \,\|\, W h_j], \quad \alpha_{ij} = \mathrm{softmax}_j(e_{ij})$$

- $h_i \in \mathbb{R}^d$ — вектор признаков вершины i;
- $W \in \mathbb{R}^{d \times d'}$ — матрица линейного преобразования;
- $a \in \mathbb{R}^{2d'}$ — вектор параметров механизма внимания;
- $e_{ij} \in \mathbb{R}$ — ненормализованная важность соседа j;
- $\alpha_{ij} \in \mathbb{R}$, — нормализованное внимание;
- $\mathcal{N}(i)$ — множество соседей вершины i.

Сложность добавления нового ребра в GAT

При появлении нового ребра (u, v) в GAT с d - размер h и k - кол-во соседей, необходимо пересчитать:

1. Линейное преобразование признаков:

$$O(d^2)$$

2. Вычисление коэффициентов внимания:

$$e_{uv} = a^\top [Wh_u \| Wh_v] \Rightarrow O(d) \text{ на одно ребро}$$

3. Нормализация softmax по соседям:

$$O(k)$$

4. Обновление эмбеддинга узла:

$$O(kd)$$

Итоговая сложность добавления одного ребра: $\boxed{O(d^2 + kd)}$

Граф, признаки и Attention-score

Граф:

$$1-2-3, \qquad \hat{A}=\begin{bmatrix}1&1&0\\1&1&1\\0&1&1\end{bmatrix}, \qquad X=\begin{bmatrix}1&0\\0&1\\1&1\end{bmatrix}, \qquad W=\begin{bmatrix}1&2\\3&4\end{bmatrix}, \quad a=[1,1,1,1]$$

После линейного преобразования:

$$h_1=[1,2], \quad h_2=[3,4], \quad h_3=[4,6]$$

Полученные веса ребер:

$$e_{11}=6,\ e_{12}=10$$
$$e_{21}=10,\ e_{22}=14,\ e_{23}=17$$
$$e_{32}=17,\ e_{33}=20$$

Веса внимания и итоговое обновление узлов

Softmax по соседям:

$$\alpha_{11} = 0.018,\ \alpha_{12} = 0.982$$
$$\alpha_{21} = 0.002,\ \alpha_{22} = 0.121,\ \alpha_{23} = 0.877$$
$$\alpha_{32} = 0.047,\ \alpha_{33} = 0.953$$

Обновление:

$$h_i' = \sum_{j \in N(i)} \alpha_{ij} h_j$$

Результаты:

$$h_1' = [2.964,\ 3.964]$$
$$h_2' = [3.873,\ 5.750]$$
$$h_3' = [3.953,\ 5.906]$$

GAT обучаемо выбирает, на кого смотреть сильнее.

GraphSAGE (SAmple and aggreGatE)

Основная идея

GraphSAGE обучает генеративную функцию, агрегируя информацию из локального окружения с помощью обучаемой агрегирующей функции.

$$h_v^{(k)} = \sigma\Big(W^{(k)} \cdot \Big(h_v^{(k-1)} \,\|\, \mathrm{AGGREGATE}^{(k)}\big(\{\, h_u^{(k-1)} : u \in \mathcal{N}(v)\}\big)\Big)\Big) \tag{2}$$

- $h_v^{(k)} \in \mathbb{R}^{d_k}$ — представление вершины v на слое k;
- $\mathcal{N}(v)$ — множество соседей вершины v;
- $\mathrm{AGGREGATE}^{(k)} : \mathbb{R}^{|\mathcal{N}(v)| \times d_{k-1}} \to \mathbb{R}^{d_{k-1}}$ (mean, max, LSTM);
- $W^{(k)} \in \mathbb{R}^{(2d_{k-1}) \times d_k}$ — матрица весов слоя;
- $\sigma : \mathbb{R} \to \mathbb{R}$ — нелинейная активация.

Сложность добавления одного ребра в GraphSAGE

При появлении нового ребра (u, v) с размером эмбединга узла - d и сколичеством соседей - k требуется обновить представления узлов через:

1. Стоимость агрегации соседей:

$$O(kd) \quad \text{(Mean Aggregator)}, \quad O(kd^2) \quad \text{(LSTM / Pooling Aggregator)}$$

2. Стоимость обновления представления узла:

$$O(d^2)$$

Итоговая сложность добавления одного ребра (обновление узлов u и v):

$$\boxed{O(kd + d^2)} \quad \text{(Mean)}, \quad \boxed{O(kd^2)} \quad \text{(LSTM / Pooling)}$$

Граф и признаки узлов

Граф:

$$1 - 2 - 3$$

Матрица смежности:

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

Признаки узлов:

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Используем **mean-aggregator**.

