



Neural Networks and Deep Learning

Lecture 9: Graph neural networks



Prof. dr. Daniel Traian PELE

Bucharest University of Economic Studies (ASE), Department of Statistics and Econometrics
Romanian Academy, Institute for Economic Forecasting (IPE)
IDA Institute Digital Assets ▪ AI4EFin ▪ MSCA Digital Finance

Learning objectives

- Represent financial data as a graph: nodes, edges, node features
  - Adjacency, degree and the normalised adjacency matrix.
- Write the message-passing scheme of a graph neural network (GNN) and its three classic layers: graph convolutional network (GCN), graph sample and aggregate (GraphSAGE), graph attention network (GAT).
- Train them on the Elliptic Bitcoin graph for anti-money laundering (AML), in time order, against a logistic regression and a random forest.
- Judge the results like an investigator: precision, recall, the area under the precision-recall curve (PR-AUC), alerts per analyst, drift and stability of explanations.

Abbreviations

AML	anti-money laundering
ASE	Bucharest University of Economic Studies
BTC	bitcoin (currency unit)
CNN	convolutional neural network
CPU	central processing unit
GAT	graph attention network
GCN	graph convolutional network
GIN	graph isomorphism network
GNN	graph neural network
GraphSAGE	graph sample and aggregate
IPE	Institute for Economic Forecasting, Romanian Academy
MLP	multilayer perceptron
MPNN	message-passing neural network
PCA	principal component analysis
PR-AUC	area under the precision-recall curve
PyG	PyTorch Geometric (library)
ReLU	rectified linear unit
SHAP	SHapley Additive exPlanations
WL	Weisfeiler–Leman graph isomorphism test
XAI	explainable artificial intelligence

Outline

- Graphs in the detection of financial crime
- Graphs as matrices
- Message passing
- GCN, GraphSAGE and GAT
- Case study: illicit Bitcoin transactions
- GNNs in production
- Wrap-up

Network structure in money-laundering detection

⊡ **In one sentence:** A single payment can look ordinary; the chain of payments around it often does not.

$$G = (\mathcal{V}, \mathcal{E}, \mathbf{X}), \qquad \mathbf{X} \in \mathbb{R}^{n \times d}, \qquad y_v \in \{0, 1\}$$

⊡ **where**

- $\mathcal{V}$ = nodes (transactions), $n = |\mathcal{V}|$
- $\mathcal{E}$ = edges (a payment flow from one transaction to the next)
- $\mathbf{X}$ = node features, d per node
- $y_v = 1$ if transaction v is illicit.

⊡ **Elliptic graph** (Weber et al., 2019)

- 203 769 Bitcoin transactions, 234 355 payment flows
- 165 features per transaction, 49 time steps

⊡ **The task**

- Node classification: illicit (scams, ransomware, dark markets) or licit

⊡ **In practice:** a bank's anti-money-laundering team already thinks in chains of payments

- A GNN learns from the same chains.

(Weber et al., 2019; Bishop & Bishop, 2024, ch. 13)

Experiment yourself: notebook §1

Description of the Elliptic data set

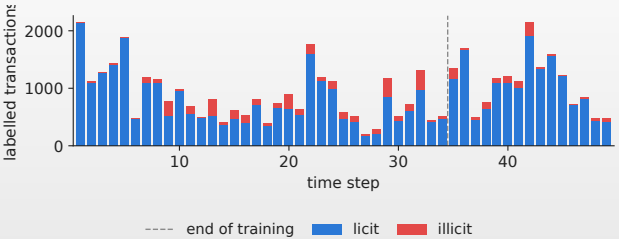
Element	Value in our copy of the data
Nodes, edges	203 769 transactions; 234 355 directed payment flows
Labels	4 545 illicit, 42 019 licit, 157 205 unknown (22.9 % labelled)
Features	93 local (amounts, fees, inputs, outputs) + 72 aggregated over the neighbours
Time	49 steps of about two weeks; edges join transactions of the same step (100.0 %)
Training / test	steps 1–34 (29 894 labelled) / steps 35–49 (16 670 labelled, 1 083 illicit)

- **Imbalance**: only 9.76 % of the labelled transactions are illicit
  - Accuracy alone would be meaningless.
- **Anonymised**: the feature names are hidden by Elliptic
  - We read them as $f_1, \ldots, f_{165}$.

(Weber et al., 2019)

Experiment yourself: notebook §1

Illicit transactions over time



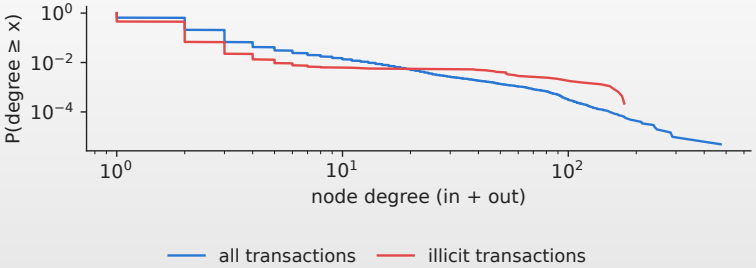
Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: Elliptic Bitcoin transaction graph, 49 time steps (about two weeks each).

- **Peak**: 342 illicit transactions in step 32
  - About 4 159 transactions per step in total.
- **Split in time**: train on steps 1–34, test once on steps 35–49, as in Weber et al. (2019)
  - Never shuffle the steps.

