Advances in Continual Graph Learning for Anti-Money Laundering Systems: A Comprehensive Review

Bruno Deprez[*†], Wei Wei[‡], Wouter Verbeke[*],
Bart Baesens[*§], Kevin Mets[‡], Tim Verdonck[†*]

Article Type: Advanced Review
Correspondence: Tim Verdonck (tim.verdonck@uantwerpen.be)
Funding: This work was supported by the Research Foundation – Flanders (FWO research project 1SHEN24N) and by the BNP Paribas Fortis Chair in Fraud Analytics, and by the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme. The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.
Keywords: Continual Learning | Anti-Money Laundering | Graph Neural Networks | Fraud Detection | Catastrophic Forgetting

Abstract

Financial institutions are required by regulation to report suspicious financial transactions related to money laundering. Therefore, they need to constantly monitor vast amounts of incoming and outgoing transactions. Given the involvement of many parties in money laundering, graph analytics is vital for effective monitoring. A particular challenge in detecting money laundering is that money launderers continuously adapt their tactics to evade detection. Hence, detection methods need constant fine-tuning. Traditional machine learning models suffer from catastrophic forgetting when fine-tuning the model on new data, thereby limiting their effectiveness in dynamic environments. Continual learning addresses this issue and enhances current anti-money laundering (AML) practices, by allowing models to incorporate new information while retaining prior knowledge. Research on continual graph learning for AML, however, is still scarce. In this review, we critically evaluate state-of-the-art continual graph learning approaches for AML applications. We categorise methods into replay-based, regularization-based, and architecture-based strategies within the graph neural network (GNN) framework, and we provide in-depth experimental evaluations on both synthetic and real-world AML datasets that showcase the effect of the different hyperparameters. Our analysis demonstrates that continual learning improves model adaptability and robustness in the face of extreme class imbalances and evolving fraud patterns. Finally, we outline key challenges and propose directions for future research.

[*]KU Leuven
[†]University of Antwerp - imec
[‡]University of Antwerp - imec, IDLab
[§]University of Southampton

GRAPHIC FOR ONLINE TABLE OF CONTENTS

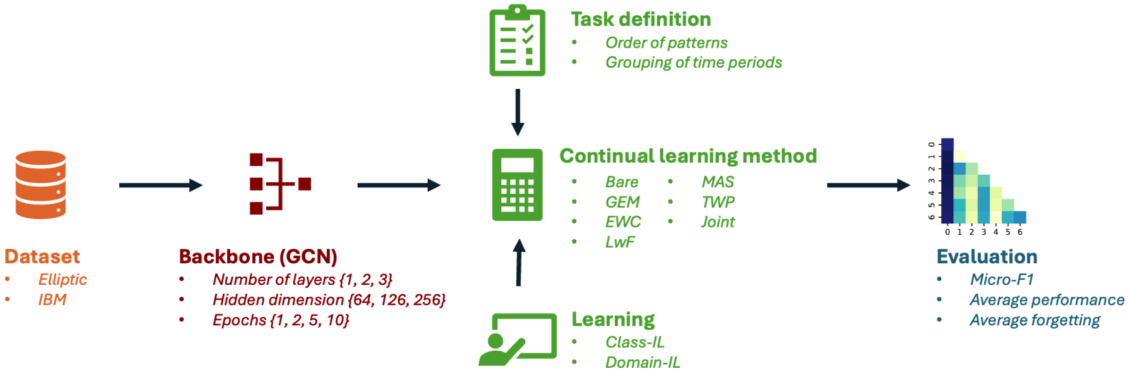

Figure: Pipeline of the comprehensive experiments on continual graph learning for anti-money laundering.

1 Introduction

Criminal enterprise activities generate income streams that cannot be directly used because of their illegal origins. Therefore, criminals launder money to make illicitly obtained funds appear legitimate (Levi and Reuter, 2006). The United Nations Office on Drugs and Crime (UNOCD) has estimated that an amount equal to about 2% to 5% of global GDP is laundered each year, amounting to USD 2 trillion. This money is used to expand criminal activity and to finance terrorism (Levi and Reuter, 2006), resulting in enormous socioeconomic pressure.

Generally, money laundering approaches involve three main steps (United Nations Office on Drugs and Crime (UNOCD); Levi and Reuter, 2006). During **placement**, illegal money enters the financial system, often in jurisdictions where regulation and enforcement are less strict. **Layering** involves mixing the illegal funds with legitimately obtained money across multiple transactions. This obscures the initial source of the money, making it harder to uncover its illegal origin. At **integration**, the money is spent on legitimate purchases. After completing this final step, the money is successfully laundered. Actual money laundering approaches may also include fewer or more steps.

The framework given above illustrates that money laundering involves many payments over multiple accounts, warranting the use of network analytics (Pramanik et al., 2017). Despite the strong case and the long history of the field, anti-money laundering (AML) has been receiving limited attention in research compared to other forms of financial fraud (Ngai et al., 2011; Kurshan and Shen, 2020). In recent literature reviews, a limited number of papers on anti-money laundering are discussed, and even fewer on network analytics for AML. A first aim of this work is to extend the application of machine learning and network analytics in AML.

A popular way of adopting network analytics is via graph neural networks (GNNs). GNNs are able to detect fraudulent behaviour by learning complex, non-linear patterns in network data (Motie and Raahemi, 2024), and have shown promising results for fighting financial crime (Motie and Raahemi, 2024) with adoption in credit card fraud detection (Van Belle et al., 2020; Van Belle et al., 2022; Van Belle and De Weerdt, 2024), insurance fraud detection (Óskarsdóttir et al., 2022; Deprez et al., 2024b), and anti-money laundering (Deprez et al., 2024a).

Financial networks and fraud characteristics evolve rapidly over time, necessitating the fine-tuning of these GNNs when new data comes in. This fine-tuning can cause the model to suffer from catastrophic forgetting (French, 1999; Carta et al., 2021; De Lange et al., 2022), where learning sequentially on *new* data while discarding *old* data leads to significant performance loss on earlier observations.

Continual learning, also known as incremental learning or lifelong learning, aims to mitigate the problem of catastrophic forgetting (Goodfellow et al., 2015). Research in this field typically aims to adapt regular deep learning methods or develop new methods that are able to accumulate and consolidate knowledge. One of the key assumptions that underlie continual learning, is that data is no longer (fully) accessible after models have been trained.

Continual learning is key for effective and dynamic AML. Financial institutions often face enormous transaction volumes that require continuous monitoring. However, they face computational constraints when implementing AML methods in practice. First, there is limited computing power and budget to update AML models, making periodical retraining from scratch impractical. Second, there are regulatory constraints on how much data can be stored and for how long. Finally, money laundering methods, as for other types of fraud, are evolving constantly (Baesens et al., 2015; Van Vlasselaer et al., 2015), so the distribution of illegitimate transactions changes over time. However, when training to detect these new tactics, models should be able to retain knowledge about old ones, in case these are used again. Otherwise, fraudsters could just rotate between tactics to evade detection.

Continual learning performs well under these constraints. First, it fine-tunes existing models, so limited additional training is required. Second, fine-tuning can be done using only the most recent data, so there is no need to store all historical data indefinitely. Finally, continual learning is specifically designed to retain previous knowledge when learning from new data, so older modi operandi should still be detected.

Another concern when developing AML models concerns their stability and robustness, whereby small variations in data and hyperparameters should not lead to significantly different outcomes. A lack of stability leads to a loss in the credibility and trustworthiness of the model. Continual learning incorporates model stability by design, where fine-tuning of the models is constrained to avoid catastrophic forgetting and to achieve a balance in the stability-plasticity trade-off.

Despite all this, research on continual graph learning for AML is rare. Furthermore, current experiments and benchmarks in literature lack an in-depth discussion on the effect of the many choices underlying the applied continual learning framework. These studies are also often limited to *simple* tasks, without the extreme class imbalance and label obfuscation as typically present in AML and other fraud detection problems. Therefore, this work sets out to answer the following research questions:

RQ1 What is the current state of the literature on continual graph learning for anti-money laundering?

RQ2 What is the impact of the hyperparameters of the GNN and continual learning methods on performance and forgetting?

RQ3 Which methods are best suited to overcome catastrophic forgetting for anti-money laundering?

To answer these questions, we conduct an in-depth literature review, summarising the current research on AML, continual learning for financial fraud detection, and previous work on the effect of the involved hyperparameters. This literature review is complemented by an extensive experimental study to analyse the performance and forgetting of AML network methods on two open-source datasets. The contributions of our work are, hence, as follows:

- We present an in-depth review of the current state-of-the-art in continual graph learning for anti-money laundering;

- We introduce and investigate the implications of continual graph learning on anti-money laundering, for edge as well as node classification;
- We present the result of extensive experiments on two open-source AML datasets and analyse the effects of various choices on performance.

The code of the presented experiments is publicly available on github[1] to facilitate peer researchers and practitioners to replicate and extend the reported results.

The remainder of this paper is organised as follows. Preliminary theory on graphs, graph neural networks and continual learning is discussed in Section 2. A literature review is presented in Section 3. Section 4 presents the experimental methodology, with results and discussion provided in Section 5. Section 6 concludes this work and presents directions for future research.

2 Preliminaries

2.1 Graphs

A graph $G(V, E)$ is defined via two sets, V and E. The elements of set $V = \{v_1, \ldots, v_n\}$ represent the nodes in the graph, while set $E \subset V \times V$ represents the edges that connect the nodes. An edge between node i and j is denoted as e_{ij}. In this work, we consider homogeneous networks. It is assumed that nodes are assigned a vector $x_i \in \mathbb{R}^m$ with feature values. The matrix containing all feature vectors is denoted by $X \in \mathbb{R}^{n \times m}$.

2.2 Graph Neural Networks

The initial idea of deep learning on graphs was introduced by Scarselli et al. (2005) and Scarselli et al. (2008), based on message passing. The idea of message passing is still present in GNNs, where a node's representation is updated iteratively based on the node's neighbours.

Formally, given a graph $G(V, E)$, graph neural networks (GNNs) construct the representation of node i at layer l, denoted by $h_i^{(l)}$, as

$$h_i^{(l)} = \phi^{(l-1)} \left(h_i^{(l-1)}, \sum_j \hat{A}_{ij} \psi^{(l-1)} \left(h_i^{(l-1)}, h_j^{(l-1)} \right) \right), \tag{1}$$

where $\phi^{(l)}$, and $\psi^{(l)}$ are layer-dependent functions, and $\hat{A}_{ij}$ is the normalized adjacency matrix, including self-loops. Most of the time, the initial embedding is set equal to the node features, $h_i^{(0)} = x_i$.

The most widely adopted graph neural networks are Graph Convolutional Networks (GCN) (Kipf and Welling, 2017), Graph SAmple and aggreGatE (GraphSAGE) (Hamilton et al., 2017), Graph ATtention network (GAT) (Veličković et al., 2018) and Graph Isomorphism Networks (GIN) (Xu et al., 2019). GCNs introduced by Kipf and Welling (2017) aggregate neighbourhood information based on convolutions. Hamilton

[1] https://github.com/VerbekeLab

et al. (2017) extends on this idea by introducing GraphSAGE resulting into an inductive method. Veličković et al. (2018) introduces attention mechanisms using GAT, allowing the distinction between important and less important neighbours in the network. Finally, Xu et al. (2019) introduced GIN, relying on the Weisfeiler-Lehman graph isomorphism test to come to a more versatile version of GNNs.

2.3 Continual Learning

In continual learning, a model needs to sequentially learn disjoint tasks $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$ (Kirkpatrick et al., 2017; Abulaish et al., 2024). Specific observations are provided with each task $\mathcal{T}_i$, while access to data of previous tasks is often limited or even prohibited. Each task has its corresponding feature set X_i, and task-specific label $y_l \in \mathcal{Y}_i$, with label set $\mathcal{Y}_i = \{y^1, \ldots, y^{c_i}\}$, where c_i represents the number of classes in task $\mathcal{T}_i$. Sometimes, a specific assumption is made that task data is provided as $(\mathcal{X}, \mathcal{Y}, \mathcal{D}_\mathcal{C})$ with $\mathcal{D}_\mathcal{C}$ the underlying distribution, also called context set (De Lange et al., 2022).

Depending on the available information and the format of the tasks provided, four different continual learning settings are discerned, i.e., task-incremental, domain-incremental, class-incremental and time-incremental learning (van de Ven et al., 2022; Ko et al., 2024). The first three are well-established in continual learning (van de Ven et al., 2022). **Task-incremental learning** consists of a sequence of distinct tasks to be learned, where the model knows which task is currently presented, even at test time. **Domain-incremental learning** describes the scenario in which the problem is the same, but the distribution of the tasks shifts. Here, no information about the task is provided at test time. In **class-incremental learning**, a growing number of classes are provided with each new task, but no task information is provided. Hence, the methods should also be able to learn to distinguish the current task that is provided. **Time-incremental learning** encompasses problems where data is provided in streaming format, and where the distribution might shift over time. Some work considers this to be a separate setting (Ko et al., 2024), while it can also be seen as a specific case of domain-IL.

To mitigate catastrophic forgetting, different methods have been developed, broadly classified into three categories, i.e., replay, regularisation-based and parameter isolation (De Lange et al., 2022). **Replay** methods preserve some historical observation - either real or synthetic - to revisit when training on new tasks. **Regularization-based** methods use heuristics to determine the important weights in the neural network and to penalize changing these weights more when learning new tasks. Finally, **parameter isolation**, also referred to as architecture-based, reserves specific weights to be updated on specific tasks. This can be done by freezing part of the network, or extending the neural network for new tasks.

Specific evaluation metrics are used to evaluate continual learning methods. The most widely used are average accuracy, average forgetting and forward transfer (Ko et al., 2024; Abulaish et al., 2024). These evaluate the performance after learning all tasks. **Average accuracy** is the average of accuracy over the tasks. **Average forgetting**, on the other hand, assesses the degradation of accuracy over the tasks by comparing the accuracy of a task after training on that tasks to the accuracy after learning on all tasks.

Negative forgetting is sometimes also called backward transfer. **Forward transfer**, also called zero-shot learning, is the increase in accuracy when using the model trained on previous tasks, compared to random initialization (Lopez-Paz and Ranzato, 2017; Abulaish et al., 2024).

2.4 Continual Graph Learning

When using network data, additional considerations come into play for continual graph learning, since observations over different tasks may be connected in the network. As a result, information from previous tasks can still be leveraged when training the current tasks due to these inter-task connections. The amount of information propagated can be significant since real-world networks often exhibit the 'small world' property, meaning that any two nodes mostly have short shortest paths between them (Zhang et al., 2022).

Suppose a classification problem where each node is part of only one task (Ko et al., 2024). The inter-task connections often fall within one of two categories, as visualised in Figure 1. The first category is where each task consists of an independent network with no links to nodes in previous tasks (Li et al., 2022b; Ko et al., 2024), so no inter-task connections are present. This can occur for two reasons. First, there are tasks which inherently do not have inter-task connections. This is often the case in graph classification (Carta et al., 2021), but also occurs when every sample in a task is a separate network. Second, separate tasks can also occur by design, where additional restrictions are put on the network by removing inter-task connections.

The second category of interaction is more prevalent, and involves a network that grows over time (Wang et al., 2020; Zhang et al., 2024b; Febrinanto et al., 2023; Yuan et al., 2023; Zhang et al., 2022; Zhou and Cao, 2021; Tian et al., 2024). With each new task, new nodes are added to the network. Some nodes in the new task have connections to nodes of previous tasks. This can help alleviate catastrophic forgetting. This also means that special care is needed if we assume that not all data is available from previous tasks. Overly optimistic results can be obtained in experimental set-ups when retaining all connections in the network.

In this work, we also evaluate a continual learning setting where the network is fixed, but where the labels change over time, as shown in Figure 2. Here, the nodes are shared across tasks. This corresponds closely to real-life AML settings, where clients are monitored continuously. A banking client might start laundering money after holding an account at the bank for a couple of years.