Агрегация соседей и конкатенация

Соседи:

$$N(1) = \{2\}, \quad N(2) = \{1, 3\}, \quad N(3) = \{2\}$$

Агрегация:

$$\text{AGGREGATE}(1) = [0, 1], \quad \text{AGGREGATE}(2) = [1, 0.5], \quad \text{AGGREGATE}(3) = [0, 1]$$

Конкатенация:

$$h_1^{cat} = [1, 0,\ 0, 1]$$
$$h_2^{cat} = [0, 1,\ 1, 0.5]$$
$$h_3^{cat} = [1, 1,\ 0, 1]$$

Весовая матрица:

$$W = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{bmatrix}$$

Обновление узлов и итоговые эмбеддинги

Обновление:

$$h_i' = \sigma(W h_i^{cat})$$

Вычисления:

$$h_1' = \text{ReLU}([3, 1])$$

$$h_2' = \text{ReLU}([2, 2.5])$$

$$h_3' = \text{ReLU}([3, 2])$$

Итоговые представления узлов:

$$h_1' = [3, 1], \quad h_2' = [2, 2.5], \quad h_3' = [3, 2]$$

GraphSAGE: обучаемая агрегация + конкатенация признаков.

Temporal Graph Networks (TGN)

- TGN — Graph Neural Network для **динамических графов**.
- Граф изменяется во времени, события приходят последовательно:

$$\mathcal{E} = \{(u, v, t, x_{uv}(t))\}$$

  где u, v — узлы, t — время события, $x_{uv}(t)$ — признаки ребра.
- TGN позволяет:
  - Обрабатывать потоки событий
  - Динамически обновлять состояния узлов
  - Получать эмбеддинги для предсказания будущих событий

Основные модули TGN

1. **Memory (Память узлов)**: хранит текущее состояние узла $m_v(t)$, отражающее историю взаимодействий.
2. **Message Function**: формирует сообщение для события на основе текущих состояний узлов и признаков ребра:

$$\text{msg}_{uv}(t) = f(m_u(t^-), m_v(t^-), x_{uv}(t))$$

3. **Memory Updater**: обновляет память узлов после каждого события, часто используется GRU или LSTM:

$$m_v(t) = \text{GRU}(m_v(t^-), \text{msg}_{uv}(t))$$

4. **Embedding Module**: генерирует эмбеддинги узлов на текущий момент времени:

$$z_v(t) = \text{AGGREGATE}(m_v(t), \{m_u(t) | u \in N(v, t)\})$$

Memory (Память узлов)

- Каждому узлу v соответствует вектор состояния $m_v(t)$
- Начальное состояние: $m_v(0) = 0$ или случайное
- После события память обновляется с помощью Memory Updater
- Смысл: сохраняет историю узла и отражает актуальное состояние

Message Function и Memory Updater

- **Message Function**:
  - Формирует сообщение для каждого события
  - Пример: u лайкнул v:

$$\text{msg}_{uv}(t) = \text{MLP}([m_u, m_v, x_{uv}])$$

- **Memory Updater**:
  - Принимает старое состояние и сообщение
  - GRU/LSTM обновляет память:

$$m_v(t) = \text{GRU}(m_v(t^-), \text{msg}_{uv}(t))$$

Embedding Module и поток работы TGN

- **Embedding Module**:

$$z_v(t) = \text{AGGREGATE}\big(m_v(t), \{m_u(t) | u \in N(v,t)\}\big)$$

- Агрегатор может быть: mean, max, attention
- Поток работы:
  1. Событие: (u,v,t)
  2. Message Function формирует msg
  3. Memory Updater обновляет память узлов u и v
  4. Embedding Module генерирует эмбеддинги $z_u(t), z_v(t)$

TGN — эффективная модель для динамических графов с потоками событий.