In practice: a model is judged on the months after its training data, exactly as it will be used.

NN_L9_elliptic_timesteps

Degree distribution of the transaction graph



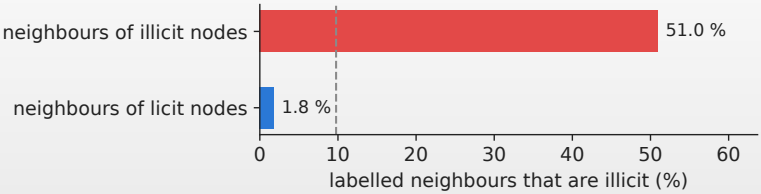
Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: Elliptic Bitcoin transaction graph, 49 time steps (about two weeks each).

- **Sparse graph**: mean degree 2.30, median 2
  - 34.5 % of the nodes have one neighbour only.
- **Heavy tail**: the largest hub has 473 neighbours
  - A GNN must not let hubs drown everyone else (hence the normalisation).

$\deg(v) = |\mathcal{N}(v)|$

NN_L9_degree_dist

Homophily of illicit transactions



Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: Elliptic Bitcoin transaction graph, 49 time steps (about two weeks each).

- **Homophily**: 51.0 % of the labelled neighbours of an illicit node are illicit, against 1.8 % for a licit node.
- **Edge homophily** 95.4 % of the labelled edges join the same class
  - With random labels: 82.4 %.

- **In practice:** homophily is the reason to look at the neighbours: who you trade with says something about you.

NN_L9_homophily

Outline

- Graphs in the detection of financial crime
- Graphs as matrices
- Message passing
- GCN, GraphSAGE and GAT
- Case study: illicit Bitcoin transactions
- GNNs in production
- Wrap-up

The adjacency matrix as an algebraic representation of the graph

In one sentence: Every operation of a GNN is a product of the feature matrix with a (normalised) adjacency matrix and a weight matrix.

$$A_{uv} = \begin{cases} 1, & (u,v) \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases}, \qquad D = \operatorname{diag}\left(\sum_{v} A_{uv}\right)$$

$$\tilde{A} = A + I, \qquad \hat{A} = \tilde{D}^{-1/2}\tilde{A}\,\tilde{D}^{-1/2}$$

where

- A = adjacency, symmetrised
- D = degree matrix
- $\tilde{A}$ adds self-loops (a node keeps its own features)
- $\hat{A}$ has entries $1/\sqrt{\tilde{d}_u \tilde{d}_v}$.

Sparse storage

- A has $n^2 \approx 4 \cdot 10^{10}$ cells but only 234 355 edges
- Libraries keep an edge list $2 \times |\mathcal{E}|$, as in PyTorch Geometric (PyG)

Why normalise

- Without it a hub of degree 473 would add 473 messages and explode the scale

In practice: a bank's payment network is stored as an edge list, never as a dense matrix.

(Kipf & Welling, 2017)

Experiment yourself: notebook §2

Permutation invariance and equivariance of graph models

- **In one sentence:** Transaction 17 and transaction 4,203 are names, not positions; relabelling them must not change any prediction.

$$f\left(P\mathbf{X},\ PAP^{\top}\right) = P\, f(\mathbf{X}, A) \quad \forall P$$

- **where**
  - P = any $n \times n$ permutation matrix
  - f returns one output row per node: *equivariance*
  - A graph-level output must be *invariant*: $g(P\mathbf{X}, PAP^{\top}) = g(\mathbf{X}, A)$.

- **Why not a multilayer perceptron (MLP)?**
  - On the rows of A it would need n inputs and change with the numbering
- **Why not a convolutional neural network (CNN)?**
  - It assumes a grid with a fixed number of ordered neighbours
  - A node has between 1 and 473 neighbours, in no order

- **In practice:** ask of any vendor model: does the score change if the input file is sorted differently? It must not.

(Bishop & Bishop, 2024, sec. 13.1)

Smooth signals on a graph: the Laplacian

- **In one sentence:** The graph Laplacian measures how much a node signal differs between neighbours; GCN layers are low-pass filters of it.

$$L = D - A, \qquad \mathbf{x}^\top L\,\mathbf{x} = \sum_{(u,v)\in\mathcal{E}} \left(x_u - x_v\right)^2 \geq 0$$

$$L = U\Lambda U^\top, \qquad \hat{\mathbf{x}} = U^\top \mathbf{x} \ \text{(graph Fourier transform)}$$

- **where**
  - $\mathbf{x} \in \mathbb{R}^n$ = one feature on all nodes
  - Small $\mathbf{x}^\top L\mathbf{x}$ = neighbours look alike (homophily)
  - U, Λ = eigenvectors and eigenvalues of L (frequencies).

- **Spectral filter**
  - $g_\theta \star \mathbf{x} = U\, g_\theta(\Lambda)\, U^\top \mathbf{x}$
  - Costly: U is dense, $O(n^2)$
- **GCN**
  - First-order polynomial in L gives $\hat{A}$ of the previous slide
  - Derivation in Appendix A

- **In practice:** homophilous fraud graphs are smooth signals: averaging over neighbours removes noise, not information.