3 Literature Review

In the past years, deep representation learning has gained increased adoption for AML, although remaining under-explored (Deprez et al., 2024a). In the same vein, the field of continual learning is mature, but less attention has been given to continual graph learning (Zhang et al., 2022; Ko et al., 2024). To answer RQ1, this section provides an overview of the literature, with supporting summaries on the classification of the most prominent methods (Table 1), whether the paper discusses fraud detection or network analytics with related backbones (Table 2), and the continual learning methods applied (Table 3).

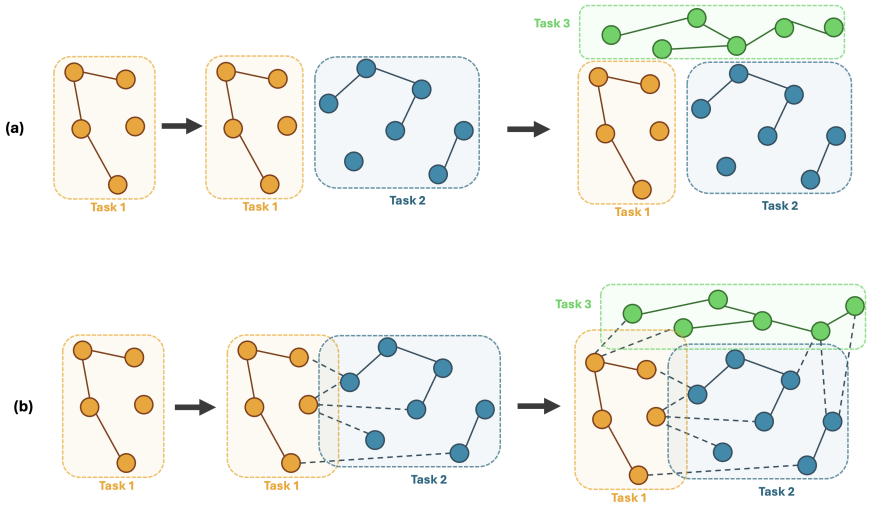

Figure 1: Two assumptions in continual graph learning for node-based learning. Either a separate network is provided for each task (a), or the network grows over time (b).

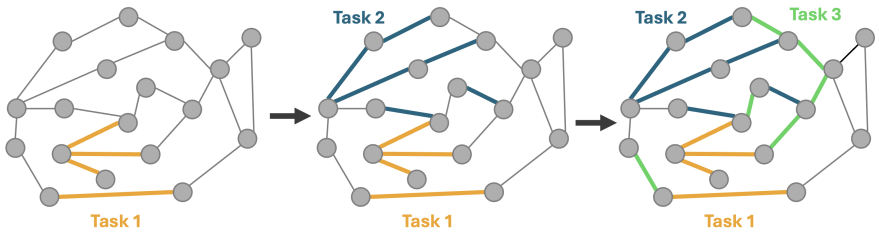

Figure 2: Visualisation of edge-based continual learning, where the network stays fixed, but the edge labels gradually become known.

3.1 Anti-Money Laundering

Anti-money laundering research has a relatively long history, and became an area of interest after the Bank Secrecy Act was passed in the USA (Welling, 1989). The importance of the research field increased after the 9/11 terror attacks with the introduction of the USA PATRIOT Act.

Since regulations include pre-defined thresholds above which transactions should be reported, many institutions still rely on rule-based and expert-based systems (Senator et al., 1995; Oztas et al., 2024; Bolton and Hand, 2002; Jensen and Iosifidis, 2023). Although these rules offer transparency in decision-making (Zhu and Hu, 2013), they become cumbersome and costly to maintain and update (Van Belle et al., 2023). Additionally, once these rules are known, they may easily be circumvented (Jensen and Iosifidis, 2023). Therefore, machine learning has been introduced to AML since it captures more complex laundering patterns and scales better to very large transaction datasets (Chen et al., 2018; Jensen and Iosifidis, 2023).

More than for other financial fraud cases—where money is drained as quickly as possible from a victims bank account—money launderers try to hide their activities, making money laundering a long-term process. To that end, individual transactions are made to appear as normal as possible. Therefore, research is focussing more on uncovering money laundering using the flow of money, i.e., the combination of transactions across multiple accounts (Zhou et al., 2017; Li et al., 2020; Tariq and Hassani, 2025).

This has resulted in a strong case for the application of network analytics in AML (Deprez et al., 2024a), with one of the first paper by Senator et al. (1995), who used the visualisation of links among people, businesses, accounts and locations, and combined it with heuristics to uncover interesting new leads. Recent work indicates that network-based methods hold a lot of potential for AML (Kurshan and Shen, 2020; Gao and Ye, 2007).

Previous literature tackles AML from various different angle, such as unsupervised learning, anomaly detection and supervised learning. Although a popular approach, unsupervised learning on networks is often limited to role definition in criminal networks based on neighbourhood and centrality metrics (Fronzetti Colladon and Remondi, 2017; Drężewski et al., 2015; Ovelgönne et al., 2012) or on the aggregation of local (node-specific) information (Zhdanova et al., 2014). However, recent studies try to uncover suspicious streams of money (Zhou et al., 2017; Li et al., 2020; Phetsouvanh et al., 2018; Tariq and Hassani, 2025).

Anomaly detection methods are less popular in AML. Due to obfuscation tactics, money laundering transactions are not easily picked up as anomalies, resulting in weak results, especially for real-life datasets (Lorenz et al., 2021; Deprez et al., 2024a).

In the case of supervised learning, a major part of the literature applies Graph Neural Networks (GNNs) since GNNs can combine topological as well as feature information to capture more complex network patterns. These are discussed in the next section.

3.2 Graph Neural Networks for Anti-Money Laundering

Alarab et al. (2020) present experiments with a GCN (Kipf and Welling, 2017) to construct meaningful network embeddings for AML. Cases are classified by combining the embedding with original transaction features. Lo et al. (2023) introduced inspection-L, which applies a GIN (Xu et al., 2019) and uses Deep graph infomax to find predictive node embeddings by comparing the embeddings of the real network to that of a corrupted network. The application of GNNs is extended by Jin et al. (2022). First, a heterogeneous interaction network is constructed to extract additional features. Then, the authors apply GCN (Kipf and Welling, 2017), GraphSAGE (Hamilton et al., 2017) and GIN (Xu et al., 2019) on the homogeneous network to arrive at network embeddings augmented with the features from the heterogeneous network, so as to obtain the final predictions.

As money laundering typically occurs over a longer period of time, research has also aimed at extending GNNs to a spatio-temporal setting. This is often done by constructing embeddings on snapshots that are fed into a deep learning method for time series analysis. The work by (Xia et al., 2022) feeds the embeddings coming from a GCN (Kipf and Welling, 2017) to an LSTM, while (Zhang et al., 2021) applies a transformer model to the embedding vectors of the different snapshots.

Furthermore, GNNs have been used for AML anomaly detection. Cardoso et al. (2022) use GAT (Veličković et al., 2018) for link prediction between nodes in the network. These predictions are compared to the real links in the network, leading to anomaly scores that indicate suspicious transactions.

3.3 Continual (Graph) Learning Methods

When training (graph) neural networks in the classic way, it is assumed that the data is identically distributed, often with the possibility to shuffle it before using it to train the model (Parisi et al., 2019). However, this is not the case in many real-life applications, where data becomes available in streams over time. An added difficulty arises when there is also a distributional shift in the tasks to learn. This can lead to catastrophic forgetting (French, 1999; McCloskey and Cohen, 1989; Goodfellow et al., 2015), where updating the model with new information leads to interference with earlier-acquired knowledge. The trade-off between the ability to retain old knowledge while also processing new information is called the stability-plasticity dilemma (De Lange et al., 2022; Zhou et al., 2024; Wang et al., 2024).

Continual learning, which was introduced to mitigate catastrophic forgetting, has mostly been applied for image classification (Lian et al., 2024). One starts with a couple of images to classify, e.g., cat versus dog. The classes are then extended where the model also needs to be able to distinguish between, e.g., cars and planes. When learning this second task, the model should retain its ability to distinguish between cats and dogs.

Given the origin of the field, many of the proposed methods are evaluated on image classification (Rebuffi et al., 2017; Lopez-Paz and Ranzato, 2017; Kirkpatrick et al., 2017; Zenke et al., 2017; Li and Hoiem,

2018; Mallya and Lazebnik, 2018; Liu et al., 2023; De Lange et al., 2022). The most frequently used datasets are MNIST, CIFAR100 and ImageNet, and these also serve as the main datasets in benchmarking studies (De Lange et al., 2022; Carta et al., 2021; van de Ven et al., 2022; Zhou et al., 2024).

Continual learning methods are classified in three categories, i.e., replay, regularisation-based and architecture-based (De Lange et al., 2022; Lian et al., 2024; Tian et al., 2024), whereas hybrid methods also exist. A summary is given in Table 1, based on the classifications given by De Lange et al. (2022) and Tian et al. (2024).

Replay methods rely on a memory buffer for retaining a subset of data from previous tasks, called exemplars. These methods assume that resources are available to store previous data. Replay emerged in reinforcement learning (Rolnick et al., 2019), where past transitions are replayed for better learning on a single task, while replay in continual learning is used to retain information across tasks. Care should be taken when adopting such methods with regards to regulation. Due to privacy concerns, not all data is allowed to be stored indefinitely. Additionally, it is possible that these methods overfit on the few exemplars that are kept in memory (Wang et al., 2020).

Pure replay methods only specify a buffer size $\mathcal{B}$, and randomly assign $\mathcal{B}/k$ observations from the current task to be replayed during new tasks. This random selection is also applied by GEM (Lopez-Paz and Ranzato, 2017) to select their exemplar set. iCarl (Rebuffi et al., 2017), on the other hand, uses smart allocation by selecting exemplars that best preserve the average feature vector of that task.

Another replay strategy, called pseudo-rehearsal, is applied by Li and Hoiem (2018) for their method Learning without Forgetting (LwF) in a task-IL setting. The authors assume that each task has its own output head. As data on previous tasks is not available, LwF takes the data of the current task, and makes predictions for the output heads of the previous tasks as well. This gives a new 'ground truth' for previous tasks, to which output is compared to during training, to keep the output on previous tasks stable.

Replay methods have also been extended to graph learning. Zhou and Cao (2021) use the *mean of features* of selecting exemplars, and extend it for graph data by selecting exemplars based on the mean embedding, based on coverage maximisation and based on influence maximisation. Wang et al. (2020) employ a two-step sampling approach where first the network is divided into clusters, after which nodes are selected within each cluster based on an importance metric.

Liu et al. (2023) proposes CaT for class-IL. CaT selects a subset of nodes randomly, and uses a structure-free graph condensation method (Zheng et al., 2023) to align the mean of latent features in the subset of nodes by training the input node features as weights.

Regularisation-based methods limit the updates to weights in the (graph) neural network. Determining which weights can change and by how much leads to a trade-off between the stability and plasticity of the model (Parisi et al., 2019).

Regularisation-based methods often capture which weights are important for previous tasks, and limit how much these can be changed. This is done by introducing additional terms in the loss function. EWC (Kirk-

Method	Replay		Generative		Regularisation				Architecture		
	Sampling				Weight Constraint		Knowledge Distillation		Fixed		Dynamic
	Random	Biased	Pseudo-Rehearsal	Graph Condensation	Topology	Importance	Knowledge Type	Distillation Paradigm	Param. Allocation	Param. Efficiency	Architecture Learning
iCarl	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗
SI	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
EWC	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
MAS	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
GEM	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
TWP	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗
LwF	✗	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗
ER-GNN	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
PackNet	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗
Piggyback	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗
HAT	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗
CaT	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗
PI-GNN	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
CGNN	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗

Table 1: Classification of the main continual learning methods.

patrick et al., 2017) uses the elements in the Fisher information matrix to express which weights were important for previous tasks. As mentioned by Zenke et al. (2017), EWC only makes point estimates of the importance using the diagonal elements of the Fisher information matrix. They put forward SI (Zenke et al., 2017), where the importance of the weights are continuously calculated throughout the training process.

MAS (Aljundi et al., 2018) on the other hand, is based on the gradient of the squared l_2-norm of the learning function output, making it applicable to unlabelled data as well. Simplifications of the importance calculations are given in case all layers use a ReLU activation function.

Regularisation can also be approached by altering the gradient before updating the weights. Using the exemplars in GEM, Lopez-Paz and Ranzato (2017) use orthogonal projection of the gradient on the span of the gradients of the previous tasks. The authors claim that this does not increase the loss on older tasks when updating the weights for the new task.

Liu et al. (2021) extends regularisation to network data with TWP. Two importance scores are included, i.e., a task-related one similar to EWC and a topology-related one based on the attention mechanism in graph attention networks (GAT) (Veličković et al., 2018). Similarly, to mitigate the problem of overfitting on the exemplars, Wang et al. (2020) applies the same idea of EWC to GNNs. The authors use the diagonal elements of the Fisher information matrix to regularize the updates of important weights in the GNN.

Architecture-based methods, as their name suggests, alter the architecture of the (graph) neural network based on the tasks to avoid interference between tasks. Specific parameters can be isolated to be fine-tuned, or the architecture itself can be extended for new classes, e.g., have separate output heads for each task (Li and Hoiem, 2018). Here, knowledge of the current task must be provided. Some methods also assume that the total number of tasks is known upfront. This limits their adoption in practice.

van de Ven et al. (2022) mention the use of entirely separate output layers or networks to learn each task. In the task-IL setting, Li and Hoiem (2018) initializes a new output layer for each new task. Other methods include the usage of gating. XdG (Masse et al., 2018) introduced a context-dependent gating signal, to have sparsely connecting, mostly non-overlapping parts of the network trained on the different tasks.

Similar to gating, PackNet (Mallya and Lazebnik, 2018) and piggyback (Mallya et al., 2018) apply task-specific masks to set some weights in the neural network equal to 0. When learning a task, Mallya and Lazebnik (2018) train the network and then prune it. The remaining weights are then fine-tuned in a second, shorter training round. These weights are fixed and the pruned weights are made available for the next task. The main drawback is that less capacity is available for training on the next task, meaning that PackNet can only learn a limited number of tasks. This is mitigated in the work by Mallya et al. (2018). Here, the network weights are fixed and the task-specific masks are learned. The authors use gradient-based learning to obtain real-valued masks, which are then converted to binary masks using a fixed threshold. The main drawback here, next to the need to know which task is considered, is that piggyback requires a pre-trained network. The performance of piggyback is highly dependent on this pre-training step (Mallya et al., 2018), and this might even not be available depending on the application.

Paper	Graph Data	Fraud	GCN	GraphSAGE	GAT	GIN
Ko et al. (2024)	✓	✗	✓	✗	✗	✗
Zhang et al. (2022)	✓	✗	✓	✗	✗	✗
Zhou et al. (2024)	✗	✗	✗	✗	✗	✗
De Lange et al. (2022)	✗	✗	✗	✗	✗	✗
Febrinanto et al. (2023)	✓	✗	✓	✓	✓	✓
Zhang et al. (2024b)	✓	✗	✗	✗	✗	✗
Wei et al. (2024b)	✓	✗	✓	✗	✗	✗
Wang et al. (2020)	✓	✓	✗	✓	✗	✗
Hemati et al. (2022)	✗	✓	✗	✗	✗	✗
Li et al. (2022b)	✓	✓	✗	✗	✓	✗
van de Ven et al. (2022)	✗	✗	✗	✗	✗	✗
Carta et al. (2021)	✓	✗	✗	✗	✗	✗
Zhou and Cao (2021)	✓	✗	✗	✗	✓	✗
Lebichot et al. (2024)	✗	✓	✗	✗	✗	✗
Perini et al. (2022)	✓	✓	✗	✓	✗	✗
Zhang et al. (2024a)	✓	✓	✗	✓	✓	✗
Cha and Cho (2025)	✗	✗	✗	✗	✗	✗
Galke et al. (2023)	✓	✗	✗	✓	✓	✗
Tian et al. (2024)	✓	✓	✗	✗	✗	✗
Liu et al. (2023)	✓	✗	✓	✗	✗	✗

Table 2: Summary of the main literature in continual (graph) learning, representing if the paper uses graph or fraud data and which backbone is used.

