THE UNREASONABLE EFFECTIVENESS OF RANDOMIZED REPRESENTATIONS IN ONLINE CONTINUAL GRAPH LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Catastrophic forgetting is one of the main obstacles for Online Continual Graph Learning (OCGL), where nodes arrive one by one, distribution drifts may occur at any time and offline training on task-specific subgraphs is not feasible. In this work, we explore a surprisingly simple yet highly effective approach for OCGL: we use a fixed, randomly initialized encoder to generate robust and expressive node embeddings by aggregating neighborhood information, training online only a lightweight classifier. By freezing the encoder, we eliminate drifts of the representation parameters, a key source of forgetting, obtaining embeddings that are both expressive and stable. When evaluated across several OCGL benchmarks, despite its simplicity and lack of memory buffer, this approach yields consistent gains over state-of-the-art methods, with surprising improvements of up to 30% and performance often approaching that of the joint offline-training upper bound. These results suggest that in OCGL, catastrophic forgetting can be minimized without complex replay or regularization by embracing architectural simplicity and stability.

1 INTRODUCTION

In Online Continual Graph Learning (OCGL), nodes arrive sequentially, are observed only once, and undergo drifts in both distribution and task to solve (Donghi et al., 2025). Therefore, models must adapt to new operating conditions on the fly, while making anytime predictions and retaining past knowledge from limited observations, and under strict memory and latency constraints. This makes OCGL one of the most challenging continual learning scenarios. This setting poses several additional requirements to the traditional Continual Learning (CL) (Parisi et al., 2019; De Lange et al., 2022; Van De Ven et al., 2022), Online CL (Mai et al., 2022), and Continual Graph Learning (CGL) (Zhang et al., 2022a), enabling applications that require fast adaptations and anytime predictions (Koh et al., 2021) such as in healthcare (Le Baher et al., 2023), IoT intrusion detection (Lin et al., 2024) and financial markets (Weber et al., 2019), other than modeling citation networks (Liu et al., 2021; Zhou & Cao, 2021), transportation networks (Chen et al., 2021) and recommender systems (Caroprese et al., 2025).

In this paper, we introduce a simple, yet surprisingly effective approach for node-level OCGL, decoupling node representations from the predictive model, and yielding excellent prediction accuracy while taming the forgetting of previously learned concepts. Our approach leverages the effectiveness of randomized and over-parametrized architectures to provide rich, untrained representations, combined with a lightweight trained classifier on top (Rahimi & Recht, 2008; Rudi & Rosasco, 2017b; Scardapane & Wang, 2017). As demonstrated in related literature, fixed representations from pre-trained models help prevent forgetting at the feature extraction level even when used in CL for image classification (Hayes & Kanan, 2020; Pelosin, 2022; Mehta et al., 2023). The beneficial effect appears to be even more striking in graphs, where stable neighborhood embeddings eliminate the need to retain the entire topological information for replay strategies (Zhang et al., 2024a). Additionally, the literature on randomized models provides us results that ensure good approximation and generalization properties for this family of models, given that we use a sufficiently large embedding dimension (Scardapane & Wang, 2017). Coupled with a Streaming Linear Discriminant Analy-

sis (SLDA) classifier (Hayes & Kanan, 2020), this approach generally outperforms state-of-the-art OCGL methods, without need for a memory buffer.

Our main contributions are the following.

1. We propose a surprisingly strong OCGL method. A simple, yet effective method coupling principled untrained features with a lightweight streaming classifier, granting both expressivity and forgetting-resilience.
2. We demonstrate how the approach achieves generally best results across seven OCGL benchmarks, in both class-incremental and time-incremental setups.
3. We theoretically motivate our research findings, opening to the development of new, simpler CL methods without compromising on accuracy.

Beyond providing a strong new method with said advantages, our findings suggest that future OCGL research should rethink model designs, emphasizing architectural simplicity and stability.

2 Related works

In this section, we first outline the specifics of Online Continual Learning for graphs, the setting used for our experiments, distinguishing it from related paradigms. We then discuss randomized neural models and other fixed feature extraction strategies in Continual Learning. These concepts provide context for the problem addressed in the paper and motivate our design choices.

Online continual learning for graphs Continual Learning has been studied mainly in domains such as computer vision (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017) or reinforcement learning (Kirkpatrick et al., 2017; Rolnick et al., 2019), with clearly defined task boundaries and no dependencies between successive tasks. In contrast, working on a node level in a graph domain introduces structural dependencies between data points, complicating the definition of the task. Continual Graph Learning (CGL) (Febrinanto et al., 2023; Yuan et al., 2023; Zhang et al., 2024b) extends CL to graph settings, and assumes that the graph is presented in blocks, that is, as subsets of nodes connected in subgraphs (Zhang et al., 2022a). Several methods for CL on graph data have been proposed (Zhou & Cao, 2021; Liu et al., 2021; 2023; Hoang et al., 2023; Sun et al., 2023; Cui et al., 2023), yet, similarly to traditional CL, state-of-the-art performance is achieved by replay-based methods which leverage a memory buffer tailored to leverage graph topology (Zhang et al., 2022b; 2024a). However, the use of Graph Neural Network (GNN) (Scarselli et al., 2009; Micheli, 2009; Kipf & Welling, 2017) models poses an issue regarding task separation: nodes form connections also with nodes that were observed in past tasks, thus requiring access to past information due to message passing (Gilmer et al., 2017). This is sometimes addressed by ignoring inter-task edges (Zhang et al., 2022a), yet it is an unrealistic solution since it assumes supervisory information in the form of task identifiers also at test time. On the other hand, due to small-world nature of most real-world graphs, accessing the entire neighborhood information of the nodes would likely mean using the entirety of past data, rendering the use of the CL approach less meaningful. In this context, a more principled setting is Online Continual Graph Learning (OCGL) (Donghi et al., 2025), which bridges the gap between existing research on online CL and CGL. This setting addresses the issue of access to node neighborhoods with the use of neighborhood sampling, to keep a constant computational footprint even as the graph gets denser over time. In particular, the online setting for CL entails a single pass over the streaming data (Chaudhry et al., 2018; Mai et al., 2022; Soutif–Cormerais et al., 2023), in contrast to traditional offline CL where training is done offline in a batched setting on each task (De Lange et al., 2022). This constraint, together with stricter computational and memory requirements, is motivated by applications where quick model adaptation is necessary, and anytime predictions may be required (Koh et al., 2021), thus impeding the task-wise offline training.

Randomized representations In the literature on neural networks, multiple approaches have leveraged some form of randomization, due to training efficiency, and often theoretical guarantees (Scardapane & Wang, 2017). Notable examples for vector data are Random Vector Functional-Link networks (RVFL) (Pao & Takefuji, 1992; Pao et al., 1994), Extreme Learning Machines (ELM) (Huang et al., 2006; 2015) and Echo State Networks (ESN) (Jaeger & Haas, 2004). These fix most network parameters randomly, from an appropriate distribution, training only lightweight readouts. Theoretical results such as universal approximation properties and generalization guarantees have

been proved for some these architectures (Liu et al., 2012). Randomized strategies have also been explored for kernel approximation, such as with Random Fourier Features (RFF) (Rahimi & Recht, 2007), a randomized projection that approximates the RBF kernel. Some works have also explored randomized strategies for graph data, such as with GESN (Gallicchio & Micheli, 2010) or MRGNN (Pasa et al., 2022). Different instantiations of untrained Graph Convolutional Networks (Kipf & Welling, 2017) have been proposed in recent years: GCELM (Zhang et al., 2020) uses a single GCN layer, with randomly initialized and fixed parameters. GCN-RW (Huang et al., 2023) improves on this approach by considering an increased receptive field, employing the square of the adjacency matrix for aggregation. The more recent UGCN (Navarin et al., 2023a) instead uses multiple GCN layers, followed by non-differentiable pooling, as the network does not need to be trained. Inspired by RFF, GRNF (Zambon et al., 2020) are derived from expressive GNN architectures, and constitute a family that can separate graphs.

Fixed feature extraction for CL Some works in the CL literature for image classification suggest the use of a frozen CNN backbone, pre-trained on a different dataset, and then training continually only a classifier such a simple MLP (Van De Ven et al., 2022; Mehta et al., 2023). In the context of graph data, this approach is not currently feasible, as it is not trivial to obtain pre-trained GNNs that can handle graphs with different input feature domains, even though there is ongoing promising work on graph foundation models (Wang et al., 2025). Additionally, despite the many randomized approaches for representation extraction, only a very limited number of CL papers rely on them, and only with image classification tasks. CRNet (Li & Zeng, 2023) uses a randomly initialized network for feature extraction, training the classification head via closed-form solution on each task. RanPAC (McDonnell et al., 2023) uses random projections of feature extracted with a pre-trained model with the objective of increasing dimensionality to facilitate class separation. More recently, RanDumb (Prabhu et al., 2025) leverages Random Fourier Features (Rahimi & Recht, 2007) used with nearest class mean classification. On the other hand, there have been some works that investigated theoretically the use of linear models in simple CL setups (Evron et al., 2022; 2023; Ding et al., 2024), leading us to adopt these simple classifiers. Furthermore, it has been observed that prototype based classifiers, such as nearest class mean, are particularly suited for online CL, due to being schedule-robust, i.e. independent from stream ordering (Wang et al., 2022).

3 METHOD

The OCGL setting (Donghi et al., 2025) considers an incremental graph $\mathcal{G}$, induced by a stream of nodes $v_1, v_2, \ldots, v_t, \ldots$ that are added one by one. At each timestep t, the graph is updated with neighborhood and attribute information about the incoming node $(v_t, \mathcal{N}(v_t), \boldsymbol{x}^t)$, obtaining an updated graph snapshot $\mathcal{G}^{(t)} = (\mathbb{V}^{(t)}, \mathbb{E}^{(t)}, \boldsymbol{X}^{(t)})$. For example, in the Elliptic dataset (Weber et al., 2019), when a new transaction is processed, its information is captured as node features, and payment flows indicate connections. The goal is to learn a model F_Θ to make node-level predictions $\hat{y}_v^{(t)}$ for given node v at time step t given information coming from its neighboring nodes in $\mathcal{G}^{(t)}$. Specifically, models are trained with a single pass over the node stream, using information from the associated l-hop ego-graph $\mathcal{G}_v^{(t,l)}$. The online setting requests bounded memory and computational cost at each time steps, even as the graph grows. Accordingly, only a subset $\widetilde{\mathcal{G}}_v^{(t,l)}$ of the ego-graph may be allowed to produce prediction

$$\hat{y}_v^{(t)} = F_\Theta\left(\widetilde{\mathcal{G}}_v^{(t,l)}\right). \tag{1}$$

In the context of CL, the node stream can encompass diverse drift patterns, such as those triggered by a class-incremental setting. Nonetheless, model predictions may be requested for nodes observed in the past, therefore requesting the model not only to adapt to evolving graph conditions but also to preserve previously learned knowledge.

3.1 PROPOSED SOLUTION

To address forgetting in the OCGL setting, we propose a decoupled approach to node prediction. In this setup, model $F_\Theta = \Phi \circ \Psi$ decomposes as a node feature extractor Ψ followed by a node-level predictor Φ. Feature extraction is performed by a fixed (randomly initialized) backbone Ψ:

$$\boldsymbol{z}_v^{(t)} = \Psi\left(\widetilde{\mathcal{G}}_v^{(t,l)}\right), \tag{2}$$

where $\boldsymbol{z}_v^{(t)} \in \mathbb{R}^d$ denotes the node embedding that captures neighborhood information. As Ψ is untrained, it does not change over time and is inherently immune to forgetting. Secondly, it offers a clear advantage for experience replay methods, as storing $\boldsymbol{z}_v^{(t)}$ is significantly more memory-efficient than storing the entire sampled neighborhood $\widetilde{\mathcal{G}}_v^{(t,l)}$.