(Kipf & Welling, 2017; Hamilton, 2020, ch. 7)

Three prediction tasks: node, edge and graph level

- **In one sentence:** The same message-passing layers serve three kinds of business questions; only the read-out changes.

$$\hat{y}_v = \sigma(\mathbf{w}^\top \mathbf{h}_v), \quad \hat{y}_{uv} = \sigma(\mathbf{h}_u^\top \mathbf{h}_v), \quad \hat{y}_G = \sigma\left(\mathbf{w}^\top \sum_v \mathbf{h}_v\right)$$

- **where**
  - $\mathbf{h}_v$ = representation of node v after the last layer
  - σ = logistic function
  - The sum makes the graph output permutation-invariant.

- **Node**
  - Is this transaction illicit? (this lecture)
- **Edge**
  - Will these two firms trade; is this payment a mule transfer?
- **Graph**
  - Is this cluster of accounts a laundering ring; is this interbank network fragile?

- **In practice:** define the unit of decision first (a payment, a client, a ring), then the graph task.

(Hamilton, 2020, ch. 1)

Outline

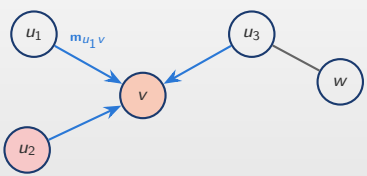
- Graphs in the detection of financial crime
- Graphs as matrices
- Message passing
- GCN, GraphSAGE and GAT
- Case study: illicit Bitcoin transactions
- GNNs in production
- Wrap-up

A GNN layer: message, aggregation and update functions

- **In one sentence:** Each node sends a message to its neighbours; each node sums what it receives and updates its own state.

$$\mathbf{m}_v^{(l)} = \bigoplus_{u \in \mathcal{N}(v)} \phi^{(l)}\left(\mathbf{h}_u^{(l)}, \mathbf{h}_v^{(l)}, \mathbf{e}_{uv}\right)$$

$$\mathbf{h}_v^{(l+1)} = \psi^{(l)}\left(\mathbf{h}_v^{(l)}, \mathbf{m}_v^{(l)}\right), \qquad \mathbf{h}_v^{(0)} = \mathbf{x}_v$$



- **where**
  - $\mathcal{N}(v)$ = neighbours of v
  - ϕ = message function
  - $\bigoplus$ = permutation-invariant aggregator (sum, mean, max)
  - ψ = update (a small network)
  - $\mathbf{e}_{uv}$ = edge features, if any.

- **Message-passing neural network (MPNN)**: every layer of this lecture is a choice of ϕ, $\bigoplus$, ψ.

- **In practice:** the weights ϕ, ψ are shared by all nodes: the model size does not grow with the graph.

(Gilmer et al., 2017; Bishop & Bishop, 2024, sec. 13.2)

Experiment yourself: notebook §3

The aggregator and the expressive power of a GNN

- **In one sentence:** Mean and max forget how many neighbours sent a message; the sum keeps it, which makes it as powerful as the Weisfeiler–Leman test.

$$\mathbf{h}_v^{(l+1)} = \mathrm{MLP}^{(l)}\left((1+\varepsilon)\,\mathbf{h}_v^{(l)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(l)}\right)$$

- **where**
  - Graph isomorphism network (GIN) layer
  - ε = learnable scalar
  - A sum over a multiset is injective if the MLP is rich enough, so GIN separates any two neighbourhoods that the Weisfeiler–Leman (WL) test separates.

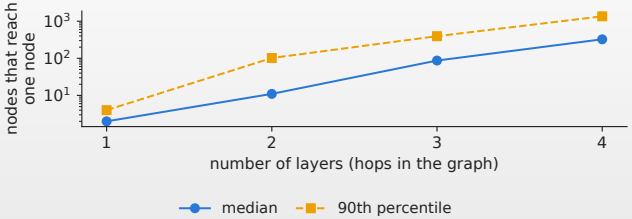
- **Example**
  - Neighbours $\{a, a\}$ and $\{a\}$: same mean, same max, different sum

- **In fraud**
  - "one payment from a mixer" and "ten payments from mixers" should not look the same
  - But sums of hubs need normalisation, which is why GCN uses $\hat{A}$

- **In practice:** choose the aggregator for the question: counts matter (sum), proportions matter (mean), one red flag matters (max).

(Xu et al., 2019)

Each layer widens the neighbourhood that reaches a node



Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: Elliptic Bitcoin transaction graph, 49 time steps (about two weeks each).

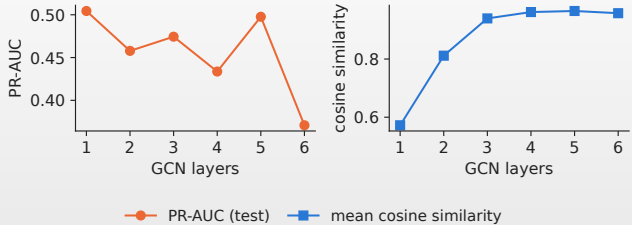
- **Receptive field** of a test transaction: median 2 nodes after 1 layer, 11 after 2, 325 after 4.
- **Tail**: the 90th percentile reaches 1 356 nodes after 4 layers
  - Hubs connect distant parts of the graph.

- **In practice:** L layers $=$ L hops of money flow
  - Two hops already cover most of what an investigator would check by hand.

(own computation, 2,000 test nodes)

NN_L9_receptive_field

Over-smoothing in deep graph networks



Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: own estimates; training steps 1-34, test time steps 35-49.