We see in Table 3 that the method that is most often used or compared against in the literature is EWC (Kirkpatrick et al., 2017), followed by naive replay, GEM (Lopez-Paz and Ranzato, 2017) and ER-GNN (Zhou and Cao, 2021). The relative popularity of replay methods is also apparent from Table 1 where six of the 14 methods integrate some form of replay in their approach. This might be the case because even simple replay methods often outperform other CL methods Wei et al. (2024a, 2025). Apart from replay, seven methods rely specifically on regularisation through importance measurement. This is often based on the Fisher information matrix (Kirkpatrick et al., 2017; Liu et al., 2021; Wang et al., 2020). In the next section, we summarise the application of these continual learning methods for financial fraud detection.

3.4 Continual Learning in Financial Fraud

When implementing fraud detection in practice, the methods need to continuously monitor millions of transactions, which results in three main challenges. The first is that, given that fraud is evolving constantly (Baesens et al., 2015; Van Vlasselaer et al., 2015), models should be updated to capture novel modi operandi. The second is constraints on computational resources. The large volume of transactions make retraining the model from scratch not always feasible given limited time and resources. The third challenge comes from the desire to retain knowledge on previously applied fraud tactics, because launderers could otherwise revert back to an older modus operandi to avoid detection. However, when fine-tuning the models, this information is lost if these tactics were not applied in the latest available data, resulting in catastrophic forgetting (French, 1999; McCloskey and Cohen, 1989; Goodfellow et al., 2015).

Although continual learning can be used to mitigate these challenges, research on the adoption of continual learning for fraud detection is scarce (Lian et al., 2024). Research on continual learning is mostly concerned

Paper	Replay	iCarl	EWC	MAS	GEM	TWP	LwF	ER-GNN	PackNet	Piggyback	HAT	CaT	PI-GNN	CGNN
Ko et al. (2024)	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zhang et al. (2022)	✗	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
Zhou et al. (2024)	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗
De Lange et al. (2022)	✗	✓	✓	✓	✓	✗	✓	✗	✓	✗	✓	✗	✗	✗
Febrinanto et al. (2023)	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓
Zhang et al. (2024b)	✗	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✓	✓	✗
Wei et al. (2024b)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
Wang et al. (2020)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Hemati et al. (2022)	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Li et al. (2022b)	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗
van de Ven et al. (2022)	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗
Carta et al. (2021)	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
Zhou and Cao (2021)	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Lebichot et al. (2024)	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Perini et al. (2022)	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Zhang et al. (2024a)	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
Cha and Cho (2025)	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Galke et al. (2023)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Tian et al. (2024)	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓
Liu et al. (2023)	✗	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗

Table 3: Summary of the main literature in continual (graph) learning, representing which continual learning method is used.

with image recognition (Lian et al., 2024). Few studies present fraud detection as the core problem (Hemati et al., 2022; Zhang et al., 2024a; Lebichot et al., 2024; Li et al., 2022b). It often only appears as one of many datasets to which continual learning methods are applied (Wang et al., 2020; Perini et al., 2022; Ko et al., 2024; Tian et al., 2024).

Lebichot et al. (2024) were among the first to quantify catastrophic forgetting for credit card fraud detection. They tested different replay strategies and EWC against fine-tuning the model. In their case, fine-tuning the model on new data seemed to be best at avoiding forgetting.

Hemati et al. (2022) applied three strategies, sequential fine-tuning, EWC, and experience replay, for auditing financial payment records. They trained auto-encoders for anomaly detection and demonstrated that continual learning has the ability to detect distributional shifts.

Zhang et al. (2024a) introduced and applied a new method, called POCL, to medical insurance fraud detection. It is a three-step approach where a GNN is pre-trained to arrive at meaningful embeddings that separate fraudulent and non-fraudulent cases. After pre-training, the detection model is trained first on historical data. For the third step, the authors rely on Temporal MAS to update the weights of their GNN during the online learning step.

Li et al. (2022b) extended continual graph learning to a case study on heterogeneous networks by introducing HTG-CFD. They apply replay and regularisation-based methods, where prototypes are constructed using the average attribute, and regularisation is done using Fisher information, inspired by EWC. HTG-CFD is constructed to transfer the fraud detection model across different regions to detect fraudulent transactions in a trade network.

ContinualGNN (Wang et al., 2020) is developed for streaming graphs to uncover new patterns over time. The authors have tested their method on the Elliptic dataset. Although ContinualGNN did not perform best, the method has competitive performance. The main strength of this method is the strong reduction in training time compared to fully retraining the GNN with new data.

3.5 Benchmarks and Evaluation in Continual (Graph) Learning

Next to performance of continual graph learning for detecting financial fraud and anti-money laundering, we are also interested in the stability of these methods and the influence of choices in the experimental set-ups. As can be seen in Table 3, most studies mentioned above only compare a limited number of methods. Therefore, in this section, we analyse general continual learning benchmarks to evaluate the effectiveness of these methods.

De Lange et al. (2022) are among the first to do an extensive benchmark study. While focusing only on task-incremental learning in classic continual learning, they implement a comprehensive benchmark in terms of methods tested, but these were tested on only three datasets, and all involve image classification. They conclude that architecture-based methods, particularly PackNet (Mallya and Lazebnik, 2018), perform best, closely followed by memory replay. However, compared to memory replay, architecture-based methods do not

suffer from privacy issues, since they do not require storing raw data.

As part of the introduction of their new two-step hyperparameter tuning framework, Cha and Cho (2025) also tested a number of continual learning methods on three image classification datasets. They report mixed results across the datasets. The best performing model is architecture-based (DER (Yan et al., 2021), which is specifically introduced for class-IL), although the replay models perform better than the other architecture-based models. However, their main conclusion is that many methods are very sensitive to hyperparameter settings and tend to overfit the data they are tuned on.

One of the first benchmark studies for continual graph learning was presented by Carta et al. (2021). This study only considers graph classification, so no tests are done at node level. It is presented as an introductory benchmark experiment and it is quite limited in its scope. It involves three datasets and three continual learning strategies. These are naive replay, EWC (Kirkpatrick et al., 2017) and LwF (Li and Hoiem, 2018). Although the paper is meant to be a benchmark on networks, no strategies that were specifically developed for networks were tested.

Two other notable benchmark studies for continual graph learning are *Continual Graph Learning Benchmark (CGLB)* by Zhang et al. (2022) and ***Be**nchmarking **G**raph Conti**n**ual Learning (BeGin)* by Ko et al. (2024). Both provide the full code suite to facilitate reproduction. The initial methods compared by Zhang et al. (2022) are EWC (Kirkpatrick et al., 2017), MAS (Aljundi et al., 2018), GEM (Lopez-Paz and Ranzato, 2017), TWP (Liu et al., 2021), LwF (Li and Hoiem, 2018) and ER-GNN (Zhou and Cao, 2021). CGLB splits continual graph learning in task-IL and class-IL, and provide experiments for node-level and graph-level predictions. Zhang et al. (2022) showed that the regularisation-based methods (EWC, MAS, and TWP) have strong performance in the task-IL setting. In the class-IL setting, replay (ER-GNN) appears to perform best. The class-IL setting has more tasks and uses multi-class classification, which is more complex than in task-IL.

BeGin implements a more extensive benchmark (Ko et al., 2024). They make a fine-grained distinction between incremental settings, by considering task-IL, class-IL, domain-IL and time-IL. These settings are also introduced for link-level predictions, on top of the earlier introduced node-level and graph-level predictions. The continual learning methods are also extended. On top of the methods compared under CGLB, BeGin includes PackNet (Mallya and Lazebnik, 2018), Piggyback (Mallya et al., 2018), HAT (Serra et al., 2018), CaT (Liu et al., 2023), PI-GNN (Zhang et al., 2023) and CGNN (Wang et al., 2020). Similar to CGLB, regularisation-based methods perform well for task-IL, while GEM (replay) is one of the single methods performing well on class-IL and domain-IL. This is a confirmation that replay gives better results for more complex settings.

Both benchmarks analyse the performance of a range of methods, but pay less attention to the impact of the hyperparameters. One of the main shortcomings of both CGLB and BeGin is that all experiments are done only with GCN (Kipf and Welling, 2017) as backbone GNN.

3.6 Sensitivity to Hyperparameters

Underlying every model is a suite of hyperparameter choices that impact model performance. These are often only briefly mentioned under hyperparameter tuning, or in the best case, papers apply limited parameter sensitivity tests. An overarching study was done by Cha and Cho (2025). They introduced a two-step hyperparameter tuning protocol. The authors indicate that tuning and evaluation on the same dataset is restricted in real-world applications. However, their study still only shows performance on the final set of hyperparameters, excluding the performance of other combinations, and they only report average accuracy, without considering the extend of forgetting. In this work, we make the effect of hyperparameters explicit.

A key choice is the backbone GNN model. Overall, Table 2 shows no preference of one backbone over another. However, when dealing with fraud data, it seems that authors often opt for GraphSAGE or GAT. Most continual graph learning methods are constructed to be backbone agnostic. One notable exception is TWP (Liu et al., 2021), which is specifically developed with GAT in mind. However, the authors provide proxies in case no attention mechanism is present.

The architecture of the backbone GNN itself - both in terms of depth and width of the hidden layers - is also important since it influences the learning capacity and the length of transaction chains that can be captured. However, the specific impact of these choices have not been given much attention in literature. Studies on general continual learning by De Lange et al. (2022) and Mirzadeh et al. (2022) demonstrated that wide and shallow models generally perform better. However, when moving to continual graph learning, Wei et al. (2024a) demonstrated that for skeleton-based action recognition this does not always hold.

Another import choice, next to the backbone, is the task definition. When considering human learning, it is hypothesized that the order of tasks is important for continual learning. *Curriculum Learning*, coined by Bengio et al. (2009), determines that knowledge can be optimally acquired if tasks are learned in ascending order of difficulty.

Previous work has performed task-order sensitivity analysis in continual learning. De Lange et al. (2022) corroborated earlier work of Nguyen et al. (2019) by showing that, in a general continual learning setting, methods exhibit order-agnostic behaviour. The authors test different setups, including an *easy to hard*, a *hard to easy*, and a random ordering of tasks.

The work by Mallya and Lazebnik (2018), on the other hand, showed that for their method PackNet, the order does matter. They found that learning tasks from hardest to easiest actually gave better results. We cannot generalise this finding, however, since this is method-specific. One explanation for this deviation is that the capacity of PackNet to incorporate novel information drops as the available free parameters decrease with each task.

In the field of continual graph learning, Zhao et al. (2024) introduced a randomly generated class appearance order to simulate the random class emergence in real world for multi-label continual graph learning. Wei et al. (2024a) focused on evaluating the task-order and class-order sensitivity in the context of continual graph learning for skeleton-based action recognition. The authors show that task-order robustness

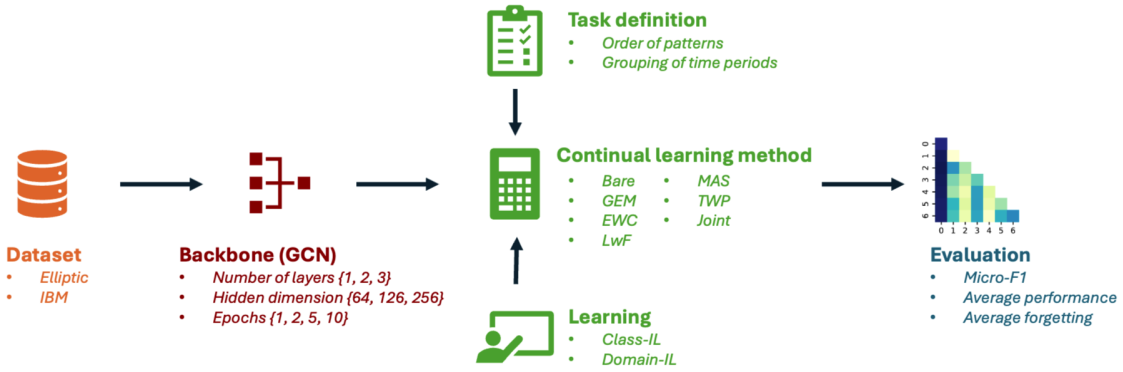

Figure 3: Visualisation of the experiment's pipeline and the hyperparameter choices for evaluation.

does not necessarily imply class-order robustness.

4 Methodology

To answer the research questions, we set up a pipeline in which we vary the different hyperparameters (RQ2), including the architecture of the GNN and the continual learning methods (RQ3) to study their effect on performance and forgetting. Figure 3 illustrates the full pipeline of our experiments, including the value of the hyperparameters. This section provides more detail on these hyperparameters. The experiments presented in this paper are built upon the BeGin framework that was introduced by Ko et al. (2024). The extended repository is made available on github[2].

4.1 Datasets

Two datasets are used in the experiments that are widely used in AML (Deprez et al., 2024a), i.e., the IBM AML dataset (Altman et al., 2023) and the Elliptic dataset (Elliptic; Weber et al., 2019). As will be discussed below, this work presents AML as edge classification (IBM) as well as a node classification (elliptic).

The IBM AML dataset (Altman et al., 2023) contains synthetic transaction data. The simulations are done in a multi-agent virtual world. These agents can be banks, individuals or companies, with payments by individuals and companies. In this virtual world, some agents are said to be malicious. For those agents, the simulations include money laundering transactions. Altman et al. (2023) model eight different money laundering patterns, i.e., fan-in, fan-out, bipartite, stack, random, cycle, scatter-gather and gather-scatter, as illustrated in Figure 4.

We select the HI-Small dataset for our evaluations, taking the agents as nodes and the transactions as edges. This results in a network with 515 080 nodes and 5 078 345 edges. For this dataset, we perform edge classification. As with most fraud datasets, the class distribution is highly imbalanced, with only 0.1%

[2] https://github.com/VerbekeLab

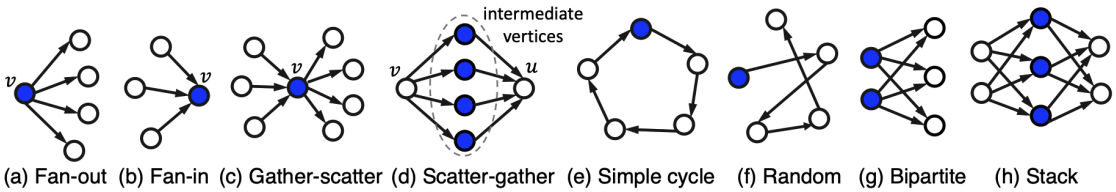

Figure 4: The different money laundering patterns as defined by Altman et al. (2023).

of transactions involving money laundering. Most of the money laundering transactions are, however, not classified under a specific pattern. Table 4 gives a detailed view on the distribution of the class labels.

As will be discussed later, we build a model that can distinguish the different patterns using multi-class classification. The 'not classified' labels likely contain observations that fall under one of the patterns, but labelling these according to one of the eight classes would induce additional noise. Therefore, we will discard the *not classified* labels in our experiments, since we have no control over the specific patterns they constitute. The nodes and edges of these instances are kept as part of the network, so relevant information from these nodes is propagated through the network and used for the classification of the other nodes. This is also a practice observed in other research when doing classification of these patterns (Altman et al., 2023; Egressy et al., 2024).

Another option is to explicitly acknowledge that money laundering cases are present among the negative labels. This could involve semi-supervised or positive and unlabelled (PU)-learning. The latter explicitly recognises that not all money laundering transactions are labelled (Ortega Vázquez et al., 2023). However, an analysis of these methods is beyond the scope of this work.

The Elliptic dataset (Elliptic; Weber et al., 2019) contains real-world Bitcoin transactions, grouped in 49 different time intervals. The network consists of 20 3769 nodes and 234 355 edges, where nodes represent transactions and edges indicate that the receiver of the first transaction was the sender of the second. For this dataset, we perform node classification. The dataset includes 166 pre-calculated numerical features — 94 transaction-specific features and 72 aggregated features summarizing a node's neighbours. The data contains only 4 545 illicit transactions (2%), again making the label distribution highly imbalanced. Although these labels do not specifically concern money laundering, we use this dataset as it has found wide adoption in the AML literature (Weber et al., 2019; Deprez et al., 2024a; Alarab et al., 2020; Xia et al., 2022; Mohan et al., 2023; Sun et al., 2022; Li et al., 2022a,c), and therefore facilitates comparison with prior experimental results.