Feature extractor Ψ also mitigates other sources of forgetting. Unlike trained models, which tend to produce task-specific representations that remove task-irrelevant information, a well-chosen Ψ can produce rich node embeddings suitable for multiple tasks. Notably, theoretical results (Rudi & Rosasco, 2017a) show that for some families of model architectures, a random initialization of Ψ yields linearly separable embeddings, allowing to effectively solve the downstream task by simple linear predictive models

$$\hat{y}_v^{(t)} = \Phi_{\boldsymbol{W},\boldsymbol{b}}\left(\boldsymbol{z}_v^{(t)}\right) = \boldsymbol{W}\boldsymbol{z}_v^{(t)} + \boldsymbol{b}\,, \tag{3}$$

which are less prone to forgetting than deep networks (Mirzadeh et al., 2022) and can be learned efficiently.

3.2 Randomized feature extraction

We consider two different types of graph feature extractors Ψ, or backbones, for our experiments, untrained GCN (Navarin et al., 2023a) and Graph Random Neural Features (Zambon et al., 2020) adapted to the node-level, which we discuss here. In general, any randomized graph network with sufficient expressivity and robustness to limited structural shifts can be adopted for our decoupled approach.

UGCN As first feature extractor, wee use the recent UGCN (Navarin et al., 2023a), which uses multiple GCN layers $\boldsymbol{H}^{(i)} = tanh(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(i-1)}\Theta)$, where $\tilde{\boldsymbol{A}}$ is the normalized adjacency matrix, with $\boldsymbol{H}^{(0)} = \boldsymbol{X}$ and with weights initialized with a Glorot uniform approach (Glorot & Bengio, 2010) regulated by a gain hyperparameter, and successively left untrained. In our case, since we adopt it for node-level predictions instead of graph-level ones, we do not use the entire adjacency matrix, but the efficient forward propagation limited to the node ego-graph of (Hamilton et al., 2017). The output of the different layers is then concatenated to obtain an embedding that reflects information at multiple resolutions, i.e.

$$\Psi_{\text{UGCN}}\left(\widetilde{\mathcal{G}}_v^{(t,l)}\right) = \text{concat}\left(\boldsymbol{H}_v^{(1)}, \ldots, \boldsymbol{H}_v^{(l)}\right)\,. \tag{4}$$

In an over-parameterized regime, this model performs competitively with trained GCN counterparts on graph classification (Navarin et al., 2023b; Donghi et al., 2024).

GRNF Graph Random Neural Features (Zambon et al., 2020) define expressive embeddings for attributed graphs, derived from a GNN model which is a universal approximator of graph functions (Maron et al., 2019; Keriven & Peyré, 2019). Each GRNF is a parametric maps $\psi(\cdot, \boldsymbol{w}) : \mathcal{T}^2 \to \mathbb{R}$ defined by the composition of equivariant and invariant affine maps, interleaved with non-linearity:

$$\psi_\omega(\cdot) = \sigma \circ H_k(\cdot, \theta_H) \circ \sigma \circ F_{2,k}(\cdot, \theta_F)\,, \tag{5}$$

where $\omega = (k, \theta_F, \theta_H)$, and k represent tensor order (formal definitions of $F_{2,k}$ and H_k, and more details can be found in (Zambon et al., 2020)). Interestingly, the family of GRNF can separate graphs, meaning that for any non-isomorphic graphs $\mathcal{G}_1, \mathcal{G}_2$, there exists ω such that $\psi_\omega(\mathcal{G}_1) \neq \psi_\omega(\mathcal{G}_2)$. Thus, under suitable assumptions on the distribution of $\boldsymbol{w}$, the family of GRNF induces a metric distance on graphs, with also results that hold in probability for a fixed number of features (Zambon et al., 2020). To adapt GRNF to make node level predictions, we use them on $\widetilde{\mathcal{G}}_v^{(t,l)}$, and since the the output is permutation invariant over the entire ego-graph, we concatenate it to the component relative to v of the output of the equivariant map, using $d/2$ independent draws of ω, resulting in a d-dimensional embedding (assuming even d).

3.3 Linear classifier

As indicated in equation 3, the extracted features are fed to a linear layer, which can be trained continually on the stream. Effectively, thanks to this strategy we have simplified the Continual Graph Learning problem by eliminating the graph specificity, relegating it to the fixed feature extractor,

and reducing the problem to Continual Learning on vectors. A linear layer may thus be trained with gradient descent, supported by any standard CL learning strategy, such as experience replay which can now leverage informative node embeddings.

Furthermore, we can also address forgetting in the linear classifier by adopting a specific form which does not require gradients: we use Streaming Linear Discriminant Analysis (SLDA) (Hayes & Kanan, 2020), which learns by accumulating class means, independently for each class. This choice makes training more efficient and provides further robustness against catastrophic forgetting risk, as further discussed in Section 3.4. Specifically, during training, SLDA keeps a cumulative mean μ_y for the features of each class y, and a shared covariance matrix Σ with streaming updates. To make predictions, using precision matrix $\Lambda = [(1-\epsilon)\Sigma + \epsilon \boldsymbol{I}]^{-1}$ with $\epsilon = 10^{-4}$, the parameters $\boldsymbol{w}_y$ (rows of $\boldsymbol{W}$) and $\boldsymbol{b}$ of equation 3 are computed as

$$\boldsymbol{w}_y = \Lambda \mu_y \,, \qquad \boldsymbol{b}_y = -\frac{1}{2}(\mu_y^T \Lambda \mu_y) \,. \tag{6}$$

3.4 Effectiveness of the approach

As the graph evolves, multiple sources of forgetting can impair the model's prediction accuracy. This section elaborates on our method's effectiveness in mitigating forgetting while yielding high anytime classification accuracy.

One is a common challenge in all CL setups and stems from the continual update of the model's parameters based on recent training signals, both for the feature extraction backbone (if trained) and for the classifier. A second, graph-specific issue arises from the evolving nature of the graphs and associated structural shifts (Su et al., 2023; 2024): predictions made for the same node at different time steps may rely on different and potentially inconsistent neighborhoods. To put our intuitions more formally, we consider time interval δt and define the forgetting risk for node v at time t as the increase in the loss from time $t_0 = t - \delta t$ to time t:

$$\Delta\mathcal{R}_v = \Delta\mathcal{R}_{v,t}(\delta t) = \ell\left(y, \Phi_{W_t}\left(\Psi_{\Theta_t}\left(\mathcal{G}_v^t\right)\right)\right) - \ell\left(y, \Phi_{W_{t_0}}\left(\Psi_{\Theta_{t_0}}\left(\mathcal{G}_v^{t_0}\right)\right)\right) \,, \tag{7}$$

with loss function ℓ and where we made explicit the decomposition into feature extraction (Ψ_Θ) and linear classifier ($\Phi_{\mathbf{W}}$); to avoid overwhelming notations, we omit the bias terms in $\Phi_{\mathbf{W}}$ and use $\mathcal{G}_v^t$ to indicate $\widetilde{\mathcal{G}}_v^{(t,l)}$. Then, assuming that the loss is Lipschitz continuous with constant L_ℓ, and using the triangle inequality, we can isolate three components of forgetting risk:

$$\Delta\mathcal{R}_v \leq L_\ell \left(\left\| \Phi_{W_t}\left(\Psi_{\Theta_t}\left(\mathcal{G}_v^t\right)\right) - \Phi_{W_t}\left(\Psi_{\Theta_t}\left(\mathcal{G}_v^{t_0}\right)\right) \right\| \right. \qquad \textit{structural drift} \tag{8}$$

$$+ \left\| \Phi_{W_t}\left(\Psi_{\Theta_t}\left(\mathcal{G}_v^{t_0}\right)\right) - \Phi_{W_t}\left(\Psi_{\Theta_{t_0}}\left(\mathcal{G}_v^{t_0}\right)\right) \right\| \qquad \textit{backbone parameters drift} \tag{9}$$

$$\left. + \left\| \Phi_{W_t}\left(\Psi_{\Theta_{t_0}}\left(\mathcal{G}_v^{t_0}\right)\right) - \Phi_{W_{t_0}}\left(\Psi_{\Theta_{t_0}}\left(\mathcal{G}_v^{t_0}\right)\right) \right\| \right) . \qquad \textit{classifier parameters drift} \tag{10}$$

While the first term is inherent in the data-generating process, and therefore irreducible, a crucial advantage of using a fixed feature extractor is that it eliminates the second term, as $\Theta_t = \Theta_{t_0}$. Furthermore, if we assume the norm of the embeddings to be bounded by B_z, the third term is bounded by $B_z \left\| \boldsymbol{W}_t - \boldsymbol{W}_{t_0} \right\|$. For SLDA, the term is expected to decrease as the number of observed examples of a class increases, due to the update scheme through separate, class-specific cumulative means; this is in contrast with SGD-based training, where weight updates are less separated by class. Therefore, SLDA with fixed feature extraction provides high stability, with decreasing forgetting risk as the node stream progresses. Higher stability compared to SGD-trained methods can be empirically observed in Figures 2-3 of Appendix F.

This analysis illustrates the robustness against forgetting of untrained feature extraction with SLDA. However, stability is not sufficient for good CL performance: the model must also remain plastic enough to acquire new knowledge as the node stream evolves. For this, we can rely on the over-parameterized randomized feature extractors, which give us expressive and general topological embedding independently of task, allowing for the training of just a simple linear classifier on top (Scardapane & Wang, 2017).

4 Experimental setting

For our experiments we adopt the OCGL setting described in (Donghi et al., 2025). In particular, we follow the requirement of neighborhood sampling, and we consider small mini-batches of nodes

instead of individual ones. We compare the use of randomized feature extractors, UGCN and GRNF, with linear classifier, either SLDA or coupled with CL strategies, across multiple benchmarks.

Benchmarks We use the same six node-classification graph datasets of (Donghi et al., 2025): CoraFull (Bojchevski & Günnemann, 2018), Arxiv (Hu et al., 2021), Reddit (Hamilton et al., 2017), Amazon Computer (Shchur et al., 2019), Roman Empire (Platonov et al., 2022) and Elliptic (Weber et al., 2019). On all except Elliptic (as it only has two classes) we consider a class-incremental stream: nodes in the graph arrive one by one in blocks consisting of two classes (each segment can be identified with a task in the context of CL, even though the models in this setting are agnostic to task boundaries). On Elliptic and Arxiv we consider a time-incremental stream: since real node timestamps are available, we use them for a realistic node stream (dividing the stream into 10 blocks simply for evaluation). We split the nodes in each graph into 60% for training, 20% for validation and 20% for testing, and we use a transductive setting. We consider the same mini-batch sizes as (Donghi et al., 2025): 10 for the smaller CoraFull, Amazon Computer and Roman Empire, 50 for Arxiv, Reddit and Elliptic.

Metrics To evaluate model predictions in the considered setting, we use three metrics: *Average Performance (AP)*, *Average Forgetting (AF)* (Lopez-Paz & Ranzato, 2017), and *Average Anytime Performance (AAP)* (Caccia et al., 2021). The performance metric is accuracy for all datasets except for Elliptic, as it is highly unbalanced with only two classes, and therefore F1 score of the minority class is used. For anytime predictions, we obtain AAP by evaluating the model on the validation nodes after each training mini-batch. The metrics are described in detail in Appendix B.

Baselines In addition to the described SLDA, we couple the linear classifier with some popular CL strategies: we consider A-GEM (Chaudhry et al., 2018), ER (Chaudhry et al., 2019), EWC (Kirkpatrick et al., 2017), LwF (Li & Hoiem, 2018) and MAS (Aljundi et al., 2018). Furthermore, we consider the *bare* baseline which consists of simply finetuning the linear layer on the stream without any CL method. We also provide the *joint* baseline consisting of jointly training the linear layer offline on the embeddings from all the nodes in the entire final graph. We do not consider graph-specific methods, as once the features are extracted with the untrained backbone they are no longer relevant. However, we provide a comparison with recent state-of-the-art graph methods for OCGL (Donghi et al., 2025) in Appendix E and in summary in Table 3.