- **Accuracy**: PR-AUC 0.504 with 1 GCN layer, 0.458 with 2, 0.371 with 6
  - Best depth 1.
- **Similarity**: mean cosine similarity of node representations rises from 0.57 (1 layer) to 0.96 (6 layers)
  - Repeated averaging drives them to a common vector.

- **In practice:** GNNs are shallow by design: two or three layers, then residual links or jumping knowledge if more is needed.

(own computation, 2 seeds; Li, Han & Wu, 2018)

NN_L9_oversmoothing

Outline

- Graphs in the detection of financial crime
- Graphs as matrices
- Message passing
- GCN, GraphSAGE and GAT
- Case study: illicit Bitcoin transactions
- GNNs in production
- Wrap-up

GCN: degree-weighted average of the neighbours followed by a linear map

In one sentence: A GCN layer is a linear regression on the normalised average of a node and its neighbours, followed by a rectified linear unit (ReLU).

$$H^{(l+1)} = \mathrm{ReLU}\left(\hat{A}\, H^{(l)} W^{(l)}\right)$$

$$\mathbf{h}_v^{(l+1)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{W^{(l)\top} \mathbf{h}_u^{(l)}}{\sqrt{\tilde{d}_u \tilde{d}_v}}\right)$$

where

- $H^{(l)} \in \mathbb{R}^{n \times d_l}$ = node representations, $H^{(0)} = \mathbf{X}$
- $W^{(l)}$ = weights shared by all nodes
- $\mathrm{ReLU}(x) = \max(0, x)$
- Output: $\mathrm{softmax}(\hat{A} H^{(1)} W^{(2)})$.

Our GCN

- 2 layers, 100 hidden units, dropout 0.2: 26 902 parameters
- Full-batch training on the whole graph, 66 s per seed on the processor (CPU)

Cost

- $O(|\mathcal{E}|\, d_l + n\, d_l d_{l+1})$ per layer: linear in the edges

In practice: a GCN is the graph analogue of a moving average: robust, cheap, and blind to which neighbour matters most.

(Kipf & Welling, 2017)

Experiment yourself: notebook §3

GraphSAGE: separate node representation and neighbour sampling

- **In one sentence:** GraphSAGE concatenates a node's own state with the mean of a sample of its neighbours, so it also works on nodes never seen in training.

$$\mathbf{h}_v^{(l+1)} = \mathrm{ReLU}\left(W_1^{(l)} \mathbf{h}_v^{(l)} + W_2^{(l)} \frac{1}{|\mathcal{S}(v)|} \sum_{u \in \mathcal{S}(v)} \mathbf{h}_u^{(l)} \right)$$

- **where**
  - $\mathcal{S}(v) \subseteq \mathcal{N}(v)$ = a random sample of at most k neighbours (all of them in our full-batch run)
  - W_1, W_2 = separate weights for the node and for its neighbourhood.

- **Inductive**
  - The weights do not depend on the graph: tomorrow's transactions get a score without retraining
- **Scalable**
  - Mini-batches of k-hop samples: cost bounded by k^L per node, not by the hub degree
- **On Elliptic**
  - Illicit F1 0.564 (sd 0.022), PR-AUC 0.582

- **In practice:** payment systems that score transactions in real time need inductive layers such as GraphSAGE.

(Hamilton, Ying & Leskovec, 2017)

Experiment yourself: notebook §4

GAT: learned attention weights over the neighbours

- **In one sentence:** A graph attention layer weights each neighbour by a learned score: the attention of Lecture 7, restricted to the edges of the graph.

$$\alpha_{vu} = \frac{\exp\left(\mathrm{LeakyReLU}(\mathbf{a}^\top [W\mathbf{h}_v \,\|\, W\mathbf{h}_u])\right)}{\sum_{k \in \mathcal{N}(v)\cup\{v\}} \exp\left(\mathrm{LeakyReLU}(\mathbf{a}^\top [W\mathbf{h}_v \,\|\, W\mathbf{h}_k])\right)}$$

$$\mathbf{h}_v^{(l+1)} = \Big\|_{k=1}^{K} \ \mathrm{ReLU}\left(\sum_u \alpha_{vu}^k W^k \mathbf{h}_u\right)$$

- **where**
  - $\mathbf{a}$ = attention vector
  - $\|$ = concatenation
  - K = number of heads (4 here)
  - Weights α_{vu} sum to one over the neighbours of v.

- **Link with transformers**
  - A transformer is a GAT on the complete graph of the tokens
- **On Elliptic**
  - 27 302 parameters; illicit F1 0.489, PR-AUC 0.496

- **In practice:** attention weights are a first, not a final, explanation: they show where the model looked, not why.

(Veličković et al., 2018)

Experiment yourself: notebook §4

Training on one big graph: masks, class weights, early stopping

- **In one sentence:** All nodes pass messages, but only labelled training nodes enter the loss; validation steps decide when to stop.

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{V}_{tr}|} \sum_{v \in \mathcal{V}_{tr}} \Big[w_1 \, y_v \log \hat{p}_v + w_0 (1 - y_v) \log(1 - \hat{p}_v) \Big]$$

- **where**
  - $\mathcal{V}_{tr}$ = labelled nodes of steps 1–29
  - $\hat{p}_v$ = predicted probability of illicit
  - $w_1 = 0.7$, $w_0 = 0.3$ = class weights of Weber et al. (2019)
  - Adam, learning rate 0.003, weight decay $5 \cdot 10^{-4}$.