Обработка нового события в TGN

При появлении нового сообщения (события) (u, v, t, x_{uv}) TGN выполняет:

1. **Message Function**: формирует сообщение $\text{msg}_{uv}(t)$
2. **Memory Updater**: обновляет память узлов u и v
3. **Embedding Module**: обновляет эмбеддинги узлов с учетом соседей
4. **Опциональная агрегация**: если используется attention или pooling по соседям

Сложность по шагам

Обозначения:

- d — размер эмбеддинга узла
- k — среднее количество соседей
- $C_{\text{msg}} \sim O(d^2)$ — формирование сообщения (MLP)
- $C_{\text{update}} \sim O(d^2)$ — обновление памяти (GRU/LSTM)
- $C_{\text{agg}} \sim O(kd)$ — агрегация соседей

Шаги:

Message Function: $O(d^2)$

Memory Updater (2 узла): $O(d^2)$

Embedding Module с соседями: $O(kd)$

Итоговая сложность

Суммарная сложность одного события:

$$O_{\text{event}} = O(d^2 + kd)$$

- При больших эмбеддингах $d \gg k$ доминирует $O(d^2)$
- При очень высокой степени узлов $k \gg d$ доминирует $O(kd)$
- Для батчей из B событий:

$$O_{\text{batch}} = O(B(d^2 + kd))$$

- TGN эффективен, так как обновляются только локальные узлы, а не весь граф

Исходные данные

Граф: $1 - 2 - 3$
Событие: узел 1 взаимодействует с узлом 2 в момент $t = 1$
Память узлов (Memory):

$$m_1(0) = [0, 0], \quad m_2(0) = [0, 0], \quad m_3(0) = [0, 0]$$

Признаки ребра:

$$x_{12} = [1, 0]$$

Time Encoding:

$$\phi(1) = [\sin(1), \cos(1)] \approx [0.841, 0.540]$$

Message Function: обучаемая матрица W_{msg}:

$$W_{\text{msg}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Формирование сообщения и обновление памяти

Сообщение (Message Function):

$$[m_1 \| m_2 \| x_{12} \| \phi(1)] = [0, 0, 0, 0, 1, 0, 0.841, 0.540]$$

$$\text{msg}_{12} = W_{\text{msg}} \cdot [\dots]^\top = [1.841, 0.540]$$

Обновление памяти (Memory Updater):

$$m_1(1) = \text{ReLU}(m_1(0) + \text{msg}_{12}) = [1.841, 0.540]$$

$$m_2(1) = \text{ReLU}(m_2(0) + \text{msg}_{12}) = [1.841, 0.540]$$

$$m_3(1) = m_3(0) = [0, 0]$$

Агрегация соседей (Embedding Module) и итог

Mean aggregator:

$$z_v(t) = \text{mean}(m_v(t), \{m_u(t) | u \in N(v)\})$$

Эмбеддинги узлов после события:

$$z_1(1) = \text{mean}([1.841, 0.540], [1.841, 0.540]) = [1.841, 0.540]$$

$$z_2(1) = \text{mean}([1.841, 0.540], [1.841, 0.540], [0, 0]) \approx [1.227, 0.360]$$

$$z_3(1) = \text{mean}([0, 0], [1.841, 0.540]) = [0.920, 0.270]$$

Итог: Эмбеддинги учитывают память узлов, сообщения и время события.

Сравнение GCN, GAT, GraphSAGE и TGN

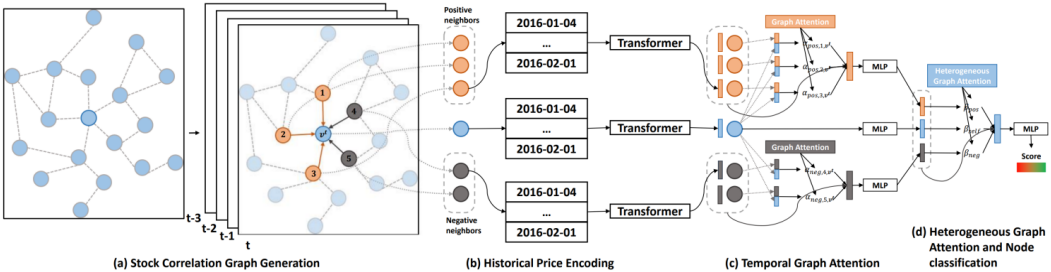
Модель	Тип графа	Учёт времени	Сложность
GCN	Статический	Нет	$O(N^2d + Nd^2)$
GAT	Статический	Нет	$O(d^2 + kd)$
GraphSAGE	Статический / индуктивный	Нет	$O(kd^2)$
TGN	Динамический	Да	$O(d^2 + kd)$

TGN единственный учитывает *время*, память и поток событий.