Implementation details Following (Donghi et al., 2025), we consider sparsified 2-hop node neighborhoods, sampling recursively 10 neighbors per layer for each node. For UGCN, we therefore use a 2-layer network with 1024 units per layer, resulting in a 2048-dimensional node embedding due to layer concatenation. For GRNF, we use 1024 features, which amount to a 2048-dimensional node embedding since we concatenate the equivariant and invariant components. In both cases, magnitude of weight initialization is regulated by a tunable gain hyperparameter. We use Adam optimizer (Kingma & Ba, 2017) without weight decay nor dropout, tuning the learning rate as an hyperparameter. Another hyperparameter is the number of passes on a mini-batch before passing to the next, as suggested by Aljundi et al. (Aljundi et al., 2019). Hyperparameters are tuned following the protocol outlined by Chaudhry et al. (Chaudhry et al., 2018): they are selected according to validation performance (AP) only on a small section of the data stream. All training and method specific hyperparameters are reported in Appendix D. All experiments are performed with 5 different random initializations, and results are reported as average and standard deviation over them.

5 RESULTS

The results of our experiments in the OCGL setting are reported in Table 1 for benchmarks with class-incremental node stream and Table 2 for those on which a time-incremental stream is defined. Additionally, for comparison with results obtainable by a model trained end-to-end, we report in Table 3 the best AP reported in (Donghi et al., 2025) (Arxiv is reported only with class-incremental stream as the time-incremental one is not considered in (Donghi et al., 2025)) by any OCGL strategy, in most cases SSM (Zhang et al., 2022b) and PDGNN (Zhang et al., 2024a), together with the respective joint upper bound, and we highlight the increase in performance using the proposed decoupled approach of randomized feature extraction with SLDA.

	Method	UGCN AP% ↑	UGCN AAP_{val}% ↑	UGCN AF% ↑	GRNF AP% ↑	GRNF AAP_{val}% ↑	GRNF AF% ↑
CoraFull	A-GEM	32.99±1.02	52.35±0.90	−50.53±1.19	33.33±0.42	48.04±0.89	−29.93±0.67
	ER	55.67±0.47	65.01±0.60	−24.90±0.71	52.53±0.60	59.79±0.56	−27.40±0.70
	EWC	34.46±1.28	52.28±0.35	−49.31±1.82	30.53±0.52	46.66±0.92	−50.44±0.75
	LwF	35.30±0.50	52.94±1.18	−46.41±0.74	30.70±0.85	45.87±0.98	−35.06±0.50
	MAS	34.44±1.30	52.59±0.36	−49.44±1.79	30.12±0.63	46.58±0.87	−51.28±0.73
	SLDA	64.03±0.50	74.65±0.44	−14.48±0.39	62.05±0.27	72.42±0.31	−16.87±0.50
	BARE	32.10±1.50	51.48±0.55	−51.85±1.62	28.18±0.71	44.82±1.11	−35.56±0.65
	Joint	66.10±0.28	-	−8.91±0.36	65.23±0.78	-	−8.05±0.54
A. Computer	A-GEM	60.35±4.86	58.12±3.18	−31.60±7.88	53.69±5.85	62.51±3.59	−39.10±8.55
	ER	80.71±2.83	83.30±0.51	−13.32±3.99	62.62±10.93	75.25±2.93	−20.04±11.26
	EWC	48.02±1.30	59.01±1.07	−30.29±1.55	30.80±5.50	49.10±1.05	−44.21±2.34
	LwF	47.31±11.89	60.84±3.24	−30.40±13.68	40.59±5.37	53.53±1.97	−50.96±2.41
	MAS	41.81±2.06	62.90±0.87	−45.72±3.10	43.14±3.89	57.19±2.01	−44.81±3.95
	SLDA	86.65±0.48	90.64±0.13	−7.72±0.47	84.32±0.42	89.95±0.23	−10.94±0.39
	BARE	43.98±3.39	50.21±1.26	−42.56±8.59	42.53±6.22	52.62±1.53	−49.47±2.49
	Joint	86.84±0.47	-	−7.30±0.47	87.43±0.34	-	−6.80±0.33
Arxiv	A-GEM	22.68±3.24	30.81±2.18	−59.57±1.86	14.32±1.82	32.39±1.80	−64.85±1.27
	ER	15.12±4.06	32.36±3.98	−49.55±4.96	14.63±1.78	31.84±2.05	−57.71±1.41
	EWC	17.26±2.12	27.71±1.02	−56.43±2.11	17.31±1.06	26.33±0.66	−67.39±0.52
	LwF	19.52±1.49	27.04±0.88	−56.52±0.92	21.72±0.98	29.84±0.40	−60.04±1.71
	MAS	18.77±2.44	28.92±1.27	−53.20±3.47	16.45±1.35	29.10±0.82	−67.50±1.20
	SLDA	55.71±0.08	64.55±0.03	−18.47±0.08	52.69±0.19	62.69±0.07	−27.67±0.16
	BARE	21.38±3.37	24.28±1.69	−41.42±2.17	14.13±1.14	24.67±0.33	−75.06±1.31
	Joint	59.06±0.15	-	−16.09±0.30	57.87±0.28	-	−16.72±0.19
Reddit	A-GEM	60.60±0.88	78.29±0.31	−36.77±0.89	46.51±0.91	61.06±0.20	−25.63±0.96
	ER	80.40±1.04	89.75±0.05	−16.60±1.11	83.46±0.43	89.22±0.12	−14.44±0.40
	EWC	40.91±1.22	61.13±1.07	−49.58±1.37	44.26±1.12	60.49±0.45	−29.80±1.18
	LwF	13.60±0.43	49.00±0.85	−84.37±0.43	39.09±0.77	58.00±1.07	−35.99±0.87
	MAS	11.93±0.66	50.54±0.48	−86.50±0.66	42.99±0.99	58.22±0.50	−32.62±0.97
	SLDA	89.31±0.03	95.17±0.03	−5.56±0.09	91.42±0.14	95.84±0.03	−3.81±0.15
	BARE	41.35±1.27	60.55±1.05	−48.97±1.26	42.67±0.75	57.80±0.36	−33.13±0.41
	Joint	88.07±0.20	-	−3.57±0.19	90.90±0.17	-	−2.77±0.17
Roman E.	A-GEM	17.18±0.46	43.99±0.20	−69.65±0.42	31.19±0.98	37.77±1.42	−23.28±4.54
	ER	24.80±1.21	46.35±0.50	−32.56±0.84	40.45±1.18	46.83±1.46	−15.68±2.20
	EWC	15.82±1.00	37.30±0.50	−43.62±0.95	21.54±0.85	40.22±0.91	−22.71±2.53
	LwF	21.51±0.65	45.51±0.37	−42.38±0.83	20.29±1.84	37.25±0.76	−46.63±2.02
	MAS	16.33±0.66	35.12±0.81	−28.71±1.77	22.47±0.58	40.52±0.42	−26.38±0.60
	SLDA	34.61±0.02	57.57±0.06	−34.38±0.20	52.71±0.09	72.07±0.03	−29.43±0.21
	BARE	7.09±0.03	34.74±0.69	−77.07±0.93	16.82±0.29	42.44±0.66	−68.56±1.54
	Joint	49.12±0.51	-	−5.22±0.48	71.50±0.10	-	−6.54±0.37

Table 1: Results for class-incremental node stream.

Class-incremental benchmarks We first observe from Table 1 that the upper bounds with randomized features are comparable to those obtained with a trained GCN model, as reported in Table 3. This proves that the extracted features are expressive enough for these tasks, confirming our approach's viability. On Roman Empire specifically, which is highly heterophilic, we see a much higher upper bound with GRNF: this is due to the implicit bias of the GCN design, which smooths node embeddings with neighborhood information, while GRNF are more expressive, as they are derived from a universal approximator of graph functions. We also note how AF for the joint baseline represents the increasing difficulty of the classification task as new classes are added.

From the results the superior performance of SLDA also emerges clearly, as it outperforms all considered CL methods on a linear layer trained with gradient descent, and in most cases with a wide margin. Additionally, the AP results of SLDA with both UGCN and GRNF closely approach their respective upper bounds. The results of the randomized feature extractors coupled with SLDA also significantly outperform the best results obtained with state-of-the-art replay methods tailored for graphs, as seen in Table 3, despite SLDA not using any memory buffer. The differences between UGCN and GRNF are in most case limited, especially for the SLDA classifier, with UGCN showing

		UGCN			GRNF		
	METHOD	AP% ↑	AAP_{val}% ↑	AF% ↑	AP% ↑	AAP_{val}% ↑	AF% ↑
ELLIPTIC	A-GEM	42.43±0.99	40.72±0.57	−13.51±1.54	53.65±0.73	50.10±0.33	−12.08±1.16
	ER	44.61±1.29	45.54±0.40	−9.17±1.82	57.30±1.69	53.36±1.29	−8.90±1.65
	EWC	36.62±1.23	34.64±0.20	−13.36±1.59	50.44±1.45	48.81±0.36	−13.77±1.16
	LWF	37.88±1.52	34.63±0.30	−11.43±1.39	55.16±1.26	50.15±0.59	−10.45±0.88
	MAS	36.80±0.84	34.47±0.21	−13.21±1.30	50.26±1.19	48.30±0.36	−14.29±0.97
	SLDA	55.49±0.64	54.13±0.28	−1.85±1.17	65.92±0.71	65.43±0.53	−3.21±1.90
	BARE	37.33±0.68	34.97±0.32	−12.21±1.01	50.14±1.44	48.81±0.45	−13.23±1.11
	JOINT	59.29±0.82	-	−4.94±0.62	70.23±0.76	-	−3.59±1.00
ARXIV	A-GEM	67.66±0.12	64.97±0.07	0.98±0.17	67.73±0.06	63.07±0.11	2.32±0.09
	ER	68.67±0.18	65.26±0.05	2.12±0.18	68.30±0.04	63.56±0.09	2.93±0.09
	EWC	67.65±0.07	64.96±0.03	0.94±0.08	67.65±0.08	63.93±0.12	1.48±0.10
	LWF	68.21±0.09	65.22±0.08	1.37±0.09	67.56±0.12	62.94±0.12	2.22±0.11
	MAS	67.65±0.07	64.96±0.03	0.94±0.08	67.64±0.09	63.93±0.12	1.47±0.10
	SLDA	66.69±0.07	61.97±0.04	2.29±0.17	65.09±0.21	60.14±0.10	2.42±0.20
	BARE	67.59±0.20	64.98±0.04	0.97±0.18	67.68±0.09	63.89±0.04	1.57±0.10
	JOINT	70.35±0.16	-	2.67±0.30	69.53±0.12	-	2.89±0.09

Table 2: Results for time-incremental node stream.

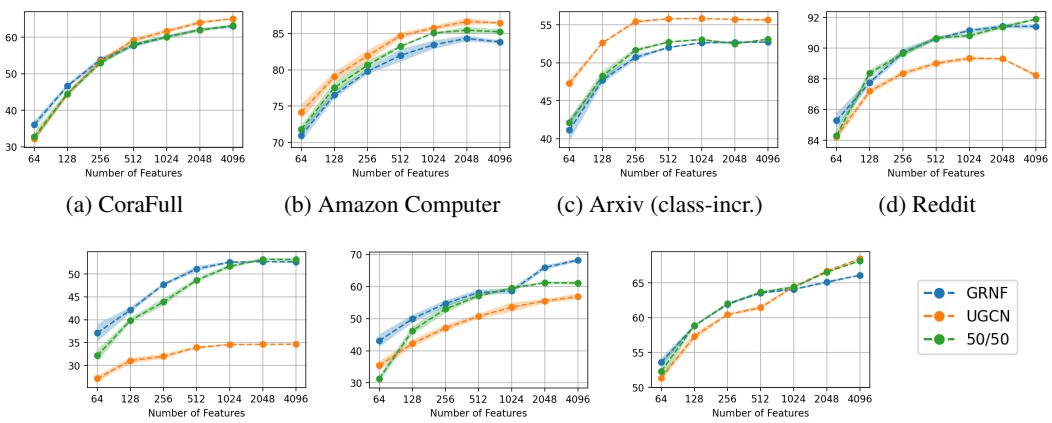
METHOD	CORAFULL	A. COMPUTER	ARXIV (CL.-INCR.)	REDDIT	ROMAN E.	ELLIPTIC
BEST OCGL	40.45±0.77	70.45±3.66	35.86±1.20	58.08±8.04	14.20±0.87	51.13±1.74
JOINT	67.55±0.05	83.07±1.30	58.58±0.28	90.02±0.12	39.47±0.33	71.97±0.83
UGCN+SLDA	+23.6	+16.2	+19.9	+31.2	+20.4	+4.4
GRNF+SLDA	+21.6	+13.9	+16.8	+33.3	+38.5	+14.8

Table 3: Top rows: best AP results from Table 5 of Appendix E for CL strategies with trained GCN and joint offline training upper bound. Bottom rows: increase in AP of the proposed approach (Tables 1-2) compared to the best performing method with trained GCN. All the increases in performance show statistical significance with p-value ≤ 0.005 with a one-sided Mann–Whitney U test.

a slight advantage on CoraFull, Amazon Computer and Arxiv benchmarks, while GRNF appears superior on Reddit and more significantly on Roman Empire, due to the heterophily of the graph as discussed above.