- **Validation**
  - Labelled nodes of steps 30–34: best epoch by PR-AUC, patience 40
  - Median best epoch of the GCN: 47

- **Leakage-free scaling**
  - Feature means and standard deviations from steps 1–34 only

- **In practice:** the unknown nodes are not wasted: they carry messages between labelled ones (semi-supervised learning).

(Kipf & Welling, 2017; Weber et al., 2019)

Experiment yourself: notebook §3

Outline

- Graphs in the detection of financial crime
- Graphs as matrices
- Message passing
- GCN, GraphSAGE and GAT
- Case study: illicit Bitcoin transactions
- GNNs in production
- Wrap-up

Eight models evaluated on a chronological test set

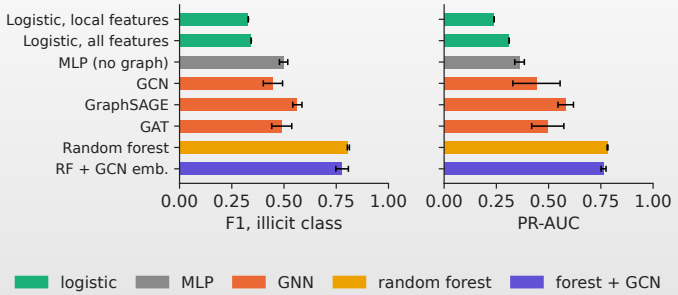
Model	What it sees
Logistic regression	local features only (93), then all 165
Random forest	all features; 100 trees, 50 features per split (Weber et al., 2019)
MLP (no graph)	all features, same layers as the GCN but no messages: isolates the value of the edges
GCN, GraphSAGE, GAT	all features + the graph; 2 layers, 100 units
Random forest + GCN	all features + the 100 hidden GCN outputs of each node

- **Metrics on steps 35–49**: illicit-class precision, recall and F1 at threshold 0.5
  - PR-AUC (threshold-free).
- **Uncertainty**: 3 seeds per network and per forest
  - Bootstrap of the test nodes for the differences.

(Weber et al., 2019)

Experiment yourself: notebook §2

Results on Elliptic: the random forest and the contribution of graph layers



Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: own estimates; training steps 1-34, test time steps 35-49.

- **Illicit F1**: random forest 0.808, forest + GCN 0.778, GCN 0.447, MLP 0.498, logistic 0.343.
- **Value of the edges**: PR-AUC of the GCN minus the MLP, same layers: +8.1 points.

(own computation; mean ± sd over 3 seeds)

NN_L9_model_comparison

Precision, recall and the cost of training

Model	Prec.	Recall	F1	PR-AUC	Sec.
Logistic, local	0.208	0.789	0.329	0.239	0.8
Logistic, all	0.220	0.771	0.343	0.311	1.8
MLP	0.517	0.483	0.498	0.361	59.2
GCN	0.484	0.494	0.447	0.442	65.5
GraphSAGE	0.569	0.561	0.564	0.582	143.2
GAT	0.488	0.500	0.489	0.496	231.2
Random forest	0.914	0.725	0.808	0.782	4.9
Forest + GCN	0.933	0.668	0.778	0.763	186.0

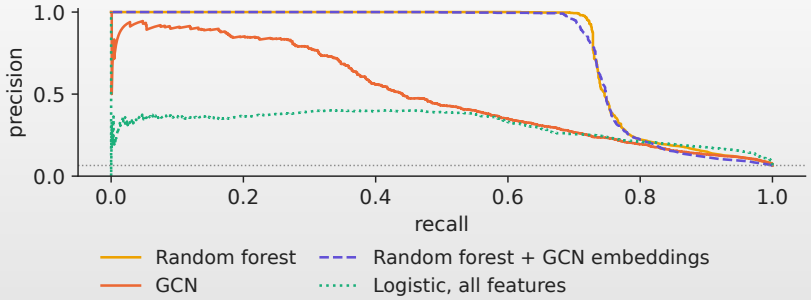
- **Reading the table**
  - Precision = share of alerts that are illicit; recall = share of illicit found
  - Green = best, red = worst in the column
- **Forest**
  - Precision 0.914, recall 0.725
- **Logistic, class weights 0.3/0.7**
  - Precision 0.220, recall 0.771: the weights buy recall with false alarms

- **In practice:** the right model depends on what an error costs: a missed laundering case or an analyst's wasted hour.

(own computation; sec. = training time per seed, CPU)

Experiment yourself: notebook §2

Precision-recall curves of the models



Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: own estimates; training steps 1-34, test time steps 35-49.

- **PR-AUC**: random forest 0.782, forest + GCN 0.763, GCN 0.442, logistic 0.311
  - A random score would give 6.50 % (dotted line).

PR-AUC $= \sum_k (R_k - R_{k-1}) P_k$

NN_L9_pr_curves

Bootstrap test of the gap on the test transactions

- **In one sentence:** One test set gives one number; resampling the test nodes shows how much that number could move.

$$\Delta^{*b} = \text{PR-AUC}_A\left(\mathcal{T}^{*b}\right) - \text{PR-AUC}_B\left(\mathcal{T}^{*b}\right), \quad b = 1, \ldots, B$$

$$\text{CI}_{95\%} = \left[q_{0.025}(\Delta^*),\ q_{0.975}(\Delta^*)\right]$$

- **where**
  - $\mathcal{T}^{*b}$ = test nodes drawn with replacement
  - $B = 1{,}000$
  - Scores = mean over 3 seeds
  - The same draw is used for both models (paired).