Применение: THGNN для финансовых временных рядов

Модель и цель

Авторы предлагают модель THGNN для прогнозирования движений цен акций, учитывая и динамику временных рядов, и гетерогенные отношения между компаниями.



Архитектура THGNN

Гибридная структура

- **Temporal Transformer** – извлечение долгосрочных зависимостей временных рядов
- **Heterogeneous GAT** – учитывает разные типы экономических связей:
  - Отраслевые
  - Корреляционные
  - Капитализационные группы
- **Multi-head attention** – выделяет ключевые связи между компаниями

Такой подход позволяет учитывать как временную зависимость с помощью транформеров, так и структурную с помощью гетерогенных GAT

Данные и признаки

Источники данных

- Дневные котировки компаний из **SP500** и **CSI300** (2016–2021)
- Граф: статический на весь период обучения

Признаки

- OHLC (Open, High, Low, Close)
- Объемы торгов
- Волатильность
- Рыночные факторы

Цель модели

Классификация направления движения цены на следующий день

Результаты

	S&P 500							CSI 300						
	ACC	ARR	AV	MDD	ASR	CR	IR	ACC	ARR	AV	MDD	ASR	CR	IR
LSTM	0.532	0.377	0.449	-0.382	0.842	0.989	0.954	0.515	0.291	**0.318**	-0.240	0.915	1.213	0.877
GRU	0.530	0.362	**0.445**	-0.379	0.813	0.955	0.934	0.517	0.312	0.320	-0.243	0.975	1.284	0.932
Transformer	0.533	0.385	0.454	-0.384	0.848	1.005	0.960	0.518	0.327	0.322	-0.245	1.016	1.335	0.969
eLSTM	0.534	0.434	0.454	-0.373	0.955	1.163	1.041	0.520	0.330	0.323	-0.239	1.022	1.381	0.991
LSTM+GCN	0.538	0.470	0.442	-0.354	1.062	1.326	1.103	0.523	0.351	0.320	**-0.217**	1.097	1.618	1.119
LSTM+RGCN	0.565	0.558	0.463	-0.366	1.205	1.522	1.203	0.536	0.509	0.326	-0.235	1.561	2.166	1.537
TGC	0.552	0.528	0.455	**-0.344**	1.163	1.535	1.180	0.531	0.453	0.323	-0.224	1.402	2.022	1.412
MAN-SF	0.551	0.527	0.467	-0.357	1.130	1.478	1.157	0.527	0.418	0.334	-0.225	1.251	1.858	1.282
HATS	0.541	0.494	0.466	-0.387	1.060	1.277	1.110	0.525	0.385	0.332	-0.249	1.160	1.546	1.116
REST	0.549	0.502	0.466	-0.359	1.079	1.398	1.117	0.528	0.425	0.331	-0.228	1.284	1.864	1.298
AD-GAT	0.564	0.535	0.457	-0.371	1.170	1.444	1.187	0.539	0.537	0.329	-0.240	1.632	2.238	1.596
THGNN	**0.579**	**0.665**	0.468	-0.369	**1.421**	**1.804**	**1.340**	**0.551**	**0.632**	0.336	-0.237	**1.881**	**2.667**	**1.875**

Ограничения и комментарии

Ограничения

- Статичный граф на весь период обучения
- Высокая вычислительная стоимость (Transformer + GAT)
- Предсказывает только направление изменения цены
- Требуется аккуратная калибровка типов рёбер

Применение: гибрид LSTM-GNN для прогнозирования акций

Модель и цель

Авторы предлагают гибридную модель, комбинирующую LSTM для временных рядов и GNN для графового анализа взаимосвязей между акциями.