In general, compared to the full results in Appendix E, also most other CL strategies in most benchmarks report performance improvements compared to the use of a trained GCN, confirming the benefits of keeping a frozen feature extractor immune to forgetting. ER specifically is significantly better when coupled with randomized features compared to a trained GCN, in some cases approaching SLDA, due to the fact that in this setting the memory buffer is much more informative, as stored examples contain neighborhood information, albeit subject to structural shift.

Time-incremental benchmarks The time-incremental setting naturally possesses more docile shifts in distribution, compared to the abrupt class changes of the class-incremental benchmark. Nonetheless, on Elliptic CL methods are still beneficial, with SLDA still outperforming all baselines, also compared to the trained GCN state-of-the-art (Table 3). Importantly, our results on Elliptic, which is a dataset of real Bitcoin transactions, highlight the feasibility and benefits of the proposed approach on realistic data streams, beyond the academic class-incremental setting. On the other hand, for the Arxiv time-incremental benchmark, we see little difference in the results of the various strategies, with SLDA no longer over-performing. In fact, we even see positive AF values for all methods, indicating that the classification becomes easier, rather than harder, as the node stream goes on. This is because Arxiv does not present a significant drift in class distribution throughout time. Therefore, a CL learning approach here is less meaningful, as the bare baseline is already close to the upper bound. Nevertheless, ER proves beneficial, and these high results are further proof that the feature extractors are robust to structural shifts that still remain.



(a) CoraFull (b) Amazon Computer (c) Arxiv (class-incr.) (d) Reddit

(e) Roman Empire (f) Elliptic (g) Arxiv (time-incr.)

Figure 1: Comparison of AP of SLDA with different number of features extracted with GRNF, UGCN, and a 50/50 mix of the two. The shaded area covers one standard deviation.

Impact of number of features Given the overwhelming over-performance of randomized feature extraction with SLDA, we investigate the impact of the number of extracted features on model performance. In Figure 1, we see that, with a lower number of features AP decreases as well, even though on most benchmarks even as few as 64 randomized features are sufficient to obtain results on par with the state-of-the-art CL methods on trained GCN of Table 3. Also, for many benchmarks performance seems to not have reached saturation even at 4096 features, indicating that further gains could be achieved with a larger feature extractor.

Finally, since UGCN and GRNF appear to have different strengths over the multiple benchmarks, we consider a hybrid feature extractor that extracts half of the features with UGCN, and half with GRNF. This mixed feature extractor shows a generally more robust performance over the benchmarks, with performance that is never lower than the individual two, except for one single point. Generally the hybrid extractor also does not improve over the best single one, indicating that the two types of feature do not benefit from integration. However, this strategy could be used to obtain a reliable feature extractor without the need to evaluate the two strategies.

6 Conclusion

This work introduces a simple, yet surprisingly effective approach for Online Continual Graph Learning, addressing forgetting by decoupling representation learning from classification. We use randomized, fixed node feature extractors that encode neighborhood information, coupled with a lightweight linear classifier trained incrementally on the node stream. By leveraging two types of untrained feature extractors – UGCN and GRNF – the proposed method provides robust and expressive node embeddings, resistant to catastrophic forgetting. Extensive experiments in the OCGL setting demonstrate that when paired with SLDA, this approach significantly outperforms other Continual Learning strategies, including state-of-the-art replay-based methods tailored for graph data. The method achieves performance close to joint offline training across various benchmarks. Beyond strong performance, its efficient streaming updates and no reliance on memory buffers make it a scalable and practical approach to deal with real-time classification on graph node streams.

Limitations and future directions While we provide motivation and empirical evidence of the improved resistance to forgetting and higher prediction accuracy of our approach, a thorough theoretical analysis is yet to be developed. Secondly, we highlight how our results are specific for the challenging OCGL scenario, while for different, offline settings other methods can perform more favorably. Finally, we focus on node-level classification, as graph-level is less interesting for OCGL, and leave regression and edge-level tasks as future research.

REFERENCES

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)*, pp. 139–154, 2018.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online Continual Learning with Maximal Interfered Retrieval. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Aleksandar Bojchevski and Stephan Günnemann. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *International Conference on Learning Representations*, February 2018.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New Insights on Reducing Abrupt Representation Change in Online Continual Learning. In *International Conference on Learning Representations*, October 2021.

Luciano Caroprese, Francesco Sergio Pisani, Bruno Miguel Veloso, Matthias Konig, Giuseppe Manco, Holger Hoos, and Joao Gama. Modelling concept drift in dynamic data streams for recommender systems. *ACM Trans. Recomm. Syst.*, 3(2), March 2025. doi: 10.1145/3707693. URL `https://doi.org/10.1145/3707693`.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient Lifelong Learning with A-GEM. In *ICLR*, September 2018.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On Tiny Episodic Memories in Continual Learning, 2019. arXiv:1902.10486.

Xu Chen, Junshan Wang, and Kunqing Xie. Trafficstream: A streaming traffic flow forecasting framework based on graph neural networks and continual learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 3620–3626. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/498.

Yuanning Cui, Yuxin Wang, Zequn Sun, Wenqiang Liu, Yiqiao Jiang, Kexin Han, and Wei Hu. Lifelong Embedding Learning and Transfer for Growing Knowledge Graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4):4217–4224, June 2023. doi: 10.1609/aaai.v37i4.25539.

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, July 2022. doi: 10.1109/TPAMI.2021.3057446.

Meng Ding, Kaiyi Ji, Di Wang, and Jinhui Xu. Understanding forgetting in continual learning with linear regression. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 10978–11001, 2024.

Giovanni Donghi, Luca Pasa, Luca Oneto, Claudio Gallicchio, Alessio Micheli, Davide Anguita, Alessandro Sperduti, and Nicolò Navarin. Investigating over-parameterized randomized graph networks. *Neurocomputing*, 606, November 2024. doi: 10.1016/j.neucom.2024.128281.

Giovanni Donghi, Luca Pasa, Daniele Zambon, Cesare Alippi, and Nicolò Navarin. Online Continual Graph Learning, 2025. arXiv:2508.03283.

Itay Evron, Edward Moroshko, Rachel Ward, Nathan Srebro, and Daniel Soudry. How catastrophic can catastrophic forgetting be in linear regression? In *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178, pp. 4028–4079. PMLR, Jul 2022.

Itay Evron, Edward Moroshko, Gon Buzaglo, Maroun Khriesh, Badea Marjieh, Nathan Srebro, and Daniel Soudry. Continual learning in linear classification on separable data. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 9440–9484, 2023.

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, February 2023. doi: 10.1109/MCI.2022.3222049.

Claudio Gallicchio and Alessio Micheli. Graph Echo State Networks. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, July 2010. doi: 10.1109/IJCNN.2010.5596796.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1263–1272. PMLR, July 2017. ISSN: 2640-3498.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Tyler L. Hayes and Christopher Kanan. Lifelong Machine Learning with Deep Streaming Linear Discriminant Analysis. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 887–896, Seattle, WA, USA, June 2020. doi: 10.1109/CVPRW50498.2020.00118.

Thanh Duc Hoang, Do Viet Tung, Duy-Hung Nguyen, Bao-Sinh Nguyen, Huy Hoang Nguyen, and Hung Le. Universal Graph Continual Learning. *Transactions on Machine Learning Research*, 2023.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs, February 2021. arXiv:2005.00687.

C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, X. Wu, and M. Li. Are Graph Convolutional Networks With Random Weights Feasible? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2751–2768, 2023.

Gao Huang, Guang-Bin Huang, Shiji Song, and Keyou You. Trends in extreme learning machines: A review. *Neural Networks*, 61:32–48, January 2015. doi: 10.1016/j.neunet.2014.10.001.

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, December 2006. doi: 10.1016/j.neucom.2005.12.126.

Herbert Jaeger and Harald Haas. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science*, 304(5667):78–80, 2004.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. arXiv:1412.6980.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 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, March 2017. doi: 10.1073/pnas.1611835114.

Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online Continual Learning on Class Incremental Blurry Task Configuration with Anytime Inference. In *International Conference on Learning Representations*, October 2021.

Hugo Le Baher, Jérôme Azé, Sandra Bringay, Pascal Poncelet, Nancy Rodriguez, and Caroline Dunoyer. Patient Electronic Health Record as Temporal Graphs for Health Monitoring. *Studies in Health Technology and Informatics*, 302:561–565, May 2023. ISSN 1879-8365. doi: 10.3233/SHTI230205.

Depeng Li and Zhigang Zeng. CRNet: A Fast Continual Learning Framework With Random Theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10731–10744, September 2023. doi: 10.1109/TPAMI.2023.3262853.

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

Lieqing Lin, Qi Zhong, Jiasheng Qiu, and Zhenyu Liang. E-GRACL: an IoT intrusion detection system based on graph neural networks. *The Journal of Supercomputing*, 81(1):42, October 2024. ISSN 1573-0484. doi: 10.1007/s11227-024-06471-5. URL https://doi.org/10.1007/s11227-024-06471-5.

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.

Xueyi Liu, Chuanhou Gao, and Ping Li. A comparative analysis of support vector machines and extreme learning machines. *Neural Networks*, 33:58–66, 2012. doi: https://doi.org/10.1016/j.neunet.2012.04.002.

Yilun Liu, Ruihong Qiu, and Zi Huang. CaT: Balanced Continual Graph Learning with Graph Condensation. In *2023 IEEE International Conference on Data Mining (ICDM)*, pp. 1157–1162, 2023. doi: 10.1109/ICDM58522.2023.00141.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, January 2022. doi: 10.1016/j.neucom.2021.10.021.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 4363–4371, Jun 2019.

Mark D. McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. RanPAC: Random Projections and Pre-trained Models for Continual Learning. *Advances in Neural Information Processing Systems*, 36:12022–12053, December 2023.

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An Empirical Investigation of the Role of Pre-training in Lifelong Learning. *Journal of Machine Learning Research*, 24(214):1–50, 2023.

Alessio Micheli. Neural Network for Graphs: A Contextual Constructive Approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, March 2009. doi: 10.1109/TNN.2008.2010350.

Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Huiyi Hu, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Wide Neural Networks Forget Less Catastrophically. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 15699–15717, June 2022.

N. Navarin, L. Pasa, C. Gallicchio, and A. Sperduti. An untrained neural model for fast and accurate graph classification. In *International Conference on Artificial Neural Networks*, 2023a.

Nicolò Navarin, Luca Pasa, Luca Oneto, and Alessandro Sperduti. An Empirical Study of Over-Parameterized Neural Models based on Graph Random Features. In *ESANN 2023 proceesdings*, pp. 17–22, 2023b. doi: 10.14428/esann/2023.ES2023-145.

Y.-H. Pao and Y. Takefuji. Functional-link net computing: theory, system architecture, and functionalities. *Computer*, 25(5):76–79, 1992. doi: 10.1109/2.144401.