- **Forest minus GCN**
  - $\Delta = 0.285$, 95 % interval [0.259; 0.313]
- **Forest + GCN minus forest**
  - $\Delta = -0.008$, 95 % interval $[-0.015; -0.002]$
- an interval that excludes 0 supports a real difference on this test period

- **In practice:** report the interval, not only the winner: a fraud committee decides on ranges, not on the third decimal.

(Efron & Tibshirani, 1993)

Experiment yourself: notebook §2

Neighbourhood aggregates in the features and the performance of the random forest

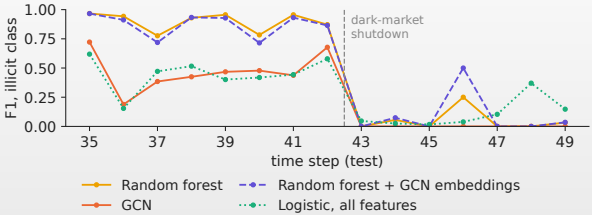
- **Hand-made message passing**
  - 72 of the 165 features are aggregates over the one-hop neighbours (max, min, sd, correlation)
  - This is one fixed GNN layer, computed by Elliptic before release
- **Local features only**
  - Logistic regression F1 0.329 vs 0.343 with the aggregates
- **Trees**
  - Thresholds on amounts and counts fit fraud patterns that are rules ("more than k inputs")
- **Where GNNs earn their keep**
  - Raw graphs without engineered aggregates
  - Multi-hop patterns (layering through many accounts)
  - Inductive scoring of new nodes, and embeddings for other models
- **Combine**
  - Forest + GCN representations: F1 change -3.0 points

- **In practice:** before training a GNN, check whether the data provider has already aggregated the neighbourhood for you.

(Weber et al., 2019)

Experiment yourself: notebook §2

Concept drift after a dark-market shutdown



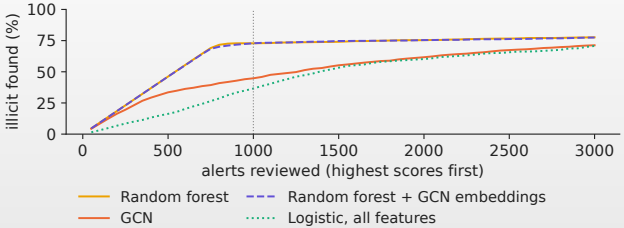
Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: own estimates; training steps 1-34, test time steps 35-49.

- **Before step 43**: mean F1 random forest 0.898, GCN 0.473
  - **from step 43**: 0.048 and 0.000.
- **New behaviour**: after a major dark market closed, illicit flows changed form
  - No model trained on steps 1–34 had seen it (Weber et al., 2019).

- **In practice:** monitor F1 per period and retrain on recent labels
  - A static AML model ages within weeks.

NN_L9_f1_by_timestep

Illicit transactions found among the top 1,000 alerts



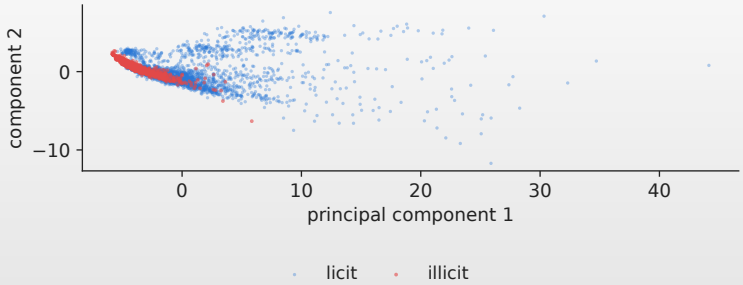
Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: own estimates; training steps 1-34, test time steps 35-49.

- **Top 1 000 scores** of the test period contain 72.9 % of all illicit transactions for the random forest, 44.8 % for the GCN, 72.9 % for forest + GCN.
- **Hit rate** of those alerts: 78.9 % (forest), 48.5 % (GCN), out of 1 083 illicit test transactions.

- **In practice:** size the team from this curve: the alert budget, not the F1, is the constraint in a compliance department.

NN_L9_alert_budget

Illicit transactions cluster in the GCN embedding space



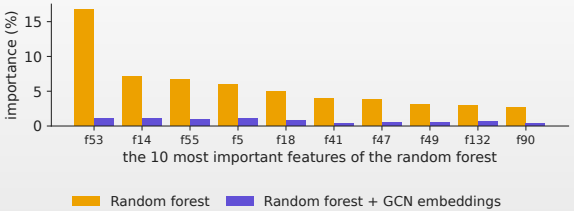
Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: own estimates; training steps 1-34, test time steps 35-49.

- **Hidden layer** of the GCN (100 dimensions) projected by principal component analysis (PCA) on its first two components: 79.2 % of the variance.
- **Reuse**: these vectors are the inputs of the forest + GCN model and can feed any other risk model.

(own computation; 1 083 illicit and 3,000 licit test nodes)

NN_L9_gcn_embeddings

Effect of graph features on the explanations



Source: Elliptic Bitcoin transaction graph (Weber et al., 2019). Data: own estimates; training steps 1-34, test time steps 35-49.