Additionally, the Elliptic dataset provides a well-suited case-study for continual learning (Perini et al., 2022; Wang et al., 2020). As mentioned by Weber et al. (2019), there was a sudden closure of a dark market at time step 43. This caused all methods to perform poorly, due to the sudden shift in feature distribution of the illicit cases. This abrupt change in a real-world dataset is ideal for getting a deeper understanding on continual learning methods.

As noted above, both datasets have extreme class imbalance. This work will test performance of continual

Type of pattern	Transactions
Fan-out	342
Fan-in	318
Gather-scatter	716
Scatter-Gather	626
Cycle	287
Random	191
Bipartite	263
Stack	466
Not classified	1 968
Total money laundering	5 177
Total number of transactions	5 078 345

Table 4: The distribution of the different types of money laundering transaction patterns for the HI-Small datasets.

graph learning under this imbalance. Methods can be applied before the continual learning step to alleviate the imbalance, such as sampling methods to make the class distribution more balanced. An important direction for future research is to evaluate the effect of such sampling methods on the performance of CL.

4.2 Backbone Graph Neural Network

In line with previous benchmarks (Zhang et al., 2022; Ko et al., 2024), we use GCN (Kipf and Welling, 2017) as backbone. The GCN layer-wise propagation is defined for the whole network at once as:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right), \tag{2}$$

with σ an activation function, and $W^{(l)}$ the layer-specific trainable weights. We limit this study to GCN, as this is currently the only backbone implemented in BeGin (Ko et al., 2024). Other popular backbones could be GraphSAGE and GAT, but a far-reaching extension of BeGin is outside the scope of this work. Additionally, GAT has many more parameters to learn, which would substantially increase calculation time.

The main choices for the backbone are the depth and width of the layers (De Lange et al., 2022; Mirzadeh et al., 2022; Wei et al., 2024a). We vary the number of layers between 1 and 3. Here, a trade-off needs to be made. On the one hand, money laundering patterns often contain nodes that are a couple of hops in the network apart, as illustrated in Figure 4, requiring more layers to capture this information. On the other hand, having too many layers leads to over-smoothing (Li et al., 2018), lowering the predictive power of the model. The dimensions are the same for all hidden layers in the GCN. We test three values, namely 64, 128 and 256.

Another choice concerns the number of epochs per task, which we set to 1, 2, 5 and 10. The more time a model is given to train on a single task, the more likely the weights are changed and previous knowledge is lost. On the other hand, the fewer epochs used, the harder it is for the model to learn and improve performance on the current task.

Some hyperparameters of the GCN are fixed. For all experiments, we use the Adam optimizer with learning rate 0.001, no weight decay and the cross-entropy loss. The activation function for the GCN is ReLU, and the dropout rate is set to 0.5.

4.3 Task Definition

We are faced with different classification tasks for the IBM and elliptic dataset, i.e., multi-class edge classification and binary node classification. Therefore, we define the tasks for these datasets differently.

The patterns in the IBM dataset each have their own separate label, allowing for multi-class classification. The tasks are defined using the different patterns present in the dataset. The first task consists of two labels, i.e., legitimate transactions and a first money laundering pattern. Subsequent tasks are defined by adding one novel pattern at a time.

The sequence in which money laundering patterns are introduced may influence the performance of continual learning methods. To investigate this potential effect, we design five distinct task orderings, each corresponding to a different pattern order. These orderings are not only designed for analytical purposes but also reflect plausible scenarios encountered in real-world AML applications. Since these different task orders result in altered data to learn on, it also allows us to have a high-level view on the stability of the methods.

One considered ordering presents the money laundering patterns in ascending order of difficulty, which is an idea initially introduced in curriculum learning (Bengio et al., 2009). This setup mirrors the initial stages of AML implementation within financial institutions, where investigators typically begin by addressing simpler patterns and progressively develop the expertise needed to identify more sophisticated laundering schemes. This progression also reflects the dynamic nature of AML enforcement, shaped by the ongoing adversarial interaction between financial institutions and criminals. As detection capabilities improve, launderers tend to adopt increasingly complex methods, necessitating a continual adaptation by AML systems. To assess the role of pattern complexity in continual learning performance, we also consider the reverse—presenting the patterns in descending order of difficulty—as a second ordering.

The determination of what patterns are more difficult than others is inspired by the work of Egressy et al. (2024). The authors extend message-passing GNNs step-by-step to prove that the extended GNNs can capture more patterns. We use those insights to order the patterns from least to most complex as follows: fan-in, fan-out, bipartite, gather-scatter, scatter-gather, stack, cycle and random.

A third ordering is based on pattern frequency in the dataset, arranging them in descending order: gather-scatter, scatter-gather, stack, fan-out, fan-in, cycle, bipartite, and random (see Table 4). This ordering is motivated by the hypothesis that more frequently occurring patterns are detected earlier by both human

analysts and machine learning models. However, this setup presents a challenge for subsequent tasks, as less frequent patterns provide fewer training examples, potentially hindering learning. Conversely, the model may benefit from knowledge transfer across tasks, using representations learned from more frequent patterns to improve performance on rarer ones. To complement this analysis, we also consider the reverse ordering—from least to most frequent—to evaluate the impact of data sparsity in the initial tasks.

Finally, as a baseline, we also include a task ordering in which the patterns are presented in a random sequence. For our experiments, this random order is: fan-out, fan-in, gather-scatter, scatter-gather, cycle, random, bipartite, and stack.

Note that for the IBM dataset, the transaction network is static. We incrementally learn the novel patterns, while keeping all nodes and edges the same.

The elliptic dataset contains fraud/non-fraud labels, and we therefore apply binary classification. We define the tasks in two ways. First, we can take each time step as an individual task, resulting in 49 tasks. Having too many tasks, however, might pose a problem for the continual learning methods, since information from a very long time ago should also be retained near. Therefore, we introduce a second way of task definition, where we group different time steps in a single task. Here, each task contains seven time steps. This is chosen to have the same number of time steps in each task.

4.4 Continual Learning Methods

Different methods to prevent forgetting are present in continual graph learning literature, starting with the type of incremental learning to use. Similar to the task definition, we apply different incremental learning strategies for the two datasets.

For the IBM dataset, we use class incremental learning, since each new pattern is seen as its own class. One challenge, as indicated in the literature, is that these different tasks in the class-incremental setting are very similar, possibly resulting in strong forgetting across tasks (Wang et al., 2024; Lee et al., 2021).

For the Elliptic dataset, we use domain-IL, where we recognise that the distribution in money laundering patterns can shift. This mimics what happens in reality, since financial institutions will also use binary classification (i.e., legit vs. money laundering), but need to consider that the modus operandi of fraudsters evolves over time.

The continual learning methods used are Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017), Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), Learning without forgetting (LWF) (Li and Hoiem, 2018), Memory Aware synapses (MAS) (Aljundi et al., 2018), and Topology-aware Weight Preserving (TWP) (Liu et al., 2021). This selection is made since the literature as presented in Section 3.4 mostly relies on these methods for fraud detection. These methods are compared to the bare and joint model.

- **Bare**: We iteratively fine-tune the model on only the data of the current task. In continual learning, this is taken as a lower bound for the performance.

- **Gradient Episodic Memory (GEM)** (Lopez-Paz and Ranzato, 2017): GEM uses a fixed budget for memory allocation, and this memory is filled without any smart allocation. The gradient of the current task is projected orthogonally onto the space spanned by the gradients calculated using the replay examples, to avoid that the losses on previous tasks will increase.

- **Elastic Weight Consolidation (EWC)** (Kirkpatrick et al., 2017): EWC uses the Fisher information matrix, based on the gradient of the loss, to find the weights that were important for the previous task. It changes the loss function by introducing heavier penalization of updating more important weights.

- **Learning without forgetting (LwF)** (Li and Hoiem, 2018): LwF is introduced using a multi-task architecture, where part of the model is shared, and part is fine-tuned for each specific task. It assumes that no data of older tasks is available. LwF starts by constructing new 'ground truth' labels by looking at the output on the parts of the old task using the data of the current task. Original capabilities are preserved by trying to keep these outputs as is while training on the new task.

- **Memory Aware synapses (MAS)** (Aljundi et al., 2018): MAS uses the gradient of the squared $l2$-norm of the learned function output to express weight importance. Contrary to EWC, the weight importance determined by MAS is done in an unsupervised manner.

- **Topology-aware Weight Preserving (TWP)** (Liu et al., 2021): TWP uses two sub-modules, one for task-related objectives and one for topology-related objectives. The task-related objective is similar to EWC where weight importance is measured via the gradient of the loss. The topology-related objective relies on the gradient vector of the attention coefficients in a GAT to incorporate network topology. The authors include a non-parametric proxy for the attention, in case the GNN backbone does not include an attention mechanism.

- **Joint**: the joint model takes an accumulative approach. Similar to the bare model, the model of the previous task is fine-tuned on the current task. Contrary to the bare model, the joint model is fine-tuned using all data of the current and past tasks. In continual learning, this is taken as an upper bound for the performance.

4.5 Evaluation

Previous work mostly uses accuracy to evaluate performance (Abulaish et al., 2024). However, AML deals with strong class label imbalance, motivating the evaluation of performance by using the micro-F1 score for both datasets.

In continual learning, special evaluation metrics are developed to reflect that the model is fine-tuned sequentially on the tasks. We use the model tuned for task j, and evaluate it on the test data of previous tasks $i \leq j$. This allows us to quantify the forgetting that has occurred after fine-tuning the model on task j.

Using this principle, we define the performance matrix, $M \in \mathcal{R}^{k \times k}$, with k the number of tasks. The elements of the performance matrix are defined as (Zhang et al., 2022):

$$M_{i,j} = \begin{cases} \text{Performance on task } i \text{ after training on task } j & \text{if } i \leq j \\ 0 & \text{otherwise} \end{cases}$$

Hence, the performance matrix is a lower triangular matrix. The visualisation of the performance matrix using a heatmap is a first, qualitative evaluation of the methods.

The performance matrix entries are used to calculate quantitative evaluation metrics, i.e., the average performance (AP) and average forgetting (AF). Average performance is the average micro-F1 over the tasks, after training on all tasks. Average forgetting compares the micro-F1 of a task after training on said tasks to the accuracy after learning on all tasks. Using the performance matrix, we defined these metrics as (Ko et al., 2024):

$$\text{AP} = \sum_{i=1}^{k} \frac{M_{k,i}}{k} \tag{3}$$

$$\text{AF} = \sum_{i=1}^{k-1} \frac{M_{i,i} - M_{k,i}}{k-1} \tag{4}$$

In the end, we are also interested in the performance of the final model on all tasks. Therefore, we include the micro-F1 score on all data after fine-tuning the model on the final task.

5 Results and Discussion

As shown in the pipeline on Figure 3, many hyperparameters choices are tested in this work. We start below with a general overview of all results using Figure 5, Figure 6 and Figure 7, in which some trends are already clear. Afterwards, a detailed discussion for each choice is given in dedicated sections.

Figure 5 contains a scatter plot and marginal kernel density estimates of the results over all different (hyper-)parameters for the IBM dataset. We see strong forgetting across experiments, which is probably caused by the strong similarity among the different tasks (Wang et al., 2024). We notice that in general more epochs lead to more forgetting, while there is a limit on the performance the model can obtain. Additionally, GEM (replay- and regularisation-based) seems to achieve good performance without suffering too much forgetting. Across the datasets, it seems that the task order has no noticeable impact on performance and forgetting.

A more fine-grained visualisation for the IBM dataset is provided in Figure 6. The results are split in different plots according to the depth and width of the model. We see that higher dimensions lead to an increase in performance, but also correlate with stronger forgetting. These plots clearly show the ability of GEM (replay- and regularisation-base) to obtain higher AP while having lower AF. We also note that the Bare model suffers strong forgetting, especially for the GCNs with three layers. For the other layers, Bare,

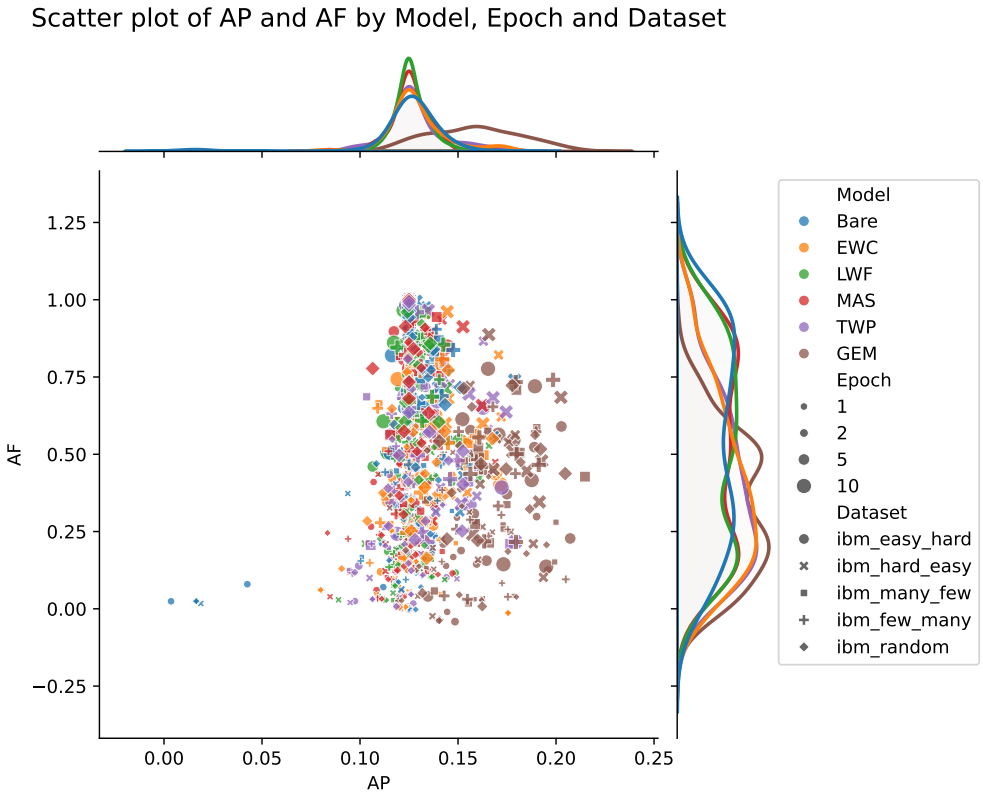

Figure 5: Scatter plot and marginal kernel density estimates of the average forgetting plotted against the average performance for the IBM dataset.