Yoh-Han Pao, Gwang-Hoon Park, and Dejan J. Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994. ISSN 0925-2312. doi: https://doi.org/10.1016/0925-2312(94)90053-1. Backpropagation, Part IV.

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, May 2019. doi: 10.1016/j.neunet.2019.01.012.

Luca Pasa, Nicolò Navarin, and Alessandro Sperduti. Multiresolution reservoir graph neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2642–2653, 2022. doi: 10.1109/TNNLS.2021.3090503.

Francesco Pelosin. Simpler is Better: off-the-shelf Continual Learning Through Pretrained Backbones, May 2022. arXiv:2205.01586.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2022.

Ameya Prabhu, Shiven Sinha, Ponnurangam Kumaraguru, Philip Torr, Ozan Sener, and Puneet Dokania. RanDumb: Random Representations Outperform Online Continually Learned Representations. *Advances in Neural Information Processing Systems*, 37:37988–38006, January 2025.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis (eds.), *Advances in Neural Information Processing Systems*, volume 20, 2007.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (eds.), *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper_files/paper/2008/file/0efe32849d230d7f53049ddc4a4b0c60-Paper.pdf.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, Honolulu, HI, July 2017. IEEE. doi: 10.1109/CVPR.2017.587.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience Replay for Continual Learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, volume 30, 2017a.

Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017b. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/61b1fb3f59e28c67f3925f3c79be81a1-Paper.pdf.

Simone Scardapane and Dianhui Wang. Randomness in neural networks: an overview. *WIREs Data Mining and Knowledge Discovery*, 7(2), 2017. doi: 10.1002/widm.1200.

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, January 2009. doi: 10.1109/TNN.2008.2005605.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of Graph Neural Network Evaluation, June 2019. arXiv:1811.05868.

Albin Soutif–Cormerais, Antonio Carta, Andrea Cossu, Julio Hurtado, Vincenzo Lomonaco, Joost Van De Weijer, and Hamed Hemati. A Comprehensive Empirical Evaluation on Online Continual Learning. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 3510–3520, October 2023. doi: 10.1109/ICCVW60793.2023.00378.

Junwei Su, Difan Zou, Zijun Zhang, and Chuan Wu. Towards robust graph incremental learning on evolving graphs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 32728–32748, 2023.

Junwei Su, Difan Zou, and Chuan Wu. On the Limitation and Experience Replay for GNNs in Continual Learning, July 2024. arXiv:2302.03534.

Li Sun, Junda Ye, Hao Peng, Feiyang Wang, and Philip S. Yu. Self-Supervised Continual Graph Learning in Adaptive Riemannian Spaces. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4):4633–4642, June 2023. doi: 10.1609/aaai.v37i4.25586.

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

Ruohan Wang, Marco Ciccone, Giulia Luise, Andrew Yapp, Massimiliano Pontil, and Carlo Ciliberto. Schedule-Robust Online Continual Learning, October 2022. arXiv:2210.05561 [cs].

Zehong Wang, Zheyuan Liu, Tianyi Ma, Jiazheng Li, Zheyuan Zhang, Xingbo Fu, Yiyang Li, Zhengqing Yuan, Wei Song, Yijun Ma, Qingkai Zeng, Xiusi Chen, Jianan Zhao, Jundong Li, Meng Jiang, Pietro Lio, Nitesh Chawla, Chuxu Zhang, and Yanfang Ye. Graph Foundation Models: A Comprehensive Survey, May 2025. arXiv:2505.15116 [cs].

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. arXiv:1908.02591.

Qiao Yuan, Sheng-Uei Guan, Pin Ni, Tianlun Luo, Ka Lok Man, Prudence Wong, and Victor Chang. Continual Graph Learning: A Survey, 2023. arXiv:2301.12230.

Daniele Zambon, Cesare Alippi, and Lorenzo Livi. Graph Random Neural Features for Distance-Preserving Graph Representations. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 10968–10977, November 2020.

Xikun Zhang, Dongjin Song, and Dacheng Tao. CGLB: Benchmark Tasks for Continual Graph Learning. *Advances in Neural Information Processing Systems*, 35:13006–13021, December 2022a.

Xikun Zhang, Dongjin Song, and Dacheng Tao. Sparsified Subgraph Memory for Continual Graph Representation Learning. In *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 1335–1340, 2022b.

Xikun Zhang, Dongjin Song, Yixin Chen, and Dacheng Tao. Topology-aware Embedding Memory for Continual Learning on Expanding Networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4326–4337, 2024a. doi: 10.1145/3637528.3671732.

Xikun Zhang, Dongjin Song, and Dacheng Tao. Continual Learning on Graphs: Challenges, Solutions, and Opportunities, February 2024b. arXiv:2402.11565.

Zijia Zhang, Yaoming Cai, Wenyin Gong, Xiaobo Liu, and Zhihua Cai. Graph Convolutional Extreme Learning Machine. In *2020 International Joint Conference on Neural Networks (IJCNN)*, July 2020. doi: 10.1109/IJCNN48605.2020.9206649.

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.

A BENCHMARKS

The benchmarks for out experiments are obtained from six node-level classification graph datasets. The CoraFull dataset Bojchevski & Günnemann (2018) is a citation network where nodes represent research papers and edges denote citation links between them, with labels corresponding to paper topics. Amazon Computer Shchur et al. (2019) is a co-purchase graph, with nodes representing products and edges indicating frequent co-purchases in the computer category on Amazon. Arxiv Hu et al. (2021) is a larger citation network based on arXiv submissions in the Computer Science domain. The Reddit dataset Hamilton et al. (2017) comprises posts from various Reddit communities, where each node represents a post, and edges connect posts that were commented on by the same user, capturing user interaction patterns. Roman Empire Platonov et al. (2022) is an heterophilous dataset constructed from the corresponding Wikipedia page, where nodes are words linked through syntactic relationships or adjacency in the text. Lastly, the Elliptic dataset Weber et al. (2019) is a graph of Bitcoin transactions, with edges representing the flow of funds. Only a subset of nodes are labeled as either *licit* (42,019 nodes) or *illicit* (4,545 nodes) transactions. Summary statistics for the six datasets are provided in Table 4.

Dataset	CoraFull	Amazon Computer	Arxiv	Reddit	Roman Empire	Elliptic
# nodes	19,793	13,752	169,343	227,853	22,662	203,769
# edges	130,622	491,722	1,166,243	114,615,892	32,927	234,355
# classes	70	10	40	40	18	2

Table 4: Dataset statistics.

B METRICS

Due to the way the node stream is built, with a definition of task boundaries, we can make use of two commonly adopted continual learning (CL) metrics: *Average Performance (AP)* and *Average Forgetting (AF)* Lopez-Paz & Ranzato (2017). These metrics are both derived from the more general performance matrix $\boldsymbol{M} \in \mathbb{R}^{T \times T}$, where T denotes the total number of tasks, and each element $M_{i,j}$ corresponds to the test performance on task j after training on task i.

The *Average Performance* is given by $\text{AP} = \frac{1}{T} \sum_{i=1}^{T} M_{T,i}$, representing the model's performance on all tasks after completing the full training stream. The *Average Forgetting* is computed as $\text{AF} = \frac{1}{T-1} \sum_{i=1}^{T-1} M_{T,i} - M_{i,i}$, and quantifies how much the model's performance on each task has deteriorated between its initial learning and the end of training. For evaluating performance, we rely on classification accuracy across all datasets, except for Elliptic, which is significantly imbalanced. For this dataset, we instead report the F1 score specific to the *illicit* class.

To track model behavior throughout the node stream, we also employ anytime evaluation: the model is evaluated on validation nodes after each mini-batch update Koh et al. (2021). This provides a fine-grained view of model performance over time, revealing its adaptability to distributional shifts. We quantify this using the *Average Anytime Performance (AAP)* metric Caccia et al. (2021), a generalization of average incremental accuracy for the online scenario. Letting APt denote the average accuracy after processing the t-th mini-batch, and n be the total number of mini-batches, AAP is defined as $\text{AAP} = \frac{1}{n} \sum t = 1^{n} \text{AP}_t$. This metric can be interpreted as the area under the accuracy curve across the training process Koh et al. (2021).

C ONLINE FEATURE CENTERING TRICK

As in our experiments we consider also using a standard linear layer trained continually with gradient descent instead of SLDA, in this case it is beneficial to have featured centered in the origin. This is especially true due to the online setting, as having centered features can make the learning of bias parameters for newly observed classes faster, since we initialize the weights symmetrically around zero. Therefore, we adopt an online centering procedure, which allows us to keep the features centered at any point during the stream. Specifically, we maintain an updated global cumulative

mean m of node embeddings, so that

$$m^{(t)} = \frac{(t-1)m^{(t-1)} + \boldsymbol{z}^{(t)}}{t}\,, \tag{11}$$

where for ease of notation we use $\boldsymbol{z}^{(t)} = \boldsymbol{z}^{(t)}_{v_t}$. With this, we then feed the centered embeddings $\boldsymbol{z}^{(t)}_v - m^{(t)}$ in equation (3) of the paper. To ensure consistency of model predictions with online feature centering under updates of the embedding mean as in equation 11, we want to correct the bias term $\boldsymbol{b} \to \boldsymbol{b}'$ to ensure that for any $\boldsymbol{z} \in \mathbb{R}^d$ predictions don't change, that is

$$\boldsymbol{W}(\boldsymbol{z} - m^{(t)}) + \boldsymbol{b}' = \boldsymbol{W}(\boldsymbol{z} - m^{(t-1)}) + \boldsymbol{b}\,. \tag{12}$$

Simplifying and using the definition of the update of $m^{(t)}$ in equation 11, we obtain:

$$-\boldsymbol{W}m^{(t)} + \boldsymbol{b}' = -\boldsymbol{W}m^{(t-1)} + \boldsymbol{b}\,, \tag{13}$$

$$-\frac{(t-1)}{t}\boldsymbol{W}m^{(t-1)} - \frac{1}{t}\boldsymbol{W}\boldsymbol{z}^{(t)} + \boldsymbol{b}' = -\boldsymbol{W}m^{(t-1)} + \boldsymbol{b}\,, \tag{14}$$

$$\boldsymbol{b}' = -\frac{1}{t}\boldsymbol{W}m^{(t-1)} + \frac{1}{t}\boldsymbol{W}\boldsymbol{z}^{(t)} + \boldsymbol{b}\,, \tag{15}$$

$$\boldsymbol{b}' = \frac{1}{t}\boldsymbol{W}(\boldsymbol{z}^{(t)} - m^{(t-1)}) + \boldsymbol{b}\,. \tag{16}$$

This is the bias update formula, and a similar one can be derived for mini-batch updates. We highlight how this bias correction is performed only for seen classes, as we grow the classification head when new classes are encountered. Also, this online centering procedure is not performed with SLDA, since it is a prototype-based classifier and is thus indifferent to feature centering. Empirically, we have observed that this trick greatly improves performance when the node embeddings are not centered, which is often the case with GRNF.

D HYPERPARAMETERS

We perform model selection using a limited section of the node stream, approximately 20% of the tasks. Therefore, for class-incremental stream we validate over 7 out 35 tasks for CoraFull, 2 out of 5 for Amazon Computer (as considering 20% of the tasks would mean using only 1, thus without any CL aspect), 4 out of 20 for Arxiv and Reddit, and 2 out of 9 for Roman Empire. For the time-incremental stream, we validate over the first 20% of the nodes (i.e., 2 out of 10 tasks). The hyperparameters are selected by running a standard grid search, over the search space that we illustrate here. For all experiments and both backbones, we consider the gain hyperparameter for weight initialization in $\{0.1, 1, 10\}$. For all methods, except SLDA, we select the learning rate from the set $\{0.01, 0.001, 0.0001, 0.00001\}$, and the number of passes on each batch before going to the next one between 1 and 5. For ER and A-GEM, we consider the proportion of memories to use with respect to each training batch in $\{1, 2, 3\}$. Additionally, we set the same memory buffer size as Donghi et al. (2025): 4000 for CoraFull, Amazon Computer, Roman Empire and Elliptic, 16000 for Arxiv and Reddit. The regularization hyperparameter for EWC and MAS is selected in $\{10^0, 10^2, 10^4, 10^6, 10^8, 10^{10}\}$. For LwF, we consider lambda_dist in $\{1, 10\}$, T in $\{0.2, 2\}$ and the number of mini-batches after which to update the teacher model in $\{10, 100\}$.