- **Importance mass**: the 10 top features of the forest carry 58.2 % of the importance
  - With GCN embeddings they keep 7.4 %, the embeddings take 73.9 %.
- **Stability across 3 refits**: mean Spearman correlation of the importances 0.973 (forest) and 0.896 (forest + GCN).

- **In practice:** compare explanations across refits before reading a change of model as a change of explanation (next: research from our group).

NN_L9_importance_shift

Outline

- Graphs in the detection of financial crime
- Graphs as matrices
- Message passing
- GCN, GraphSAGE and GAT
- Case study: illicit Bitcoin transactions
- GNNs in production
- Wrap-up

Common errors in applying graph models in finance

- **Leakage through edges**
  - A test node must not receive messages from the future
  - In Elliptic 100.0 % of the edges stay inside one time step, so the temporal split is clean
- **Few labels**
  - Only 22.9 % of the nodes are labelled; labels come from past investigations and are biased towards cases already found
- **Drift**
  - Criminals adapt: the shutdown at step 43 changed the patterns
  - Temporal GNNs (EvolveGCN) let the weights evolve with time
- **Explainability**
  - A suspicious-activity report needs a reason a regulator can read
  - GNNExplainer finds the subgraph and features behind one prediction

- **In practice:** test a graph model the way it will run: on later periods, with the labels available at that time.

(Pareja et al., 2020; Ying et al., 2019)

Applications of graph models in finance

- **Payments and cards**
  - Cardholder–merchant graphs: a compromised terminal links many fraud victims
- **Credit**
  - Guarantor and supplier networks: default spreads along trade credit
- **Crypto**
  - Address and transaction graphs (this lecture), mixer detection, wallet clustering
- **Systemic risk**
  - Interbank exposures and tail-dependence networks: who falls if one bank fails
- **Markets**
  - Graphs of stocks linked by correlation, supply chain or ownership, for return forecasts
- **Energy**
  - Power grids and interconnected price zones are graphs by construction

In practice: whenever the risk of one unit depends on its counterparties, a graph model is worth a try.

(Hamilton, 2020; Bishop & Bishop, 2024, ch. 13)

Outline

- Graphs in the detection of financial crime
- Graphs as matrices
- Message passing
- GCN, GraphSAGE and GAT
- Case study: illicit Bitcoin transactions
- GNNs in production
- Wrap-up

Key takeaways

- **A GNN layer** = message, permutation-invariant aggregation, update
  - GCN averages with degree weights, GraphSAGE separates self and neighbours, GAT learns the weights.
- **Depth** = hops: two layers reach a median of 11 transactions
  - More layers over-smooth (PR-AUC 0.371 with 6 layers).
- **On Elliptic** the edges help a network (GCN vs MLP: +8.1 PR-AUC points), but a random forest on pre-aggregated features leads (F1 0.808).
- **Judge AML models** on later periods, with PR-AUC, alert budgets and bootstrap intervals
  - Watch drift and the stability of explanations.

Research from our group: graphs, fraud and explanations

- **Explanation stability in graph-augmented fraud detection** (Diaconu & Pele, 2026): treats the complete model refit as the unit of inference and compares SHapley Additive exPlanations (SHAP) of fraud models with and without graph embeddings against graph-free controls matched in dimension and covariance.
- **Key results**: augmentation can shrink the attributions of the original features mechanically
  - After accounting for retraining variability, graph-free controls reproduce most of the movement in marginal SHAP stability, while local, row-level rankings change.
- **Energy Financial Risk Meter** (Toma, Dima, Grecu, Pele & Lessmann, 2026): measures systemic risk in the European Union energy sector from the network of tail dependencies between firms.

Findings as stated in the papers; the numbers on the other slides come from our own computations.

(Bucharest University of Economic Studies; Romanian Academy, Institute for Economic Forecasting)

Further reading and resources

- **Bishop & Bishop (2024)**, chapter 13: graph neural networks, from message passing to graph attention.
- **Hamilton (2020)**, *Graph Representation Learning*, free draft: cs.mcgill.ca/~wlh/grl_book.
- **PyTorch Geometric** tutorials: pytorch-geometric.readthedocs.io.
- **Elliptic data set** and the original paper: arXiv:1908.02591.
- **This course**: code, data, notebooks — github.com/danpele/NEURAL_BIZ.
- **Experiment yourself**: the notebook reproduces every chart and number of this lecture.

Quiz 1: permutations

- We renumber the transactions of the graph.
  - What must happen to the GCN scores?

Answer

- They are permuted in the same way and are otherwise unchanged: a GNN layer is permutation-equivariant, because the aggregation over neighbours does not depend on order.

Quiz 2: depth

- Why not stack ten GCN layers to see ten hops of money flow?

Answer

- Repeated degree-weighted averaging makes all representations similar (over-smoothing): on Elliptic PR-AUC falls from 0.458 (2 layers) to 0.371 (6 layers).

Quiz 3: the strong benchmark

- Why does a random forest without the graph beat the GCN on Elliptic?

Answer

- 72 of its features are already aggregates over the neighbours, a hand-made message-passing layer
  - Trees also fit rule-like fraud thresholds well.

Quiz 4: evaluation

- Only 6.50 % of the test transactions are illicit.
  - Which metric would mislead a compliance officer?

