Machine Learning (2025) 114:270
https://doi.org/10.1007/s10994-025-06915-w

Efficient quantification on large-scale networks

Alessio Micheli[1] · Alejandro Moreo[2] · Marco Podda[1] · Fabrizio Sebastiani[2] · William Simoni[1] · Domenico Tortorella[1]

Received: 3 April 2025 / Revised: 24 September 2025 / Accepted: 8 October 2025

Abstract
Network quantification (NQ) is the problem of estimating the proportions of nodes belonging to each class in subsets of unlabelled graph nodes. When prior probability shift is at play, this task cannot be effectively addressed by first classifying the nodes and then counting the class predictions. In addition, unlike non-relational quantification, NQ demands enhanced flexibility in order to capture a broad range of connectivity patterns, resilience to the challenge of heterophily, and scalability to large networks. In order to meet these stringent requirements, we introduce XNQ, a novel method that synergizes the flexibility and efficiency of the unsupervised node embeddings computed by randomized recursive Graph Neural Networks, with an Expectation-Maximization algorithm that provides a robust quantification-aware adjustment to the output probabilities of a calibrated node classifier. In an extensive evaluation, in which we also validate the design choices underpinning XNQ through comprehensive ablation experiments, we find that XNQ consistently and significantly improves on the best network quantification methods to date, thereby setting the new state of the art for this challenging task. XNQ also provides a training speed-up of up to 10x–100x over other methods based on graph learning.

Keywords Quantification · Network quantification · Graph neural networks · Graph learning · Reservoir computing

1 Introduction

Quantification (Esuli et al., 2023; González et al., 2017) is the machine learning task of estimating the prevalence (or proportions) of each class in a set of unlabelled datapoints. Unlike standard classification, which focuses on predicting a label for each individual example, quantification works at the aggregate level by estimating the overall fraction of unlabelled

Alessio Micheli, Alejandro Moreo, Marco Podda, Fabrizio Sebastiani, William Simoni, and Domenico Tortorella contributed equally to this work.

Editors: Riccardo Guidotti, Anna Monreale, Dino Pedreschi.

Extended author information available on the last page of the article

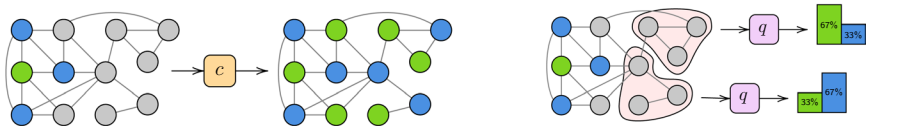
datapoints belonging to each class. Quantification finds application in all disciplines interested in characterizing populations (such as the social sciences, political science, epidemiology, ecological modelling, and market research), but also in downstream tasks such as improving the accuracy of classifiers (Saerens et al., 2002), measuring the fairness of classifiers (Fabris et al., 2023) and rankers (Jaenich et al., 2025), predicting the accuracy of classifiers (Volpi et al., 2024; Volpi et al.., 2025), and calibrating classifiers (Moreo, 2025).

Quantification methods are explicitly designed to account for *dataset shift*, which occurs when the statistical properties of the training data differ from those of the test data, due to changes in input features, labels, or their relationships. Most quantification methods are tailored to one specific type of dataset shift, namely, *prior probability shift* (PPS) (Storkey, 2009), also referred to as "label shift" (Lipton et al., 2018). Specifically, PPS occurs when the class-conditional distribution of the input features does not change across the training and the test data (i.e., $P_{\text{tr}}(X|Y) = P_{\text{te}}(X|Y)$), while the prior distribution of the class labels does (i.e., $P_{\text{tr}}(Y) \neq P_{\text{te}}(Y)$). In simple words, the class proportions in the training set might differ significantly from those of the test set. In data affected by PPS, standard classifiers have been repeatedly shown to be inaccurate quantifiers, leading to class prevalence estimates biased towards the class distribution of the training set (González et al., 2024). Therefore, quantification has evolved from traditional classification, with its own models and custom evaluation protocols that assess performances while varying the class proportions in the test data to simulate PPS (see Sect. 2.1).

This work deals with the task of *network quantification* (**NQ**), which consists of performing quantification on interlinked datapoints, i.e., on nodes belonging to a graph. As noted by Tang et al. (2010), NQ is suitable in settings where the goal is to estimate how a population of interrelated individuals is distributed according to classes of interest (e.g., to infer the proportion of spam or malicious accounts in a social network).