We highlight how the only hyperparameter considered for SLDA is the gain of the backbone, making it even easier to use than other methods, avoiding expensive hyperparameter search.

E RESULTS WITH TRAINED GCN

In Table 5 we report the AP of CL strategies when used with a trained GCN in the OCGL setting Donghi et al. (2025). These results are obtained in the same configurations used for the experiments in the main paper, only with a trained 2-layer GCN with 256 hidden units instead of an untrained feature extractor. The results on Arxiv are provided only for the class-incremental stream. For the full results with the other metrics (AAP and AF), we point the reader to the original paper introducing the setting.

METHOD	CORAFULL	A. COMPUTER	ARXIV	REDDIT	ROMAN E.	ELLIPTIC
EWC	28.10±2.76	14.86±6.00	4.81±0.08	4.33±2.77	8.85±0.05	51.08±1.10
LWF	15.74±1.56	19.33±0.14	4.79±0.08	13.13±1.92	8.81±0.01	50.79±1.36
MAS	8.40±0.62	12.68±8.28	3.35±0.99	10.21±1.03	11.02±1.36	51.08±1.10
TWP	13.98±1.66	19.80±3.41	4.74±0.05	12.79±2.51	8.99±0.06	**51.13±1.74**
ER	20.65±2.33	38.53±2.93	23.43±1.65	22.34±2.46	10.43±0.20	43.94±0.52
A-GEM	**40.45±0.77**	38.49±2.80	17.16±1.45	**58.08±8.04**	9.07±0.15	47.06±1.18
PDGNN	38.48±1.15	68.91±0.33	**35.86±1.20**	53.98±0.44	**14.20±0.87**	49.91±1.44
SSM-ER	23.23±2.69	**70.45±3.66**	31.28±1.91	50.48±1.69	11.71±0.51	24.45±2.97
SSM-A-GEM	36.63±4.63	50.01±9.12	13.12±2.46	22.54±4.51	9.00±0.09	40.48±0.82
BARE	12.59±0.82	20.51±4.26	4.74±0.08	12.59±2.59	8.78±0.10	51.28±2.37
JOINT	67.55±0.05	83.07±1.30	58.58±0.28	90.02±0.12	39.47±0.33	71.97±0.83

Table 5: AP results in the OCGL setting from (Donghi et al., 2025) of CL strategies with trained GCN and joint offline training upper bound. In particular, TWP (Liu et al., 2021), PDGNN (Zhang et al., 2024a), SSM-ER and SSM-A-GEM (Zhang et al., 2022b) are graph-specific CL baselines. Best performing method for each dataset is highlighted in bold. For Arxiv class-incremental stream is considered.

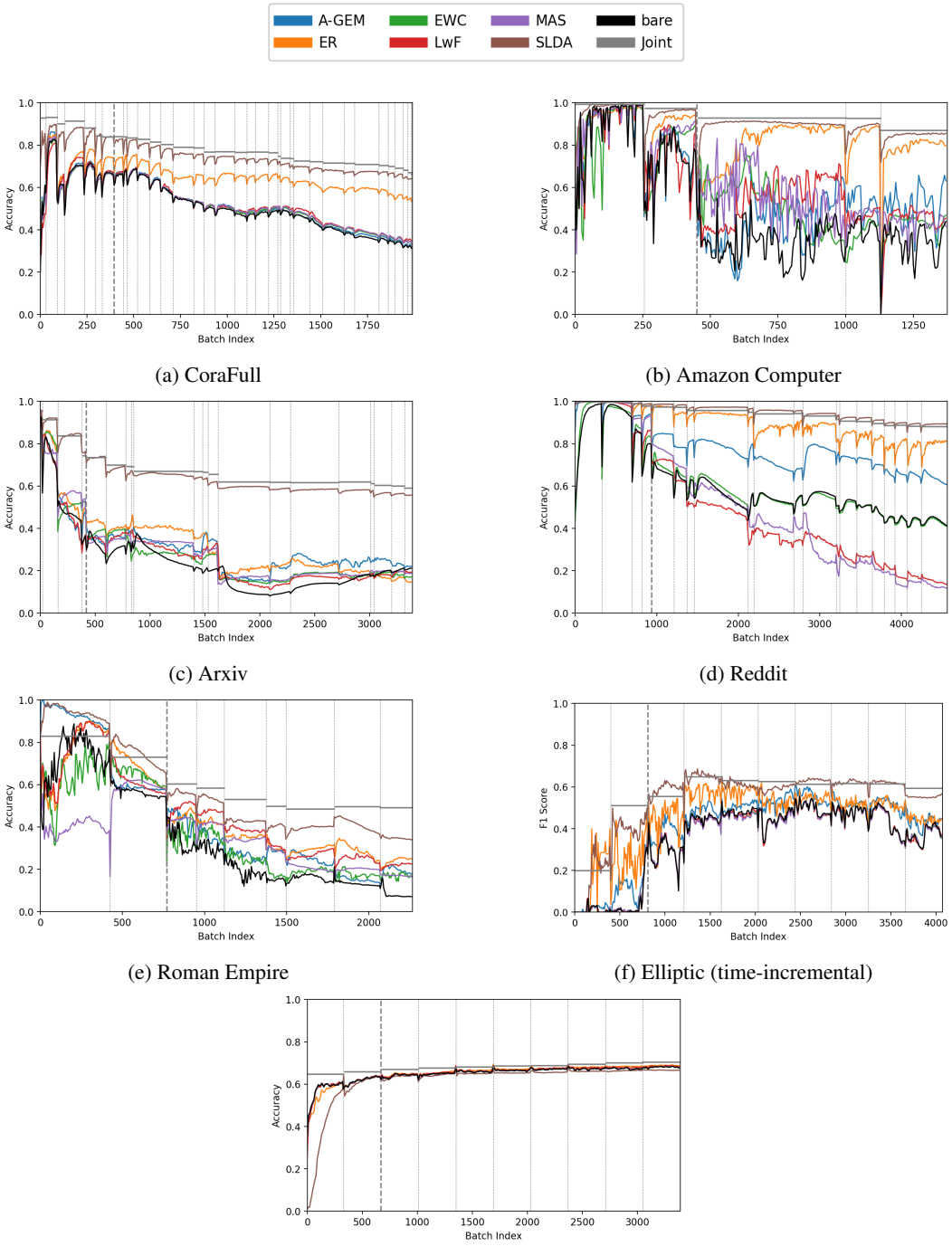
F PERFORMANCE PLOTS

We report here plots that show model performance along the node stream, to provide a more detailed understanding of the dynamics of training and forgetting. In Figures 2 and 3 we plot for each benchmark a comparison of the performance using the considered methods, for UGCN and GRNF backbones respectively. We highlight the task boundaries with dotted vertical lines, with the thicker dashed one indicating the threshold at which hyperparameter selection is performed. The upper bound of joint training on data up to the present task is represented as an horizontal line over the batches of each task.

In Figures 4-17, we instead illustrate a more detailed breakdown of model performance: for each benchmark, backbone and considered method, we plot the results of anytime evaluation broken down on individual tasks, allowing a better understanding of when and where forgetting occours.

G LLM USAGE

Large Language Models were used for the purpose of text editing.



(a) CoraFull

(b) Amazon Computer

(c) Arxiv

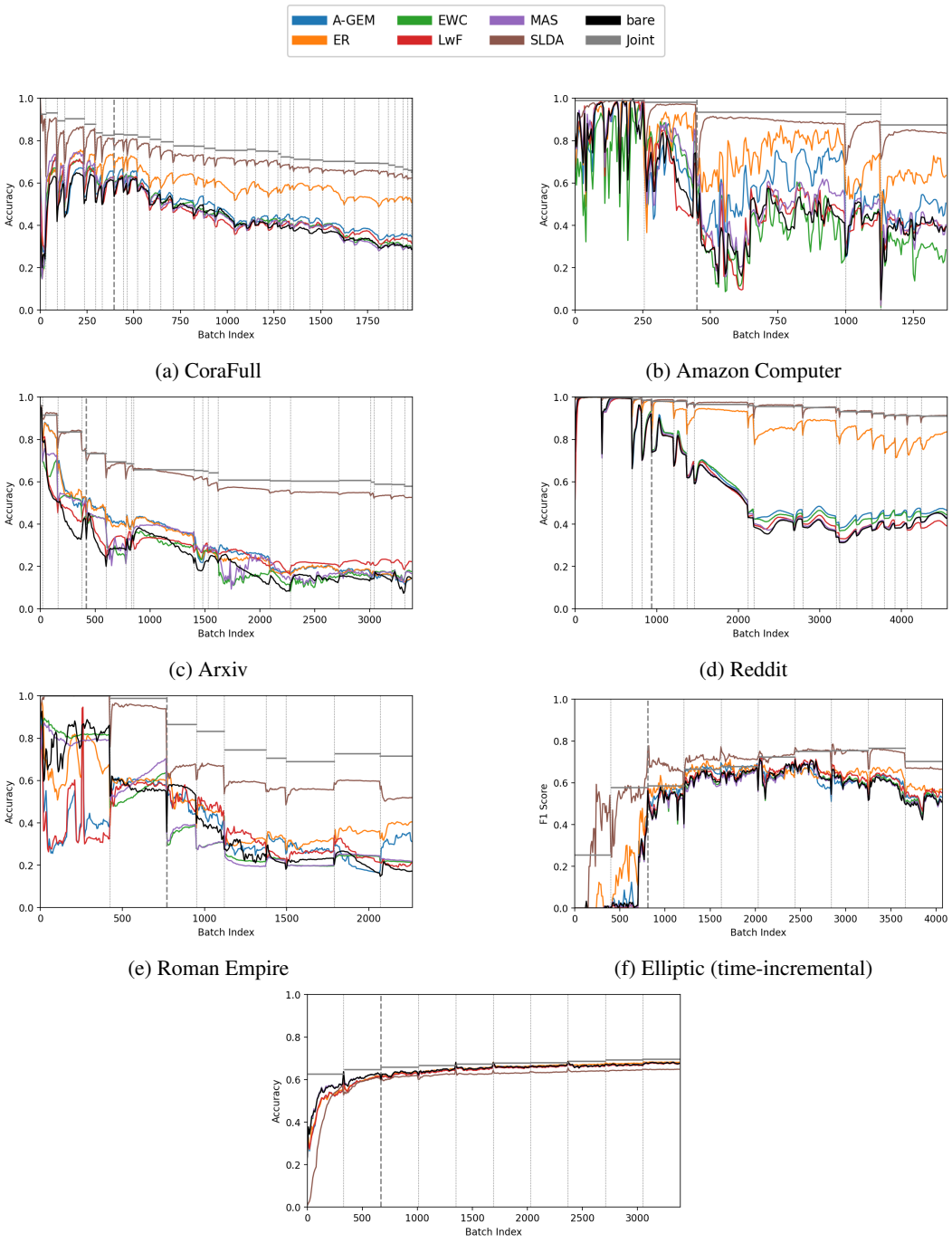
(d) Reddit

(e) Roman Empire

(f) Elliptic (time-incremental)

(g) Arxiv (time-incremental)

Figure 2: Anytime evaluation performance for the different datasets with UGCN Backbone. We highlight with vertical lines the task boundaries and the hyperparameter selection threshold.