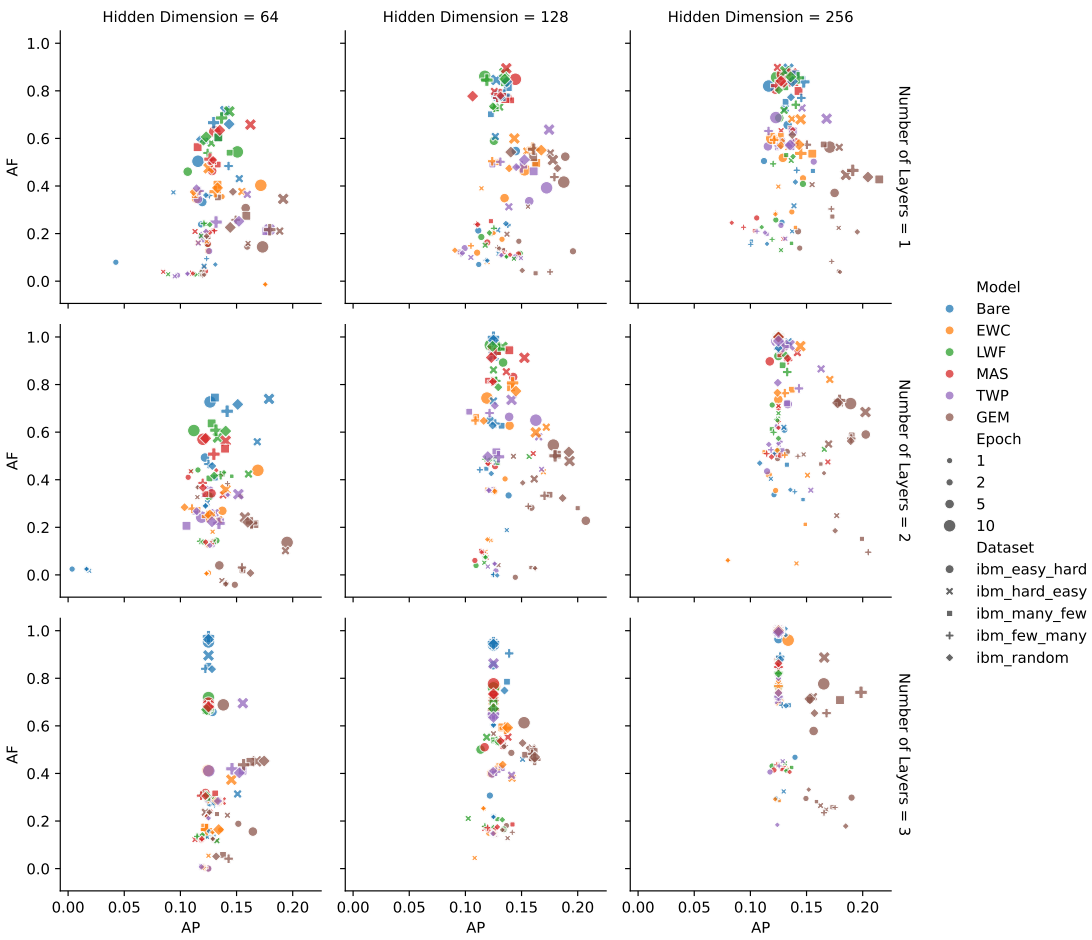

Figure 6: Scatter plot of the average forgetting (lower is better) plotted against the average performance (higher is better), split according to the breadth and width of the GCN.

MAS and LwF (regularisation-based) seem to consistently suffer more from forgetting for similar performance, compared to the other methods.

A similar figure is given for the Elliptic dataset in Figure 7, where different results are observed compared to the IBM dataset. For the Elliptic dataset, it seems that forgetting is less of a problem, but the average performance clearly increases with the number of epochs. In general, the results are similar between the setting with seven and with 49 tasks. The lack of forgetting can be because the distribution shift is less severe in this dataset. Looking at the methods, GEM again has on average lowest forgetting, while LwF and Bare have higher forgetting. We do observe that EWC and TWP need more epochs to obtain a similar performance as the rest, although both keep a fairly constant level of forgetting around 0. The difference in forgetting among methods for the Elliptic dataset is also much less pronounced than for the IBM dataset.

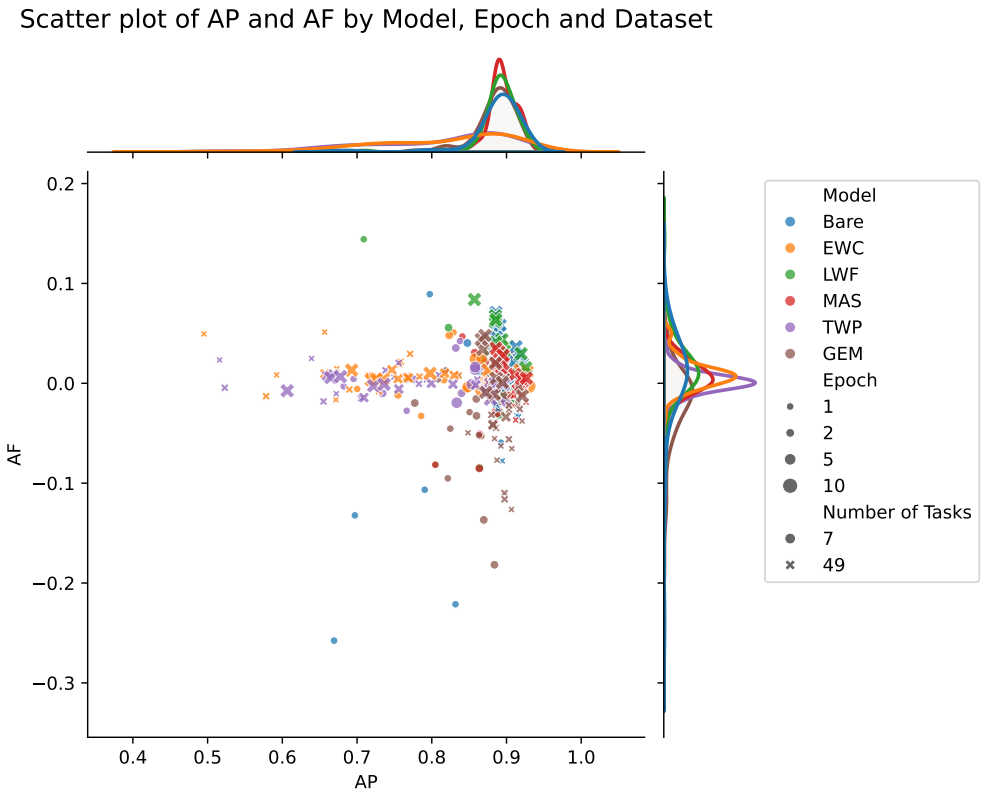

Figure 7: Scatter plot and marginal kernel density estimates of the average forgetting plotted against the average performance for the Elliptic dataset.

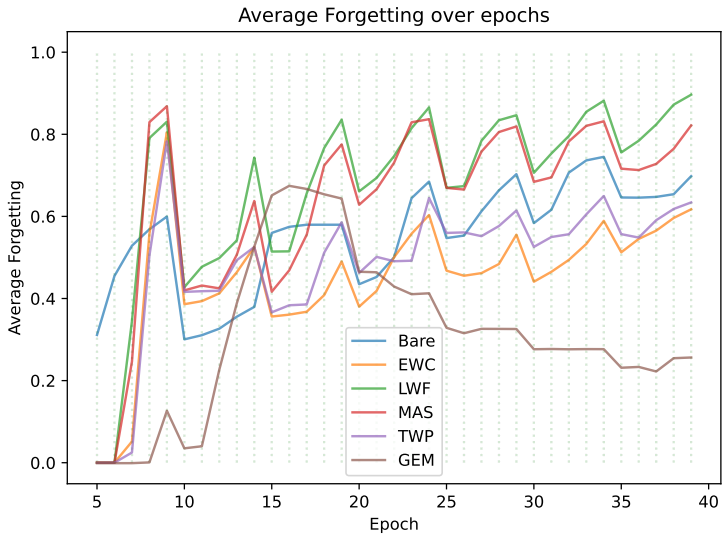

Figure 8: Average forgetting for the different methods on the IBM dataset, where the backbone is a GCN with two layers, each having dimension 128, and we take five epochs per task.

5.1 Number of Epochs

We start by illustrating the importance of carefully choosing the number of epochs. The evolution of the average forgetting as a function of the number of epochs is shown in Figure 8 for the IBM dataset. We can clearly see that the average forgetting stays relatively low for a couple of epochs, but suddenly jumps up. This illustrates that, in this case, knowledge from previous tasks is not lost gradually, but suddenly.

We note that the average forgetting drops when going to a new task, after every five epochs. This is because a new task is added to the average forgetting calculations. As the model has just fine-tuned on data from that task, the forgetting is expected to still be low. This results in a lower average, and hence a drop in forgetting when considering the next task.

The average forgetting, average performance and final performance are reported for the IBM dataset in Table 5 over the different number of epochs per task. We see that performance improves when increasing the number of epochs. However, the amount of forgetting becomes a major issue with a higher number of epochs. Only GEM seems to be able to keep forgetting at a lower level.

We see that the final performance goes down considerably when the number of epochs per task goes up. This is caused by the high forgetting on previous tasks. Especially forgetting for the first task is problematic, given that it contains the majority class.

The average forgetting, average performance and final performance for the Elliptic dataset are reported in Table 6 for seven tasks and in Table 7 for 49 tasks over the different number of epochs per task. As before, the performance increases with the number of epochs per task, but the increase of forgetting is not as drastically as for the IBM dataset. Here, both TWP—which incorporates network topology—and GEM seem

Epochs	Bare			EWC			LWF			MAS			TWP			GEM		
	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.
1	0.1250	0.0000	0.0004	0.1353	0.4036	0.0003	0.1095	0.0394	**0.7800**	0.1084	0.0606	0.7431	0.1272	0.0171	0.6529	**0.1444**	**-0.0100**	0.6372
2	0.1385	0.3340	0.0001	0.1259	0.3501	0.0076	0.1253	0.4897	0.0001	0.1265	0.4569	0.0001	0.1250	0.3574	0.0002	**0.1588**	**0.2823**	**0.0339**
5	0.1220	0.6489	0.0001	0.1393	0.6274	0.0002	0.1336	0.8924	0.0001	0.1425	0.8318	0.0001	0.1389	0.6642	0.0002	**0.2072**	**0.2277**	**0.2783**
10	0.1250	0.9804	0.0001	0.1192	0.7433	0.0001	0.1222	0.9657	0.0001	0.1250	0.9576	0.0001	0.1627	0.6503	0.0003	**0.1783**	**0.5456**	**0.2337**

Table 5: Performance metrics (AP and AF) and final performance on all data (Fin.) for different models and epochs for the IBM dataset. The backbone is a GCN with two layers, each having dimension 128.

Epochs	Bare			EWC			LWF			MAS			TWP			GEM		
	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.
1	0.8318	**-0.2213**	0.8284	0.8903	-0.0010	0.9019	0.8908	0.0012	0.9029	**0.8922**	0.0000	**0.9041**	0.8900	-0.0018	0.9020	0.8860	-0.0196	0.8930
2	0.8971	-0.0012	0.9074	0.8936	-0.0035	0.9028	0.8946	0.0012	0.9057	**0.8973**	-0.0026	**0.9077**	0.8936	-0.0011	0.9050	0.8696	**-0.1368**	0.8771
5	0.9004	0.0156	0.9102	0.9037	0.0045	0.9113	0.9040	0.0158	0.9133	0.8978	0.0166	0.9085	0.9063	-0.0093	0.9150	**0.9097**	**-0.0107**	**0.9168**
10	0.9084	0.0287	0.9149	0.9156	0.0116	0.9204	0.9091	0.0248	0.9157	0.8996	0.0275	0.9099	0.9162	**-0.0026**	**0.9229**	**0.9175**	-0.0023	0.9227

Table 6: Performance metrics (AP and AF) and final performance on all data (Fin.) for different models and epochs for the Elliptic dataset, with seven tasks. The backbone is a GCN with two layers, each having dimension 128.

to be able to achieve high performance in combination with negative forgetting, meaning that the model also improves its performance on previous tasks when fine-tuning on novel tasks.

The final performance of the models seems less affected by the number of epochs. This is probably caused by the very little forgetting due to limited data shift over the tasks. Therefore, the model has had ample training time at the end of fine-tuning on all tasks, even when the number of epochs per task is small.

These results also stress the importance of the network structure in CGL. Figure 8 and Table 5 illustrate that on the IBM dataset the Bare model, although considered as a lower bound, does not suffer the strongest forgetting of all models. There can be a couple of reasons for this. A first reason is that the network structure, via the inter-task connections, is beneficial for the bare model to retain knowledge from previous tasks. This reason seems to be supported by the results of the Elliptic dataset in Table 6, where the Bare model seems to be performing worse, although not in all cases. Inter-task connections are absent in the Elliptic dataset. A second reason for the deviant performance of the Bare model on the IBM dataset is visible in the performance matrices in Figure 9. It seems that for some tasks, the bare model has difficulty obtaining good performance,

Epochs	Bare			EWC			LWF			MAS			TWP			GEM		
	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.	AP	AF	Fin.
1	**0.9146**	**-0.0173**	**0.9229**	0.7328	0.0060	0.7200	0.8978	0.0105	0.9100	0.9129	-0.0081	0.9213	0.7927	0.0033	0.7821	0.8756	-0.0155	0.8778
2	0.9144	0.0023	0.9236	0.7584	0.0215	0.7431	0.9157	-0.0014	0.9250	**0.9203**	-0.0107	**0.9269**	0.7557	0.0204	0.7472	0.8798	**-0.0316**	0.8824
5	0.8870	0.0507	0.9031	0.8764	0.0024	0.8707	0.8913	0.0372	0.9058	0.9133	**-0.0058**	**0.9225**	0.8766	0.0005	0.8860	**0.9168**	-0.0035	0.9211
10	0.8913	0.0583	**0.9050**	0.7977	0.0100	0.7794	0.8849	0.0617	0.9020	0.8897	0.0316	0.9044	0.7362	**-0.0010**	0.7182	**0.8947**	0.0009	0.8961

Table 7: Performance metrics (AP and AF) and final performance on all data (Fin.) for different models and epochs for the Elliptic dataset, with 49 tasks. The backbone is a GCN with two layers, each having dimension 128.

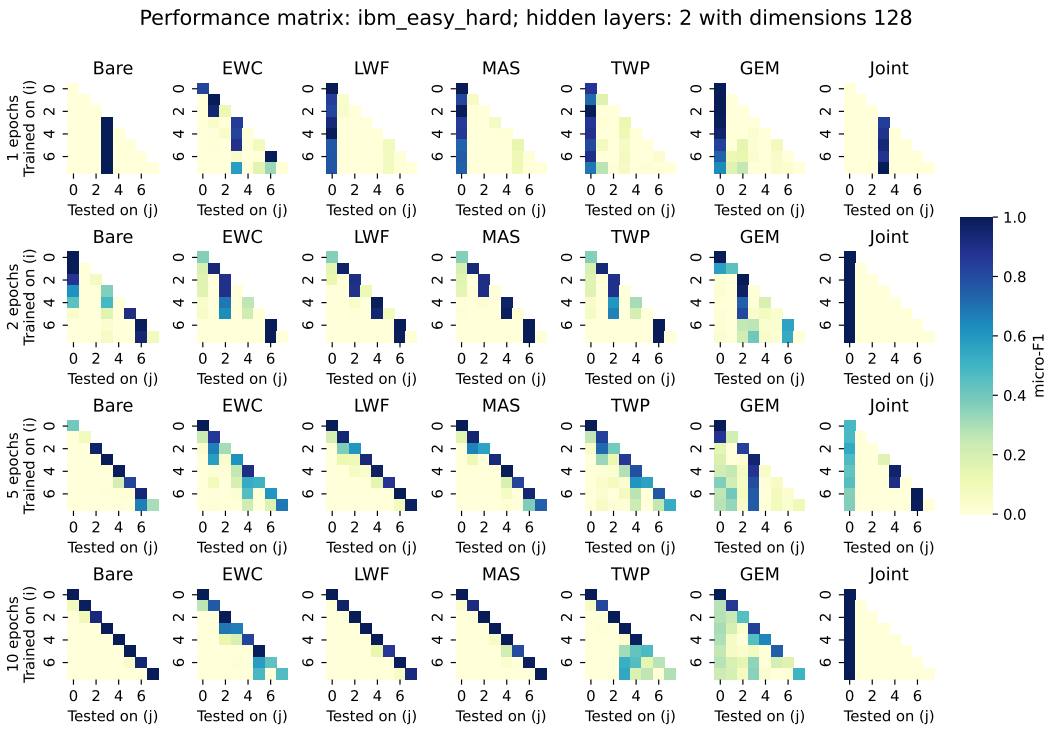

Figure 9: The performance matrices for the different methods for a varying number of epochs for the IBM dataset.

leading to less *learned information* to forget.

Further analysis of the performance matrices in Figures 9-11 illustrate that a balance needs to be struck between performance and forgetting. The forgetting is kept low with a lower number of epochs, although the model does not seem to learn any knowledge from the new tasks. On the other hand, the models tend to overfit on the latest task for the IBM dataset if epochs are set too high.