Analogously to the distinction between quantification and classification in non-relational domains, there is substantial difference between NQ and the task of node classification (Sect. 2.2). The latter is concerned with predicting labels or assigning categories to nodes in a graph based on their features and the network topology (Bacciu et al., 2020), i.e., focusing on predicting the individual classes of the unlabelled nodes, while the purpose of NQ is to predict their aggregate class distribution among different network sub-communities. The example in Fig. 1 summarizes the main differences between these two classes of tasks.

Due to its specific setting, NQ is also fundamentally different from plain quantification since it is applied to the nodes of a graph, which are not independent and identically distrib-



(a) Node classification (b) Network quantification (this work)

Fig. 1 Differences between node classification and network quantification on a partially unlabelled graph where nodes may belong to the "blue" or "green" classes. Unlabelled nodes are shown in gray. Node classification (**a**) is performed by a node classifier c, which takes as input the partially unlabelled graph and returns as output an isomorphic graph where the class of the unlabelled nodes has been predicted. In contrast, network quantification (**b**) uses a quantifier q, which takes as input subsets of unlabelled nodes (in light pink shades) and returns as output their class distribution. In this work, we study network quantification under prior probability shift

uted (i.i.d.) like non-relational datapoints, but rather interdependent according to the graph structure. Moreover, real-world networks are usually large-scale and characterized by complex properties such as non-linear connectivity patterns and heterophily (i.e., prevalence of inter-class edges), which non-relational quantification models are unable to capture.

This discussion highlights that, in order to be proficient at NQ, a learning method should (i) exploit the network connectivity to emit coherent prevalence predictions; (ii) be powerful enough to capture the complex properties of real-world networks; (iii) be flexible enough to adapt to their possibly heterophilic nature; and (iv) be efficient and resource-friendly to make operating at scale feasible. However, existing methods for NQ (reviewed in Sect. 3) hardly comply with all these requirements.

To satisfy these *desiderata* we propose XNQ, a model that integrates the representational capabilities of randomized recursive Graph Neural Networks with a powerful Expectation-Maximization approach for quantification. XNQ is purposely designed to be resource-efficient, scalable, and resilient to heterophily. Through a comprehensive experimental evaluation, we show that XNQ significantly outperforms the best methods proposed so far for NQ, setting a new state of the art. Additionally, we validate our design through extensive ablation studies, showing that XNQ achieves the best trade-off in terms of performance and computational efficiency.

The paper is organized as follows. After discussing background concepts (Sect. 2) and describing related work (Sect. 3), in Sect. 4 we move to detail the proposed method. In Sect. 5, XNQ is compared against the current state-of-the-art methods on several real-world graphs, with an in-depth ablation study and efficiency analysis. Finally, Sect. 6 draws conclusions and points out avenues for future research.

2 Background and notation

In this section, we introduce basic concepts and notation on quantification and graph learning necessary to understand our contribution.

2.1 Quantification

Let $\mathcal{X}$ be a generic input domain and $\mathcal{Y}$ be a discrete set of class labels. Assume a dataset of pairs $\mathcal{D} = \{(\boldsymbol{x}, y) \,|\, \boldsymbol{x} \in \mathcal{X}, y \in \mathcal{Y}\}$ which gives point-wise estimates of some unknown function we wish to learn. For the sake of simplicity, we formalize our method for the binary case $\mathcal{Y} = \{\oplus, \ominus\}$, using $\oplus$ to denote the "positive" class and $\ominus$ to denote the "negative" class; however, our method natively works in the multi-class case as well, as we show in Sect. 5.5. We use the symbol S to denote a *sample set*, i.e., a non-empty subset of (labelled or unlabelled) elements from $\mathcal{X}$. Let $p_S(y)$ be the true prevalence of class y in sample set S (i.e., the fraction of items in S that belong to y). Note that $p_S(y)$ is just a shorthand of $\Pr(Y = y \,|\, \boldsymbol{x} \in S)$, where $\Pr(\cdot)$ indicates probability and Y is a random variable that ranges on $\mathcal{Y}$. In the binary case, since $p_S(\ominus) = 1 - p_S(\oplus)$ holds true, it is sufficient to estimate the prevalence of the positive class only. A *binary quantifier* q is a predictor of the class prevalence $p_S(\oplus)$ in sample S. We characterize quantifiers by the class prevalence estimates they produce, using the notation $\hat{p}^q_S(y)$ to indicate that $\hat{p}_S(y)$ has been computed through q.[1] In

[1] We omit the superscript when the NQ method is clear from the context.

this work, we focus on *aggregative* quantification methods, i.e., methods that work on top of a trained classifier by aggregating their individual predictions.