(a) CoraFull

(b) Amazon Computer

(c) Arxiv

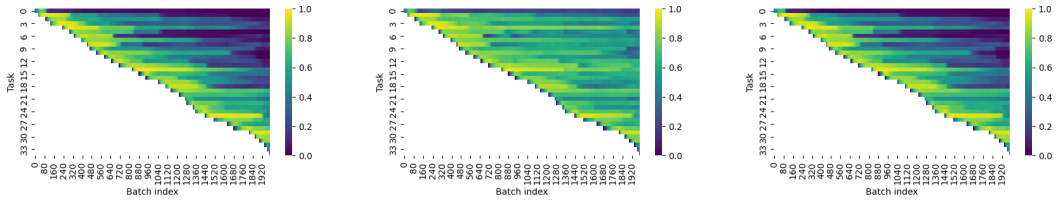
(d) Reddit

(e) Roman Empire

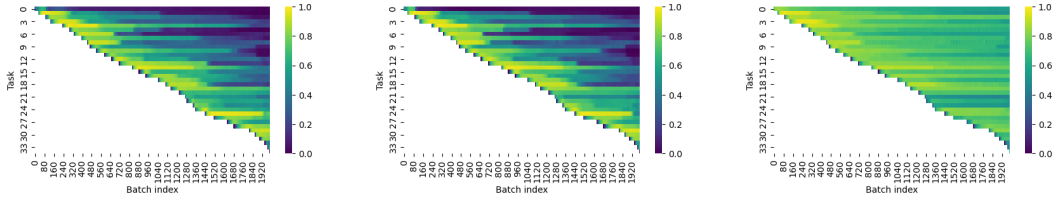
(f) Elliptic (time-incremental)

(g) Arxiv (time-incremental)

Figure 3: Anytime evaluation performance for the different datasets with GRNF backbone. We highlight with vertical lines the task boundaries and the hyperparameter selection threshold.

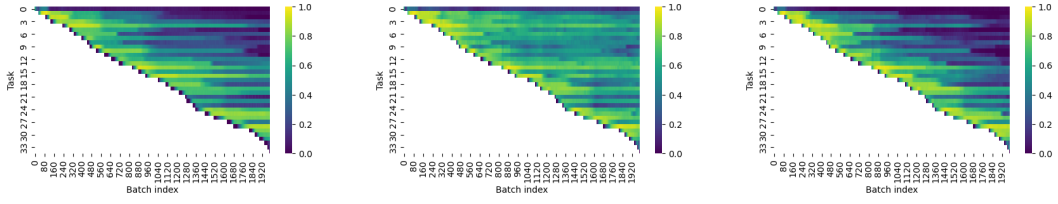


(a) A-GEM (b) ER (c) EWC

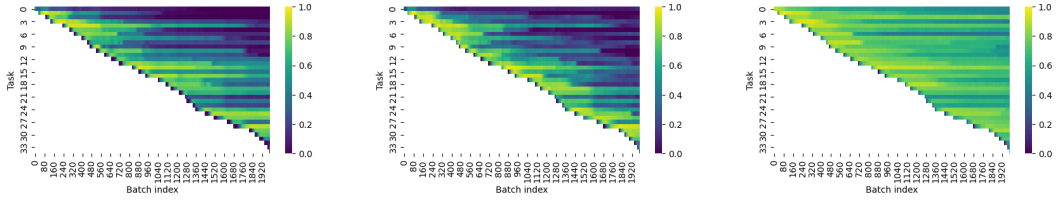


(d) LwF (e) MAS (f) SLDA

Figure 4: Anytime evaluation by task for the CoraFull dataset with UGCN backbone.

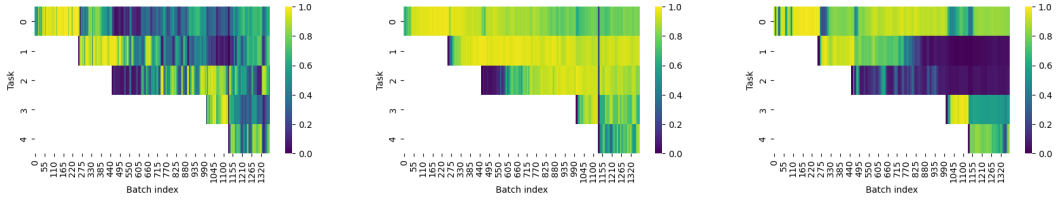


(a) A-GEM (b) ER (c) EWC

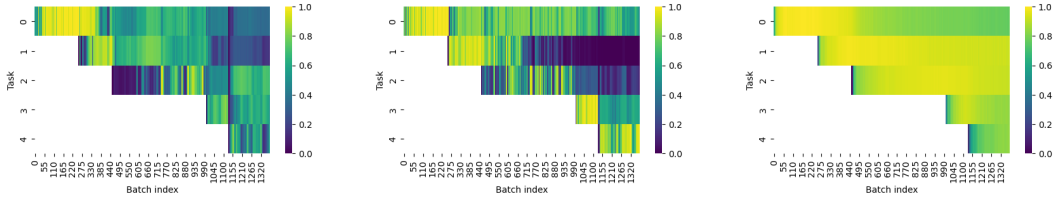


(d) LwF (e) MAS (f) SLDA

Figure 5: Anytime evaluation by task for the CoraFull dataset with GRNF backbone.

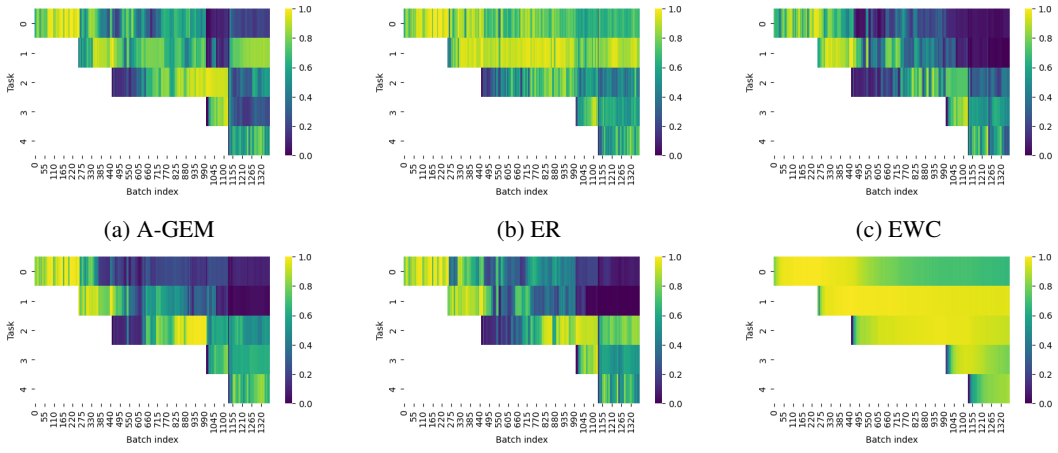


(a) A-GEM (b) ER (c) EWC



(d) LwF (e) MAS (f) SLDA

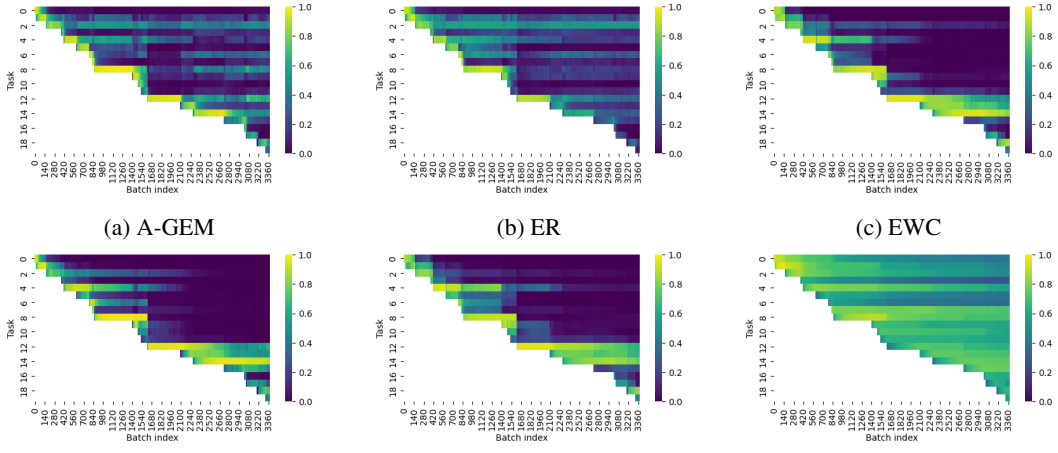
Figure 6: Anytime evaluation by task for the Amazon Computer dataset with UGCN backbone.



(a) A-GEM (b) ER (c) EWC

(d) LwF (e) MAS (f) SLDA

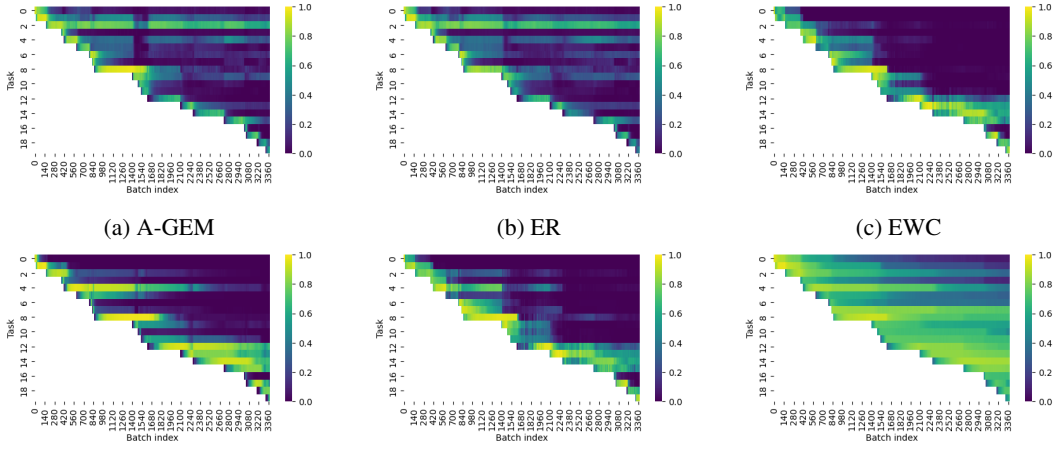
Figure 7: Anytime evaluation by task for the Amazon Computer dataset with GRNF backbone.



(a) A-GEM (b) ER (c) EWC

(d) LwF (e) MAS (f) SLDA

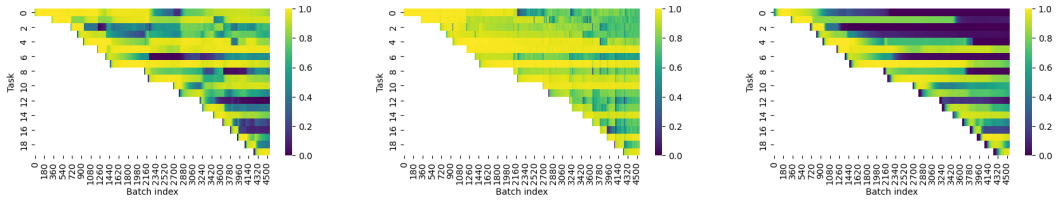
Figure 8: Anytime evaluation by task for the Arxiv dataset with UGCN backbone.



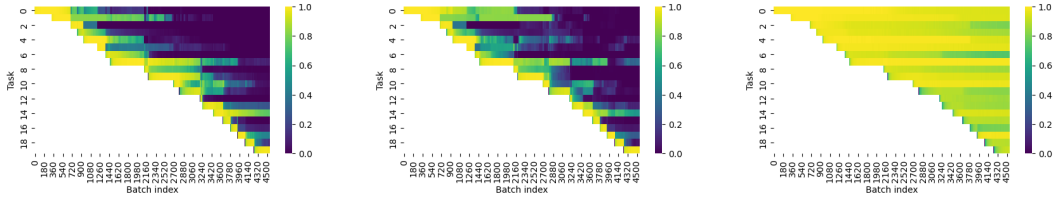
(a) A-GEM (b) ER (c) EWC

(d) LwF (e) MAS (f) SLDA

Figure 9: Anytime evaluation by task for the Arxiv dataset with GRNF backbone.