The complete collection of performance matrices is available as supplementary material online on Github[3].

5.2 Order of Patterns

The analysis of the order of patterns is only relevant for the IBM dataset. For the same architecture as before, we see in Figure 12 that the forgetting and precision is more or less stable across different pattern orders. Only for *hard to easy* we notice that the forgetting spikes for a couple of methods. Here, the average performance is also higher.

When aggregating on all experiments, as shown in Figure 13, we see that on average the order of the patterns does not seem to have a major impact on the performance nor on the forgetting. This is in line with previous studies (De Lange et al., 2022; Nguyen et al., 2019).

We notice that the boxes for EWC and TWP - which is based on EWC - go a bit higher in terms of

[3] https://github.com/VerbekeLab

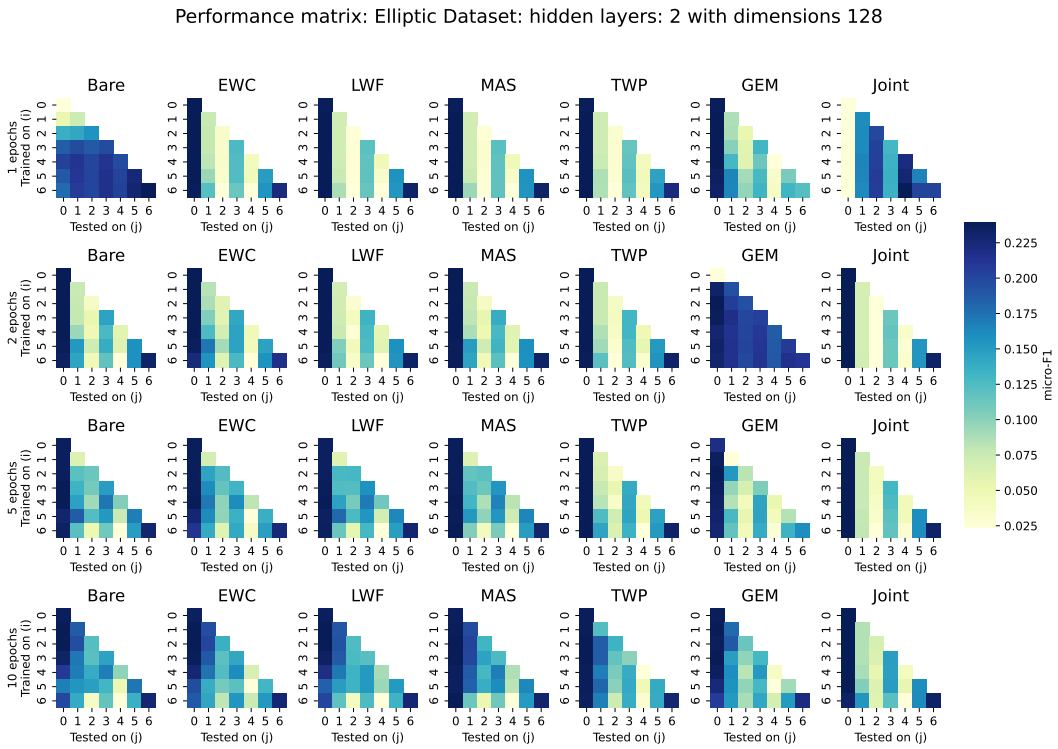

Figure 10: The performance matrices for the different methods for a varying number of epochs for the Elliptic dataset with 7 time steps per task.

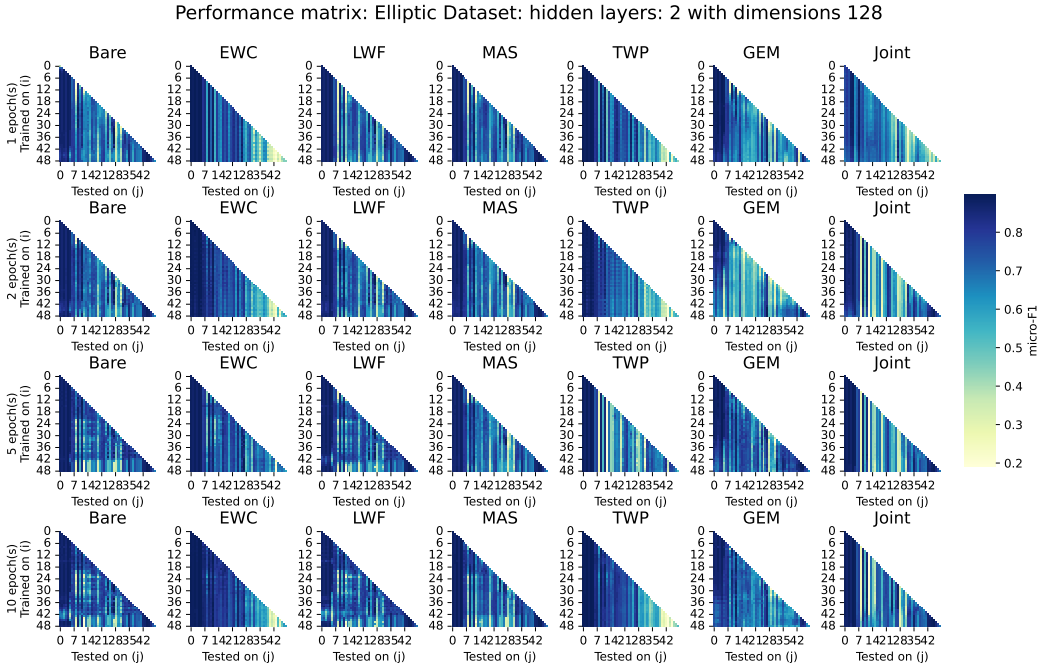

Figure 11: The performance matrices for the different methods for a varying number of epochs for the Elliptic dataset with each time step a separate task.

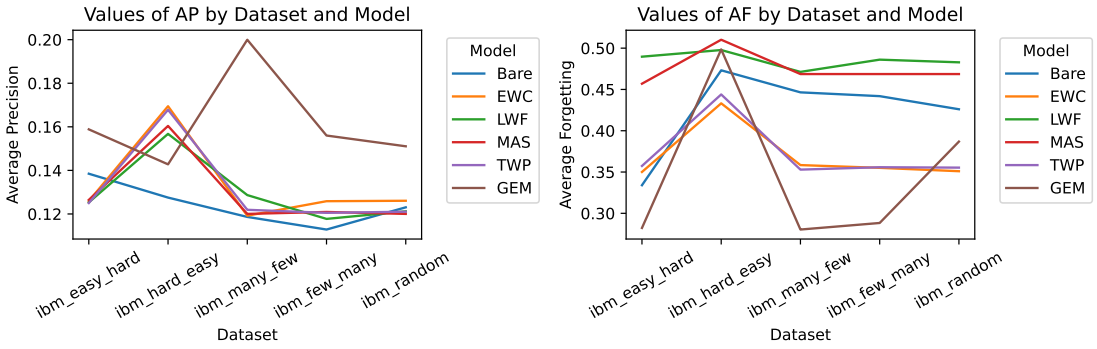

Figure 12: Global average of the average performance (left) and average forgetting (right) for the methods across the different permutations of the patterns.

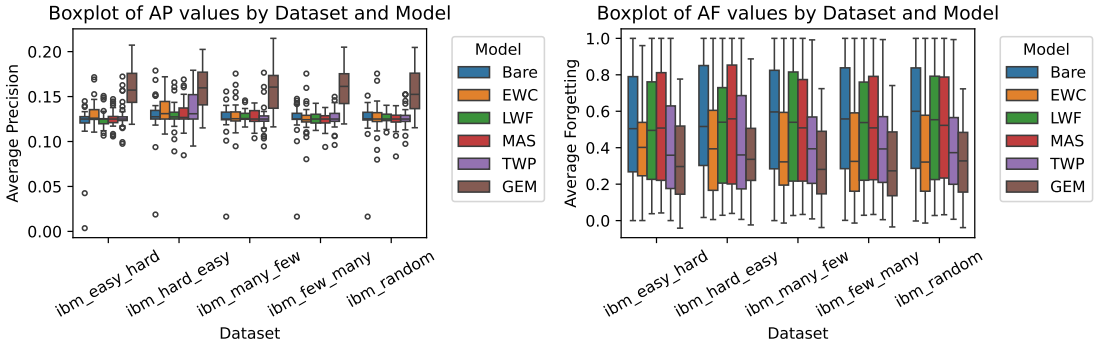

Figure 13: Boxplots of the average performance (left) and average forgetting (right) for the methods across the different permutations of the patterns.

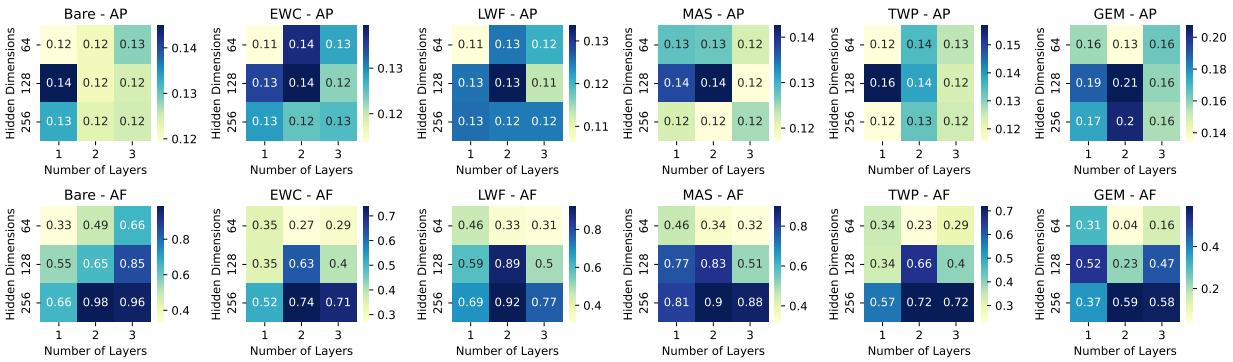

Figure 14: The average performance and average forgetting across different depths and widths of the GCN, when training for five epochs per task on the IBM dataset.

performance in the hard-to-easy setting, than for the others. This might indicate that these regularisation-based methods perform a bit better in this setting, although differences are minor.

These results illustrate that the methods are quite stable when faced with data perturbations.

5.3 Architecture of Backbone

As part of RQ2, we analyse the effect of the depth and width of the GCN model. The results in Figure 6 already show that the architecture of the GCN - both in terms of depth and width - has an impact on performance and forgetting. We provide heatmaps of the average performance and forgetting for all methods to have a detailed view on the results. The results for the IBM dataset on the easy-to-hard dataset for five epochs are shown in Figure 14. The results for the Elliptic dataset for five epochs with seven and 49 tasks are shown in Figure 15 and Figure 16, respectively.

Although intuitively more layers should be better in terms of average performance, the results do not show a clear dominance of two or three layers over one layer in the GCN. On the other hand, we can clearly see that forgetting is more severe if the GCN has more parameters. Deeper and wider GCNs tend to overfit more on the latest task.

5.4 Continual Learning Method

To evaluate the effect of the continual learning method to answer RQ3, we first look at the performance matrices in Figure 9-11. Visually, it seems that GEM most often retains previous knowledge both for a low as well as high number of epochs, while also learning the new task.

We confirm this by assessing the box plots of the ranking of the models in Figure 17 and Figure 18 for the IBM and Elliptic dataset, respectively, where the best performing method gets rank 1. On average, GEM scores best both in terms of low average forgetting as well as high average performance for both datasets.

When looking at the IBM results, EWC and TWP seems to also perform quite strongly. For the other

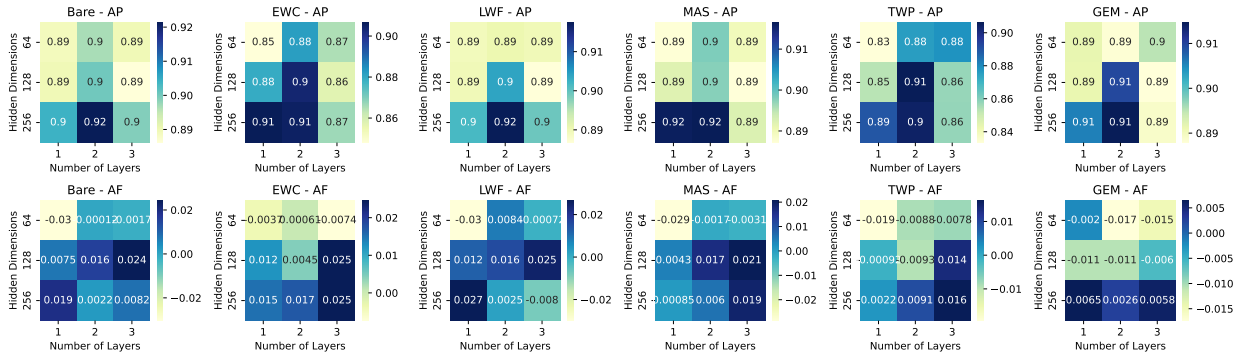

Figure 15: The average performance and average forgetting across different depths and widths of the GCN, when training for five epochs per task on the elliptic dataset with seven tasks.

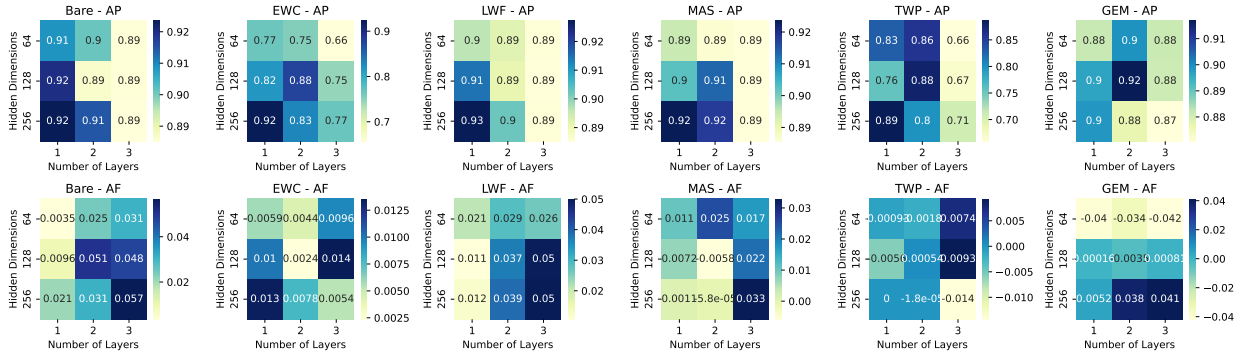

Figure 16: The average performance and average forgetting across different depths and widths of the GCN, when training for five epochs per task on the elliptic dataset with 49 tasks.

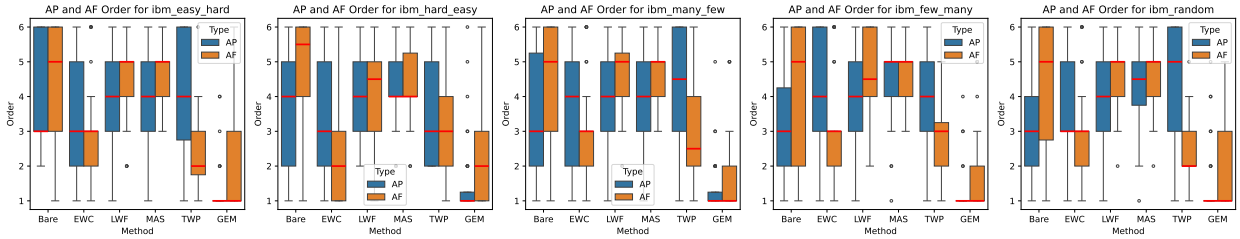

Figure 17: Box plot of the order of the methods per permutation of the patterns on the IBM dataset.