Answer

- Accuracy: labelling everything licit is right most of the time and finds no criminal
  - Use precision, recall, PR-AUC and the alert budget.

Quiz 5: attention on graphs

- What is the link between a GAT layer and the transformer of Lecture 7?

Answer

- Both weight neighbours by a softmax of learned scores
  - A transformer is attention on the complete graph of the tokens, a GAT restricts it to the edges of the graph.

References (1/2)

- Bishop, C. M. & Bishop, H. (2024). *Deep Learning: Foundations and Concepts*, ch. 13. Springer. doi:10.1007/978-3-031-45468-4
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. doi:10.1023/A:1010933404324
- Diaconu, D. & Pele, D. T. (2026). SHAP stability across random and forward-time validation in graph-augmented fraud detection. Submitted to *Expert Systems with Applications*. Code: github.com/QuantLet/GNN_XAI_Stability
- Efron, B. & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman & Hall. doi:10.1201/9780429246593
- Fey, M. & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. ICLR Workshop on Representation Learning on Graphs and Manifolds. arXiv:1903.02428
- Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. (2017). Neural message passing for quantum chemistry. *Proceedings of ICML*, PMLR 70, 1263–1272. arXiv:1704.01212
- Hamilton, W. L., Ying, R. & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30. arXiv:1706.02216
- Hamilton, W. L. (2020). *Graph Representation Learning*. Morgan & Claypool. doi:10.2200/S01045ED1V01Y202009AIM046
- Kipf, T. N. & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*. arXiv:1609.02907

References (2/2)

- Li, Q., Han, Z. & Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1). doi:10.1609/aaai.v32i1.11604
- Lundberg, S. M. & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30. arXiv:1705.07874
- Pareja, A., Domeniconi, G., Chen, J. et al. (2020). EvolveGCN: Evolving graph convolutional networks for dynamic graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(4), 5363–5370. doi:10.1609/aaai.v34i04.5984
- Toma, F.-M., Dima, D.-M., Grecu, R., Pele, D. T. & Lessmann, S. (2026). Energy Financial Risk Meter: Systemic risk in the EU energy sector. SSRN Working Paper 6985301. doi:10.2139/ssrn.6985301
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. & Bengio, Y. (2018). Graph attention networks. *International Conference on Learning Representations*. arXiv:1710.10903
- Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T. & Leiserson, C. E. (2019). Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics. KDD Workshop on Anomaly Detection in Finance. arXiv:1908.02591
- Xu, K., Hu, W., Leskovec, J. & Jegelka, S. (2019). How powerful are graph neural networks? *International Conference on Learning Representations*. arXiv:1810.00826
- Ying, R., Bourgeois, D., You, J., Zitnik, M. & Leskovec, J. (2019). GNNExplainer: Generating explanations for graph neural networks. *Advances in Neural Information Processing Systems*, 32. arXiv:1903.03894

Appendix A: from spectral filters to the GCN layer

$$g_\theta \star \mathbf{x} \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{L})\,\mathbf{x}, \quad \tilde{L} = \tfrac{2}{\lambda_{\max}} L_{sym} - I$$

$$K = 1,\ \lambda_{\max} \approx 2: \quad g_\theta \star \mathbf{x} \approx \theta_0 \mathbf{x} - \theta_1 D^{-1/2} A D^{-1/2} \mathbf{x}$$

$$\theta = \theta_0 = -\theta_1: \quad \theta\,(I + D^{-1/2} A D^{-1/2})\,\mathbf{x} \ \rightarrow\ \theta\,\hat{A}\,\mathbf{x}$$

- T_k = Chebyshev polynomials
  - $L_{sym} = I - D^{-1/2} A D^{-1/2}$
  - A polynomial of degree K only mixes nodes within K hops.
- One shared parameter per feature pair avoids overfitting
  - The renormalisation trick $I + D^{-1/2} A D^{-1/2} \rightarrow \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$ keeps eigenvalues in $(-1, 1]$.
- With d input features and d' outputs, θ becomes the matrix $W \in \mathbb{R}^{d \times d'}$.

(Kipf & Welling, 2017)

Appendix B: the Weisfeiler–Leman test

$$c_v^{(l+1)} = \text{hash}\left(c_v^{(l)},\ \{\{c_u^{(l)} : u \in \mathcal{N}(v)\}\}\right)$$

- Start with the same colour for all nodes
  - Refine by hashing each node's colour with the multiset of its neighbours' colours.
- Two graphs with different colour histograms are not isomorphic
  - Equal histograms are inconclusive.
- A message-passing GNN is at most as discriminative as WL
  - GIN reaches this bound with sum aggregation and an injective update.

(Xu et al., 2019)

Appendix C: settings of the experiment

Setting	Value
Split	train steps 1–29, validation 30–34, test 35–49 (labelled nodes only in the loss)
Networks	2 layers, 100 units, ReLU, dropout 0.2; GAT: 4 heads of 25; Adam, rate 0.003, decay $5 \cdot 10^{-4}$; at most 300 epochs, patience 40
Class weights	0.3 / 0.7
Random forest	100 trees, 50 features per split, raw features
Seeds	1, 2, 3
Software, hardware	PyTorch, PyTorch Geometric, scikit-learn; laptop CPU, total 0.1 minutes

(Weber et al., 2019; Fey & Lenssen, 2019)