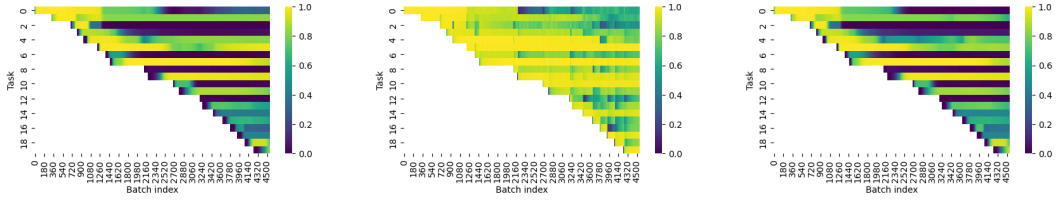


(a) A-GEM (b) ER (c) EWC

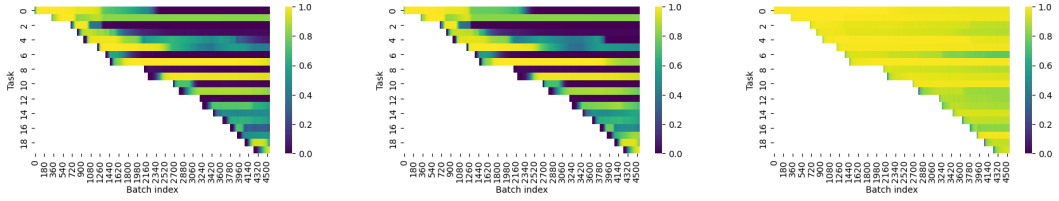


(d) LwF (e) MAS (f) SLDA

Figure 10: Anytime evaluation by task for the Reddit dataset with UGCN backbone.

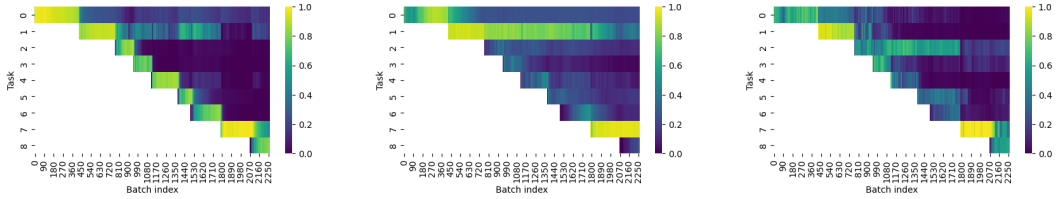


(a) A-GEM (b) ER (c) EWC

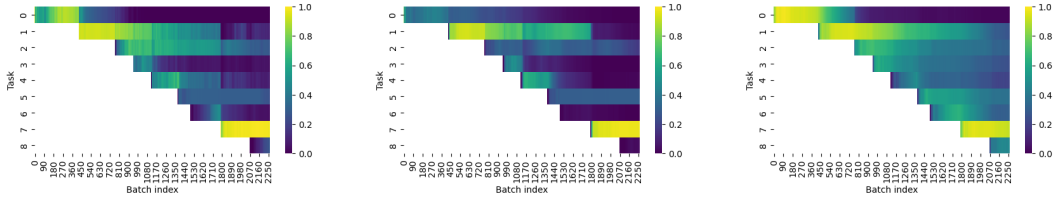


(d) LwF (e) MAS (f) SLDA

Figure 11: Anytime evaluation by task for the Reddit dataset with GRNF backbone.

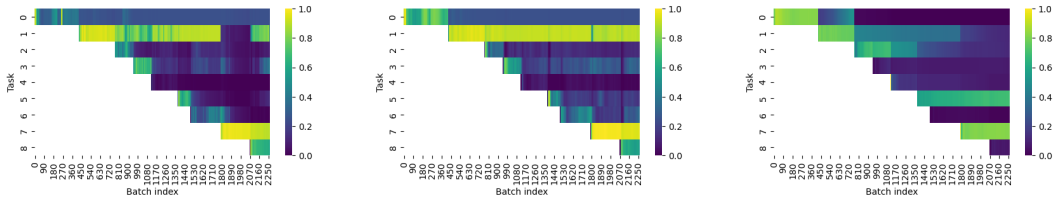


(a) A-GEM (b) ER (c) EWC

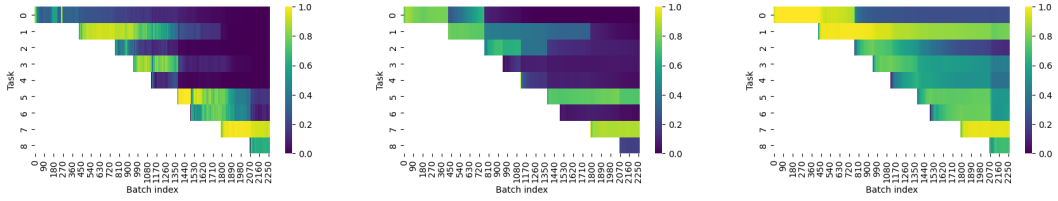


(d) LwF (e) MAS (f) SLDA

Figure 12: Anytime evaluation by task for the Roman Empire dataset with UGCN backbone.

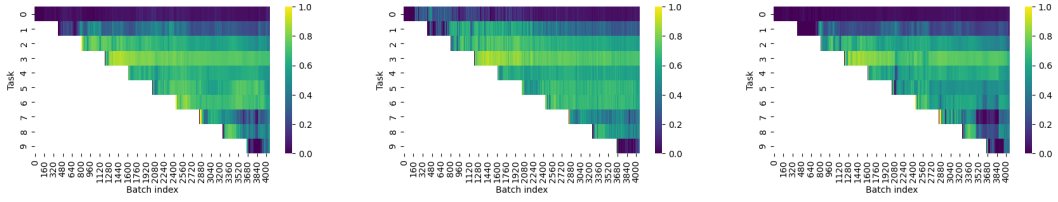


(a) A-GEM (b) ER (c) EWC

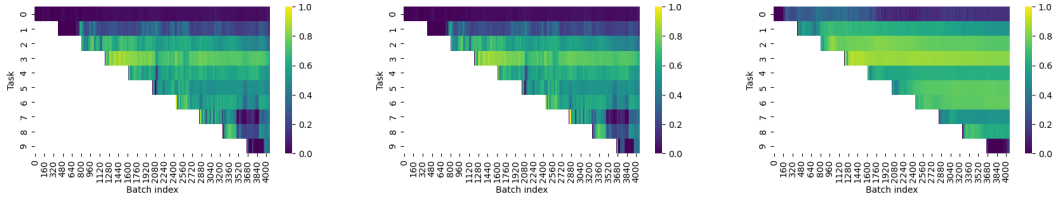


(d) LwF (e) MAS (f) SLDA

Figure 13: Anytime evaluation by task for the Roman Empire dataset with GRNF backbone.

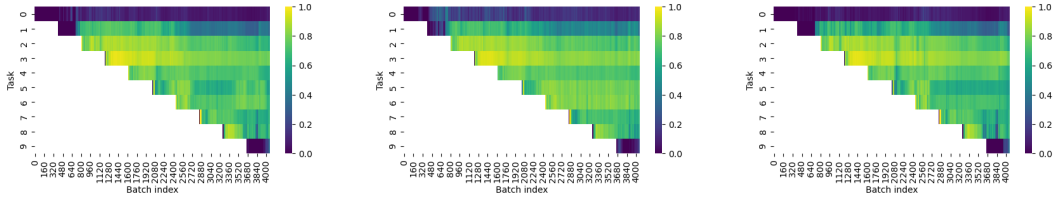


(a) A-GEM (b) ER (c) EWC

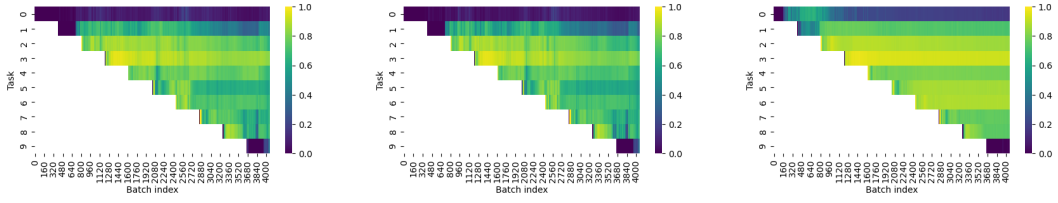


(d) LwF (e) MAS (f) SLDA

Figure 14: Anytime evaluation by task for the Elliptic dataset with UGCN backbone.

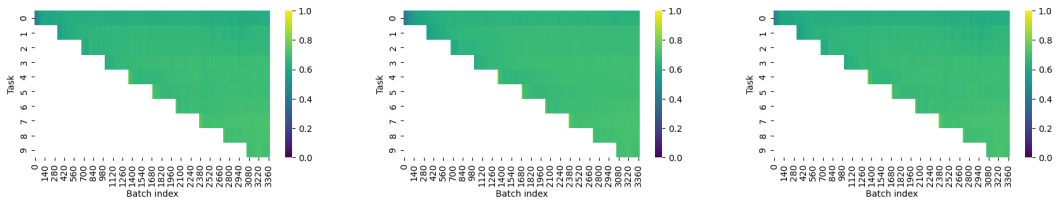


(a) A-GEM (b) ER (c) EWC

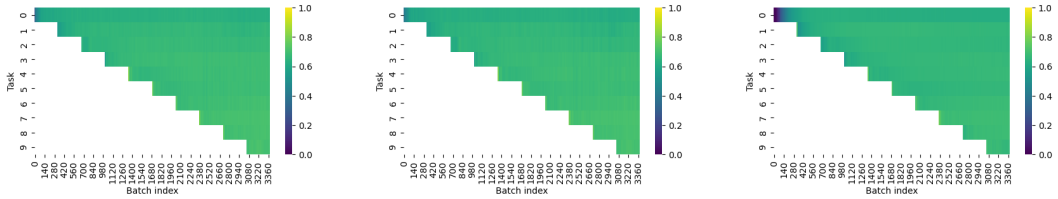


(d) LwF (e) MAS (f) SLDA

Figure 15: Anytime evaluation by task for the Elliptic dataset with GRNF backbone.

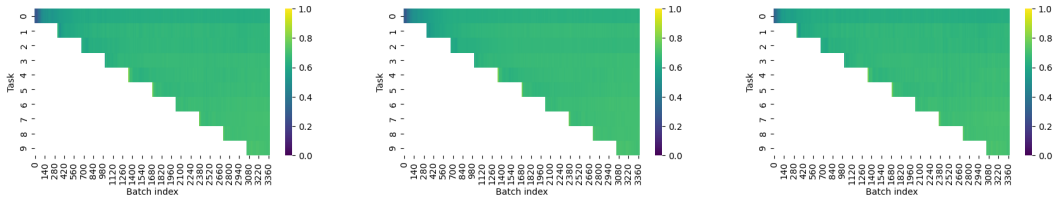


(a) A-GEM (b) ER (c) EWC

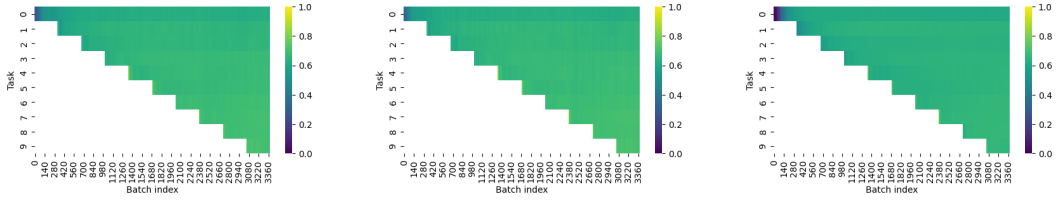


(d) LwF (e) MAS (f) SLDA

Figure 16: Anytime evaluation by task for the Arxiv dataset with UGCN backbone with time-incremental stream. We note how the very homogeneous performance compared to other benchmarks suggests the absence of significant distribution drifts between tasks.



(a) A-GEM (b) ER (c) EWC



(d) LwF (e) MAS (f) SLDA

Figure 17: Anytime evaluation by task for the Arxiv dataset with GRNF backbone with time-incremental stream. We note how the very homogeneous performance compared to other benchmarks suggests the absence of significant distribution drifts between tasks.