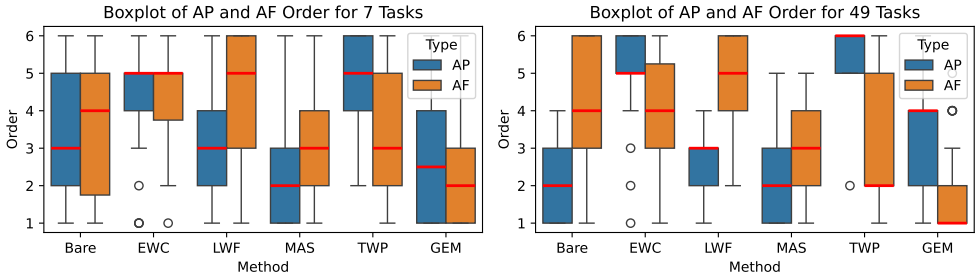

Figure 18: Box plot of the order of the methods for the number of tasks on the elliptic dataset.

methods, it seems that there is a trade-off between forgetting and performance. TWP, which is partially based on EWC, seems to be able to reduce forgetting compared to EWC by incorporating network information based on the previous tasks as well.

For the Elliptic dataset, the results are slightly different, here MAS performs well, while EWC is performing quite poorly. For the other methods, we again see a trade-off between forgetting and performance. Surprisingly, TWP has among the lowest average performance across all experiments, together with EWC, especially when dealing with 49 tasks. One explanation might be that the Fisher information is not sufficient to retain important weights across this many tasks. Another explanation is that the Elliptic dataset consist of 49 separate and distinct networks, which might explain why TWP has difficulties leveraging network information in its topology-related objective function.

An observation made for all iterations is that continual learning methods seem to result in better AML detection methods than the bare and joint model. Looking at the performance matrices of the joint model, we see strong performance for the first task, concerning the majority class, but low performance on the other tasks. This results from the extreme class imbalance in AML. The joint model seems to learn to always predict the majority class when minimising the loss function.

We also consider the performance of the model on the full test set, after seeing all tasks. The results on the IBM dataset are given in Figure 19. As expected, the joint model outperforms all other models, since it was trained on all data. We also see that GEM performs better than the other continual learning and bare models. This is due to the combination of high average performance and low average forgetting.

Similar results are provided for the Elliptic dataset in Figure 20. Here, the picture is less pronounced that

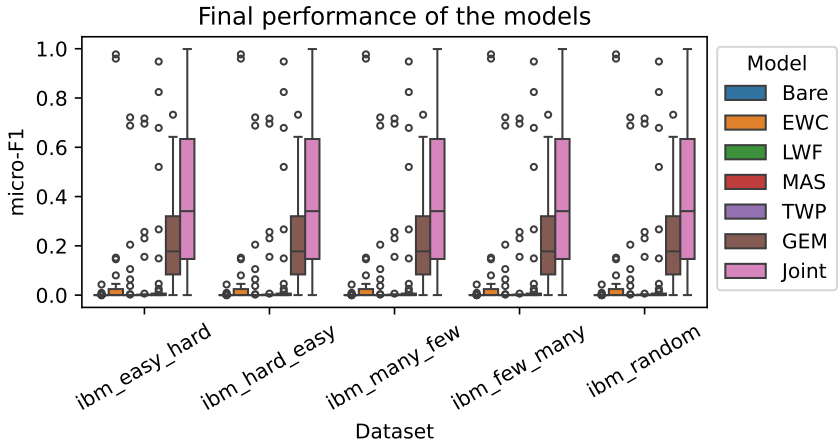

Figure 19: Box plot of the performance of the final model on all test data for the IBM dataset.

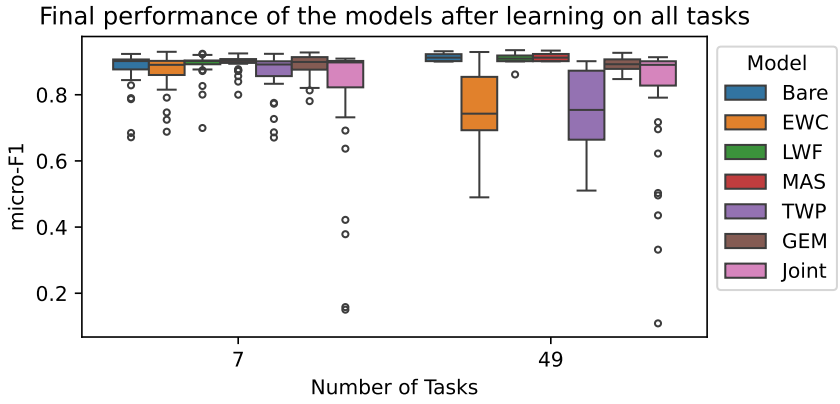

Figure 20: Boxplot of the performance of the final model on all test data for the Elliptic dataset.

before. We see consistently strong results for Bare, LwF, MAS and GEM. Contrary to the IBM dataset, TWP does not perform as well on the Elliptic dataset. TWP is the only method tested that incorporates network topology. Its performance might suffer from the Elliptic dataset being 49 disjoint and distinct networks, while each task of the IBM dataset is part of the same network.

Surprisingly, the joint model seems to perform slightly worse than these models. Given that it is trained on all data simultaneously, the joint model might suffer from the inclusion of the sudden closure of a dark market at time step 43 (Weber et al., 2019).

In summary, we report the following findings for the different choices:

- **Epochs.** Catastrophic forgetting happens suddenly and not gradually with a growing number of epochs. Additionally, the models overfit on the current task if the number of epochs is set too high. Therefore, this should be tuned carefully for AML.

- **Pattern Order.** In our experiments, the order of patterns did not have a significant impact on the performance. This is in line with previous research and indicates that for AML as well methods exhibit order-agnostic behaviour
- **Backbone Architecture.** Although performance gain is limited when going from two to three layers, forgetting is worse for deeper and wider models, as these tend to overfit on the current task.
- **Continual Learning Method.** Across the experiments, GEM which is based on replay in combination with regularisation, performed best in terms of obtained performance and least forgetting. Regularisation with Fisher information seems to suffer most when faced with many tasks. To a lesser extend, network-based methods only seemed to add value when inter-task connections are present.

6 Conclusion

Continual learning is essential in AML because (1) millions of transactions need to be monitored continuously, (2) fraud tactics are constantly evolving, causing underlying data distributions to shift, and (3) regulatory constraints often limit the amount of historical data that can be stored. Therefore, this work addresses three key research questions.

We started with reviewing the current state of the continual graph learning literature for AML (RQ1). We conclude that despite its ability to tackle these challenges well, continual graph learning for AML has received limited interest in the scientific literature.

To expand on the current body of knowledge, we present the results of a comprehensive experiment on continual graph learning on two AML datasets. We presented experiments for node and edge classification. We investigated the effect of the hyperparameters including the task order and the GNN architecture (RQ2), and the different continual learning methods (RQ3).

We conclude that increasing the number op epochs per task too much may lead to overfitting on the present task, and hence to forgetting. With regard to the task order, our experiments are in line with previous work and confirm that there is no significant effect of the task order on performance.

Based on the experimental results, we conclude that wide models are more prone to forgetting, and a balance needs to be struck between capturing longer money laundering chains and avoiding over-smoothing and forgetting when setting the depth of the GNN.

Across the experiments, GEM performed well with minimal forgetting. This indicates that replay capabilities increases model capabilities when applied to the complex problem of AML. As noted, their application in practice might be hindered by regulations limiting the storage of transaction data.

A surprising result is obtained regarding the joint model. The experiments show that continual learning methods are better at learning the different fraud patterns. When provided with all data, the joint model learns to consistently predict the majority class.

Based on our work, we have identified five gaps that future research must address:

1. **Nature of fraud.** Basic research on the effect of the nature of fraud on continual learning is missing. Future research should investigate how continual learning enhances graph learning when fraudsters rotate between patterns or when their are periods without fraud labels. These insights are critical to obtain better intuition and insights on continual graph learning for AML, and to develop custom solutions.

2. **Label distribution.** A key challenge in AML is the extreme class imbalance and undetected cases. Although continual learning exhibited some improvement over the joint model in the face of class imbalance, only limited work addresses this (Wei et al., 2025). Future work could focus on how to handle class imbalance for continual graph learning by, e.g., using sampling methods. Additionally, future research should incorporate the fact that there are undetected cases by analysing the effect of methods from PU-learning for continual graph learning.

3. **Recent advancement.** Applications of the latest network analytics methods seems generally slower for AML compared to other fields, as AML is often perceived as just another fraud detection application. However, AML presents a unique combination of challenges. Therefore, future work should investigate how well current advancements like graph transformers, graph auto-encoders and foundation models perform at continual graph learning for AML.

4. **Extended benchmark.** The main limitation of this work is the limited number of datasets and methods applied. A more extensive study is hampered as some key functionalities are missing in the BeGin framework. The BeGin framework should be extended to incorporate domain-incremental learning for edge classification and to have more methods applicable for edge classification. Additionally, more GNN backbones should be tested.

5. **Stability and robustness.** Although continual graph learning increases model stability by design and although this work has already provided a reflection on stability for the different hyperparameters, more attention should be given to stability and robustness. In particular, future research should analyse the stability of the individual predictions, not just of the global performance of the model. Positive results would increase credibility and trustworthiness of continual graph learning for AML.

Author Contributions

Bruno Deprez: Conceptualization, Data Curation, Methodology, Software, Visualization, Writing – Original Draft Preparation. **Wei Wei:** Conceptualization, Methodology, Software, Writing – Review & Editing. **Wouter Verbeke:** Conceptualization, Supervision, Writing – Review & Editing. **Bart Baesens:** Conceptualization, Supervision, Writing – Review & Editing. **Kevin Mets:** Conceptualization, Methodology, Supervision, Writing – Review & Editing. **Tim Verdonck:** Conceptualization, Methodology, Supervision, Writing – Review & Editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Muhammad Abulaish, Nesar Ahmad Wasi, and Shachi Sharma. The role of lifelong machine learning in bridging the gap between human and machine learning: A scientometric analysis. *WIREs Data Mining and Knowledge Discovery*, 14(2):e1526, 2024. doi:https://doi.org/10.1002/widm.1526. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1526.

Ismail Alarab, Simant Prakoonwit, and Mohamed Ikbal Nacer. Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, ICMLT '20, page 23–27, Beijing, China, 2020. Association for Computing Machinery. ISBN 9781450377645. doi:10.1145/3409073.3409080. URL https://doi.org/10.1145/3409073.3409080.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Erik Altman, Jovan Blanuša, Luc von Niederhäusern, Beni Egressy, Andreea Anghel, and Kubilay Atasu. Realistic synthetic financial transactions for anti-money laundering models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 29851–29874. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5f38404edff6f3f642d6fa5892479c42-Paper-Datasets_and_Benchmarks.pdf.

Bart Baesens, Veronique Van Vlasselaer, and Wouter Verbeke. *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection.* John Wiley & Sons, Inc, 2015. ISBN 9781119133124. doi:10.1002/9781119146841.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi:10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

Richard J. Bolton and David J. Hand. Statistical Fraud Detection: A Review. *Statistical Science*, 17(3):235 – 255, 2002. doi:10.1214/ss/1042727940. URL https://doi.org/10.1214/ss/1042727940.

Mário Cardoso, Pedro Saleiro, and Pedro Bizarro. Laundrograph: Self-supervised graph representation learning for anti-money laundering. In *Proceedings of the Third ACM International Conference on AI in Finance*, ICAIF '22, pages 130–138, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393768. doi:10.1145/3533271.3561727. URL https://doi.org/10.1145/3533271.3561727.

Antonio Carta, Andrea Cossu, Federico Errica, and Davide Bacciu. Catastrophic forgetting in deep graph networks: an introductory benchmark for graph classification, 2021. URL https://arxiv.org/abs/2103.11750.

Sungmin Cha and Kyunghyun Cho. Hyperparameters in continual learning: A reality check. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=hiiRCXmbAz.

Zhiyuan Chen, Le Dinh Van Khoa, Ee Na Teoh, Amril Nazir, Ettikan Kandasamy Karuppiah, and Kim Sim Lam. Machine learning techniques for anti-money laundering (aml) solutions in suspicious transaction detection: a review. *Knowledge and Information Systems*, 57(2):245–285, 2018. doi:10.1007/s10115-017-1144-z. URL https://doi.org/10.1007/s10115-017-1144-z.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022. doi:10.1109/TPAMI.2021.3057446.

Bruno Deprez, Toon Vanderschueren, Bart Baesens, Tim Verdonck, and Wouter Verbeke. Network analytics for anti-money laundering – a systematic literature review and experimental evaluation. *arXiv preprint arXiv:2405.19383*, 2024a. URL https://arxiv.org/abs/2405.19383.

Bruno Deprez, Félix Vandervorst, Wouter Verbeke, Tim Verdonck, and Bart Baesens. Network analytics for insurance fraud detection: a critical case study. *European Actuarial Journal*, 14(3):965–990, 2024b. doi:10.1007/s13385-024-00384-6. URL https://doi.org/10.1007/s13385-024-00384-6.

Rafał Drężewski, Jan Sepielak, and Wojciech Filipkowski. The application of social network analysis algorithms in a system supporting money laundering detection. *Information Sciences*, 295:18–32, 2015. ISSN 0020-0255. doi:https://doi.org/10.1016/j.ins.2014.10.015. URL https://www.sciencedirect.com/science/article/pii/S0020025514009979.

Béni Egressy, Luc von Niederhäusern, Jovan Blanuša, Erik Altman, Roger Wattenhofer, and Kubilay Atasu. Provably powerful graph neural networks for directed multigraphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(10):11838–11846, Mar. 2024. doi:10.1609/aaai.v38i10.29069. URL https://ojs.aaai.org/index.php/AAAI/article/view/29069.

Elliptic. Elliptic. www.elliptic.co. Accessed: 2024-01-31.

Falih Gozi Febrinanto, Feng Xia, Kristen Moore, Chandra Thapa, and Charu Aggarwal. Graph lifelong learning: A survey. *IEEE Computational Intelligence Magazine*, 18(1):32–51, 2023. doi:10.1109/MCI.2022.3222049.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3 (4):128–135, 2025/02/25 1999. doi:10.1016/S1364-6613(99)01294-2. URL https://doi.org/10.1016/S1364-6613(99)01294-2.

Andrea Fronzetti Colladon and Elisa Remondi. Using social network analysis to prevent money laundering. *Expert Systems with Applications*, 67:49–58, 2017. ISSN 0957-4174. doi:https://doi.org/10.1016/j.eswa.2016.09.029. URL https://www.sciencedirect.com/science/article/pii/S0957417416305139.

Lukas Galke, Iacopo Vagliano, Benedikt Franke, Tobias Zielke, Marcel Hoffmann, and Ansgar Scherp. Lifelong learning on evolving graphs under the constraints of imbalanced classes and new classes. *Neural Networks*, 164:156–176, 2023. ISSN 0893-6080. doi:https://doi.org/10.1016/j.neunet.2023.04.022. URL https://www.sciencedirect.com/science/article/pii/S0893608023002083.

Zengan Gao and Mao Ye. A framework for data mining-based anti-money laundering research. *Journal of Money Laundering Control*, 10(2):170–179, 2007. doi:10.1108/13685200710746875. URL https://doi.org/10.1108/13685200710746875.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2015. URL https://arxiv.org/abs/1312.6211.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

Hamed Hemati, Marco Schreyer, and Damian Borth. Continual learning for unsupervised anomaly detection in continuous auditing of financial accounting data, 2022. URL https://arxiv.org/abs/2112.13215.

Rasmus Ingemann Tuffveson Jensen and Alexandros Iosifidis. Fighting money laundering with statistics and machine learning. *IEEE Access*, 11:8889–8903, 2023. doi:10.1109/ACCESS.2023.3239549.

Chengxiang Jin, Jie Jin, Jiajun Zhou, Jiajing Wu, and Qi Xuan. Heterogeneous feature augmentation for ponzi detection in ethereum. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(9): 3919–3923, 2022. doi:10.1109/TCSII.2022.3177898.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi:10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Jihoon Ko, Shinhwan Kang, Taehyung Kwon, Heechan Moon, and Kijung Shin. Begin: Extensive benchmark scenarios and an easy-to-use framework for graph continual learning, 2024. URL https://arxiv.org/abs/2211.14568.