Источники данных

- Исторические временные ряды цен акций
- Expanding-window валидация: постепенное расширение обучающей выборки
- Граф корреляций между активами обновляется по мере роста окна

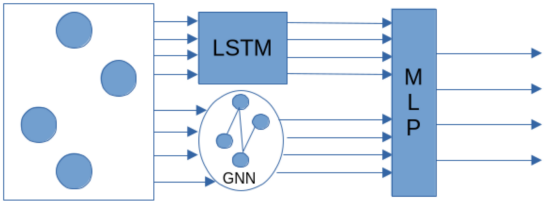
Признаки

- Цена закрытия, открытие, максимум, минимум
- Объем торгов
- Корреляционные и ассоциативные связи между активами

Архитектура LSTM-GNN

Гибридная структура

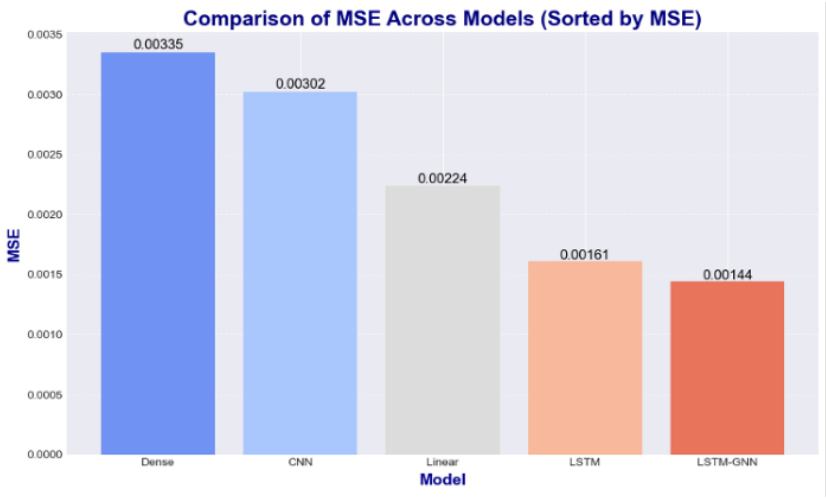
- **LSTM** – анализ временных рядов (цены, объемы)
- **GNN** – граф корреляций и ассоциативных связей между активами
- **Граф обновляется** по мере расширения обучающей выборки (expanding-window)
- Используется адаптивная перекалибровка модели



Особенности модели

- LSTM эффективно извлекает временные зависимости
- GNN учитывает структуру рынка и корреляционные связи
- Граф динамически обновляется с увеличением данных
- Адаптивная перекалибровка повышает устойчивость модели
- Решает регрессионную задачу — предсказывает числовую цену, а не только направление

Результаты


Comparison of MSE Across Models (Sorted by MSE)
MSE
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0.00335
0.00302
0.00224
0.00161
0.00144
Dense
CNN
Linear
LSTM
LSTM-GNN
Model

Ограничения и комментарии

Ограничения

- Значительные вычислительные ресурсы
- Чувствительность к выбору окна и порогов корреляции
- GNN может быть шумным в периоды турбулентности рынка
- Все модели сталкиваются с проблемой статичности графов и ограниченного учёта рыночных режимов

Комментарии

- Интеграция LSTM и GNN повышает точность прогнозов цен
- Отличается от предыдущих работ тем, что решает регрессионную задачу
- Сравнение подходов показывает, что графовые модели стабильно улучшают прогноз благодаря учёту рыночной структуры

Выводы

- **Графовые нейронные сети (GNN)** позволяют учитывать не только индивидуальные признаки агентов, но и **структуру взаимодействий** в экономических системах.
- **GraphSAGE и GAT** эффективны для агрегации соседних узлов и внимания к значимым связям.
- **Динамические GNN (TGN)** учитывают **временную значимость событий**.
- **Гибридные модели LSTM-GNN** объединяют:
  - **LSTM** — захват временных закономерностей каждого узла,
  - **GNN** — учет межузловых связей,
  - **MLP слои** — изучение сложных нелинейных взаимодействий.
- Гибридные модели позволяют улучшить точность прогнозирования финансовых временных рядов.

Литература

1. Sonani et al. *Stock Price Prediction Using a Hybrid LSTM-GNN Model: Integrating Time-Series and Graph-Based Analysis*, 2025.
2. Kipf TN, Welling M. *Semi-supervised classification with graph convolutional networks*, 2017.
3. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. *Graph Attention Networks*, ICLR, 2018.
4. Hamilton WL, Ying R, Leskovec J. *Inductive Representation Learning on Large Graphs*, 2017.
5. Sheng Xiang, Dawei Cheng, Chencheng Shang, Ying Zhang, Yuqi Liang. *Temporal and Heterogeneous Graph Neural Network for Financial Time Series Prediction*, 2022.
6. Rossi E, Chamberlain B, Eynard D, et al. *Temporal Graph Networks for Deep Learning on Dynamic Graphs*, 2020.