Eren Kurshan and Hongda Shen. Graph computing for financial crime and fraud detection: Trends, challenges and outlook. *International Journal of Semantic Computing*, 14(04):565–589, 2020.

B. Lebichot, W. Siblini, G.M. Paldino, Y.-A. Le Borgne, F. Oblé, and G. Bontempi. Assessment of catastrophic forgetting in continual credit card fraud detection. *Expert Systems with Applications*, 249:123445, 2024. ISSN 0957-4174. doi:https://doi.org/10.1016/j.eswa.2024.123445. URL https://www.sciencedirect.com/science/article/pii/S0957417424003105.

Sebastian Lee, Sebastian Goldt, and Andrew Saxe. Continual learning in the teacher-student setup: Impact of task similarity. In *International Conference on Machine Learning*, pages 6109–6119. PMLR, 2021.

Michael Levi and Peter Reuter. Money laundering. *Crime and justice*, 34(1):289–375, 2006. doi:10.1086/501508.

An Li, Zhongshuai Wang, Minghao Yu, and Di Chen. Blockchain abnormal transaction detection method based on weighted sampling neighborhood nodes. In *2022 3rd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, pages 746–752. IEEE, 2022a. doi:10.1109/ICBAIE56435.2022.9985815.

Qimai Li, Zhichao Han, and Xiao-ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi:10.1609/aaai.v32i1.11604. URL https://ojs.aaai.org/index.php/AAAI/article/view/11604.

Xiangfeng Li, Shenghua Liu, Zifeng Li, Xiaotian Han, Chuan Shi, Bryan Hooi, He Huang, and Xueqi Cheng. Flowscope: Spotting money laundering based on graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4731–4738, 2020. doi:10.1609/aaai.v34i04.5906. URL https://ojs.aaai.org/index.php/AAAI/article/view/5906.

Yujie Li, Yuxuan Yang, Xin Yang, Qiang Gao, and Fan Zhou. Forgetting prevention for cross-regional fraud detection with heterogeneous trade graph, 2022b. URL https://arxiv.org/abs/2204.10085.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. doi:10.1109/TPAMI.2017.2773081.

Ziyu Li, Yanmei Zhang, Qian Wang, and Shiping Chen. Transactional network analysis and money laundering behavior identification of central bank digital currency of china. *Journal of Social Computing*, 3(3):219–230, 2022c. doi:10.23919/JSC.2022.0011.

J. Lian, K. Choi, B. Veeramani, A. Hu, S. Murli, L. Freeman, E. Bowen, and X. Deng. Continual learning and its industrial applications: A selective review. *WIREs Data Mining and Knowledge Discovery*, 14(6): e1558, 2024. doi:https://doi.org/10.1002/widm.1558. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1558.

Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming catastrophic forgetting in graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):8653–8661, May 2021. doi:10.1609/aaai.v35i10.17049. URL https://ojs.aaai.org/index.php/AAAI/article/view/17049.

Yilun Liu, Ruihong Qiu, and Zi Huang. Cat: Balanced continual graph learning with graph condensation. In *2023 IEEE International Conference on Data Mining (ICDM)*, pages 1157–1162, 2023. doi:10.1109/ICDM58522.2023.00141.

Wai Weng Lo, Gayan K. Kulatilleke, Mohanad Sarhan, Siamak Layeghy, and Marius Portmann. Inspection-l: self-supervised gnn node embeddings for money laundering detection in bitcoin. *Applied Intelligence*, 53(16):19406–19417, 2023. doi:10.1007/s10489-023-04504-9. URL https://doi.org/10.1007/s10489-023-04504-9.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

Joana Lorenz, Maria Inês Silva, David Aparício, João Tiago Ascensão, and Pedro Bizarro. Machine learning methods to detect money laundering in the bitcoin blockchain in the presence of label scarcity. In *Proceedings of the First ACM International Conference on AI in Finance*, ICAIF '20, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450375849. doi:10.1145/3383455.3422549. URL https://doi.org/10.1145/3383455.3422549.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. doi:10.1109/CVPR.2018.00810.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss,

editors, *Computer Vision – ECCV 2018*, pages 72–88, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01225-0.

Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018. doi:10.1073/pnas.1803839115. URL https://www.pnas.org/doi/abs/10.1073/pnas.1803839115.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989. doi:https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning, 2022. URL https://arxiv.org/abs/2202.00275.

Anuraj Mohan, Karthika P. V., Parvathi Sankar, K. Maya Manohar, and Amala Peter. Improving anti-money laundering in bitcoin using evolving graph convolutions and deep neural decision forest. *Data Technologies and Applications*, 57(3):313–329, 2023. doi:10.1108/DTA-06-2021-0167. URL https://doi.org/10.1108/DTA-06-2021-0167.

Soroor Motie and Bijan Raahemi. Financial fraud detection using graph neural networks: A systematic review. *Expert Systems with Applications*, 240:122156, 2024. ISSN 0957-4174. doi:https://doi.org/10.1016/j.eswa.2023.122156. URL https://www.sciencedirect.com/science/article/pii/S0957417423026581.

E.W.T. Ngai, Yong Hu, Y.H. Wong, Yijun Chen, and Xin Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50(3):559–569, 2011. ISSN 0167-9236. doi:https://doi.org/10.1016/j.dss.2010.08.006. URL https://www.sciencedirect.com/science/article/pii/S0167923610001302. On quantitative methods for detection of financial fraud.

Cuong V. Nguyen, Alessandro Achille, Michael Lam, Tal Hassner, Vijay Mahadevan, and Stefano Soatto. Toward understanding catastrophic forgetting in continual learning, 2019. URL https://arxiv.org/abs/1908.01091.

Carlos Ortega Vázquez, Seppe vanden Broucke, and Jochen De Weerdt. A two-step anomaly detection based method for pu classification in imbalanced data sets. *Data Mining and Knowledge Discovery*, 37(3):1301–1325, 2023. doi:10.1007/s10618-023-00925-9. URL https://doi.org/10.1007/s10618-023-00925-9.

María Óskarsdóttir, Waqas Ahmed, Katrien Antonio, Bart Baesens, Rémi Dendievel, Tom Donas, and Tom Reynkens. Social network analytics for supervised fraud detection in insurance. *Risk Analysis*, 42(8): 1872–1890, 2022. doi:10.1111/risa.13693.

Michael Ovelgönne, Chanhyun Kang, Anshul Sawant, and VS Subrahmanian. Covertness centrality in networks. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 863–870. IEEE, 2012. doi:10.1109/ASONAM.2012.156.

Berkan Oztas, Deniz Cetinkaya, Festus Adedoyin, Marcin Budka, Gokhan Aksu, and Huseyin Dogan. Transaction monitoring in anti-money laundering: A qualitative analysis and points of view from industry. *Future Generation Computer Systems*, 159:161–171, 2024.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. ISSN 0893-6080. doi:https://doi.org/10.1016/j.neunet.2019.01.012. URL https://www.sciencedirect.com/science/article/pii/S0893608019300231.

Massimo Perini, Giorgia Ramponi, Paris Carbone, and Vasiliki Kalavri. Learning on streaming graphs with experience replay. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, SAC '22, page 470–478, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387132. doi:10.1145/3477314.3507113. URL https://doi.org/10.1145/3477314.3507113.

Silivanxay Phetsouvanh, Frédérique Oggier, and Anwitaman Datta. Egret: Extortion graph exploration techniques in the bitcoin network. In *2018 IEEE International conference on data mining workshops (ICDMW)*, pages 244–251. IEEE, 2018. doi:10.1109/ICDMW.2018.00043.

M.I. Pramanik, Raymond Y.K. Lau, Wei T. Yue, Yunming Ye, and Chunping Li. Big data analytics for security and criminal investigations. *WIREs Data Mining and Knowledge Discovery*, 7(4):e1208, 2017. doi:https://doi.org/10.1002/widm.1208. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1208.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf.

Franco Scarselli, Sweah Liang Yong, Marco Gori, Markus Hagenbuchner, Ah Chung Tsoi, and Marco Maggini. Graph neural networks for ranking web pages. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pages 666–672. IEEE, 2005. doi:10.1109/WI.2005.67.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. doi:10.1109/TNN.2008.2005605.

Ted E Senator, Henry G Goldberg, Jerry Wooton, Matthew A Cottini, AF Umar Khan, Christina D Klinger, Winston M Llamas, Michael P Marrone, and Raphael WH Wong. Financial crimes enforcement network ai system (fais) identifying potential money laundering from reports of large cash transactions. *AI magazine*, 16(4):21–21, 1995. doi:10.1609/aimag.v16i4.1169.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4548–4557. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/serra18a.html.

Kai Sun, Kun Meng, and Ziqiang Zheng. Game-bc: A graph attention model for exploring bitcoin crime. In *2022 6th International Symposium on Computer Science and Intelligent Control (ISCSIC)*, pages 342–346. IEEE, 2022. doi:10.1109/ISCSIC57216.2022.00077.

Haseeb Tariq and Marwan Hassani. Topology-agnostic detection of temporal money laundering flows in billion-scale transactions. In Rosa Meo and Fabrizio Silvestri, editors, *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 402–419, Cham, 2025. Springer Nature Switzerland.

Zonggui Tian, Du Zhang, and Hong-Ning Dai. Continual learning on graphs: A survey. *arXiv preprint arXiv:2402.06330*, 2024.

United Nations Office on Drugs and Crime (UNOCD). Money laundering. https://www.unodc.org/unodc/en/money-laundering/overview.html. Accessed: 2023-04-07.

Rafaël Van Belle, Sandra Mitrović, and Jochen De Weerdt. Representation learning in graphs for credit card fraud detection. In Valerio Bitetta, Ilaria Bordino, Andrea Ferretti, Francesco Gullo, Stefano Pascolutti, and Giovanni Ponti, editors, *Mining Data for Financial Applications*, pages 32–46, Cham, 2020. Springer International Publishing. ISBN 978-3-030-37720-5.

Rafaël Van Belle and Jochen De Weerdt. Shine: A scalable heterogeneous inductive graph neural network for large imbalanced datasets. *IEEE Transactions on Knowledge and Data Engineering*, 36(9):4904–4915, 2024. doi:10.1109/TKDE.2024.3381240.

Rafaël Van Belle, Charles Van Damme, Hendrik Tytgat, and Jochen De Weerdt. Inductive graph representation learning for fraud detection. *Expert Systems with Applications*, 193:116463, 2022. ISSN 0957-4174. doi:https://doi.org/10.1016/j.eswa.2021.116463. URL https://www.sciencedirect.com/science/article/pii/S0957417421017449.

Rafaël Van Belle, Bart Baesens, and Jochen De Weerdt. Catchm: A novel network-based credit card fraud detection method using node representation learning. *Decision Support Systems*, 164:113866, 2023. ISSN 0167-9236. doi:https://doi.org/10.1016/j.dss.2022.113866. URL https://www.sciencedirect.com/science/article/pii/S0167923622001373.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022. doi:10.1038/s42256-022-00568-3. URL https://doi.org/10.1038/s42256-022-00568-3.

Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 75:38–48, 2015. ISSN 0167-9236. doi:https://doi.org/10.1016/j.dss.2015.04.013. URL https://www.sciencedirect.com/science/article/pii/S0167923615000846.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. Streaming graph neural networks via continual learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 1515–1524, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi:10.1145/3340531.3411963. URL https://doi.org/10.1145/3340531.3411963.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8): 5362–5383, 2024. doi:10.1109/TPAMI.2024.3367329.

Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics, 2019. URL https://arxiv.org/abs/1908.02591.

Wei Wei, Tom De Schepper, and Kevin Mets. Benchmarking sensitivity of continual graph learning for skeleton-based action recognition. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, page 639–651. SCITEPRESS - Science and Technology Publications, 2024a. doi:10.5220/0012394400003660. URL http://dx.doi.org/10.5220/0012394400003660.

Wei Wei, Tom De Schepper, and Kevin Mets. Dataset condensation with latent quantile matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 7703–7712, June 2024b.

Wei Wei, Matthias Hutsebaut-Buysse, Tom De Schepper, and Kevin Mets. Reducing the stability gap for continual learning at the edge with class balancing. *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2025.

Sarah N Welling. Smurfs, money laundering, and the federal criminal law: the crime of structuring transactions. *Fla. L. Rev.*, 41:287, 1989.

Pingfan Xia, Zhiwei Ni, Hongwang Xiao, Xuhui Zhu, and Peng Peng. A novel spatiotemporal prediction approach based on graph convolution neural networks and long short-term memory for money laundering fraud. *Arabian Journal for Science and Engineering*, 47(2):1921–1937, 2022. doi:10.1007/s13369-021-06116-2. URL https://doi.org/10.1007/s13369-021-06116-2.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3014–3023, 2021.

Qiao Yuan, Sheng-Uei Guan, Pin Ni, Tianlun Luo, Ka Lok Man, Prudence Wong, and Victor Chang. Continual graph learning: A survey, 2023. URL https://arxiv.org/abs/2301.12230.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/zenke17a.html.

Peiyan Zhang, Yuchen Yan, Chaozhuo Li, Senzhang Wang, Xing Xie, Guojie Song, and Sunghun Kim. Continual learning on dynamic graphs via parameter isolation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 601–611, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086. doi:10.1145/3539618.3591652. URL https://doi.org/10.1145/3539618.3591652.

Rui Zhang, Dawei Cheng, Jie Yang, Yi Ouyang, Xian Wu, Yefeng Zheng, and Changjun Jiang. Pre-trained online contrastive learning for insurance fraud detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(20):22511–22519, Mar. 2024a. doi:10.1609/aaai.v38i20.30259. URL https://ojs.aaai.org/index.php/AAAI/article/view/30259.

Shilei Zhang, Toyotaro Suzumura, and Li Zhang. Dyngraphtrans: Dynamic graph embedding via modified universal transformer networks for financial transaction data. In *2021 IEEE International Conference on Smart Data Services (SMDS)*, pages 184–191, 2021. doi:10.1109/SMDS53860.2021.00032.

Xikun Zhang, Dongjin Song, and Dacheng Tao. Cglb: Benchmark tasks for continual graph learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 13006–13021. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/548a41b9cac6f50dccf7e63e9e1b1b9b-Paper-Datasets_and_Benchmarks.pdf.

Xikun Zhang, Dongjin Song, and Dacheng Tao. Continual learning on graphs: Challenges, solutions, and opportunities, 2024b. URL https://arxiv.org/abs/2402.11565.

Tianqi Zhao, Alan Hanjalic, and Megha Khosla. AGALE: A graph-aware continual learning evaluation framework. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=xDTKRLyaNN.

Maria Zhdanova, Jürgen Repp, Roland Rieke, Chrystel Gaber, and Baptiste Hemery. No smurfs: Revealing fraud chains in mobile money transfers. In *2014 Ninth International Conference on Availability, Reliability and Security*, pages 11–20. IEEE, 2014. doi:10.1109/ARES.2014.10.

Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. *Advances in Neural Information Processing Systems*, 36:6026–6047, 2023.

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9851–9873, 2024. doi:10.1109/TPAMI.2024.3429383.

Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. A local algorithm for structure-preserving graph cut. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 655–664, Halifax, NS, Canada, 2017. ISBN 9781450348874. doi:10.1145/3097983.3098015. URL https://doi.org/10.1145/3097983.3098015.

Fan Zhou and Chengtai Cao. Overcoming catastrophic forgetting in graph neural networks with experience replay. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4714–4722, May 2021. doi:10.1609/aaai.v35i5.16602. URL https://ojs.aaai.org/index.php/AAAI/article/view/16602.

Pengfei Zhu and Qinghua Hu. Rule extraction from support vector machines based on consistent region covering reduction. *Knowledge-Based Systems*, 42:1–8, 2013. ISSN 0950-7051. doi:https://doi.org/10.1016/j.knosys.2012.12.003. URL https://www.sciencedirect.com/science/article/pii/S095070511200336X.