2.1.1 Quantification methods

A trivial but inaccurate method to tackle quantification is *Classify and Count* (**CC**), which works by training a classifier, classifying all the unlabelled datapoints, counting the datapoints assigned to each class, and normalizing the counts so that the result is a probability distribution over the classes. Indeed, CC has been repeatedly shown to deliver incorrect class prevalence estimates (Bella et al., 2010; Esuli et al., 2023; Forman, 2008; González et al., 2017; González-Castro et al., 2013). One of the reasons is that classification and quantification are characterized by different loss functions, since (using the binary case as an example) in classification we want to minimize some proxy of $(\mathrm{FP} + \mathrm{FN})$, where FP (resp. FN) stands for the number of false positives (resp. negatives). In contrast, in quantification we want to minimize some proxy of $|\mathrm{FP} - \mathrm{FN}|$. Another reason is that data are often characterized by dataset shift, and the algorithms we routinely use to train our classifiers assume that no dataset shift is at play, i.e., that the training data and the test data are i.i.d.

Many quantification methods that improve on CC, especially in terms of robustness to PPS, have been proposed in the literature (see Appendix A for a detailed presentation, and Esuli et al. (2023); González et al. (2017) for more exhaustive surveys). One of the earliest is *Adjusted Classify and Count* (**ACC**), which applies to the class prevalence estimates generated by CC a correction based on the misclassification rates of the classifier as estimated via *k*-fold cross-validation (Forman, 2008). *Probabilistic Classify and Count* (**PCC**) and *Probabilistic Adjusted Classify and Count* (**PACC**) are probabilistic variants of CC and ACC, in which the integer counts and the misclassification rates needed to compute CC and ACC are replaced with their soft (i.e., probabilistic) versions (Bella et al., 2010). A radically different approach is instead embodied in **HDy** (González-Castro et al., 2013), a method that views quantification as the problem of minimizing the divergence (measured in terms of the Hellinger Distance) between two distributions of posterior probabilities, and by **DyS** (Maletzke et al., 2019), which uses the Topsøe distance instead of the Hellinger distance.

2.1.2 Evaluating quantifiers

The standard approach to assess the performance of a quantifier q is to simulate PPS, i.e., evaluate its estimations for considerably different degrees of divergence between $P_{\mathrm{tr}}(Y)$ and $P_{\mathrm{te}}(Y)$. In the binary case, the widely adopted *artificial prevalence protocol* (APP) [Esuli et al. (2023)§3.4.2] involves randomly generating (i.e., extracting from the unlabelled data) test subsets S that exhibit predetermined prevalence values lying on a regular grid, e.g., $p_S(\oplus) \in \{0.00, 0.05, ..., 0.95, 1.00\}$. In the general multi-class case, instead, the APP is implemented by:

1. randomly generating (usually, via the Kraemer algorithm—see Esuli et al. (2023, §3.4.3)) vectors of prevalence values that uniformly cover the probability simplex, i.e., the space of all legitimate vectors of prevalence values given by

$\Delta^{n-1} = \{p_1, \ldots, p_n : p_i \geq 0, \sum_{i=1}^{n} p_i = 1\}$, with p_i the prevalence of the ith class and $n = |\mathcal{Y}|$ the number of classes, and then

2. extracting, from the unlabelled data, test subsets S each complying with one of the randomly generated prevalence vectors.

In this experimental protocol, the class-conditional distributions across the training data and the test subsets remain stationary (meaning that $P_{\text{tr}}(X|Y) = P_{\text{te}}(X|Y)$), whereas the class prevalence values of a test sample ($P_{\text{te}}(Y)$) may be significantly different from those encountered in the training data ($P_{\text{tr}}(Y)$). The *Absolute Error* (AE), defined as:

$$\text{AE}(p_S, \hat{p}_S) = \frac{1}{n} \sum_{y \in \mathcal{Y}} |\hat{p}_S(y) - p_S(y)|, \tag{1}$$

between the true prevalence p_S and the estimated prevalence $\hat{p}_S^q$, averaged across all the test subsets S sampled via the APP, provides a concise metric of quantifier q's performance. In addition to AE, we also consider the *relative absolute error* (RAE), defined as:

$$\text{RAE}(p_S, \hat{p}_S) = \frac{1}{n} \sum_{y \in \mathcal{Y}} \frac{|\hat{p}_S(y) - p_S(y)|}{p_S(y) + \varepsilon}, \tag{2}$$

where ε is a smoothing factor used to avoid division by zero when $p_S(y) = 0$, which we set, following Forman (2008), to $\varepsilon = 1/(2|S|)$. AE and RAE are the two *de facto* standard evaluation measures for binary and multi-class quantification [Esuli et al. (2023), §3.1], and it has been argued (Sebastiani, 2020) that AE better mirrors the needs of some applications of quantification while RAE better mirrors others.

In the binary case, diagonal plots (Esuli et al., 2023) are a useful tool to visualize a quantifier performance across different prevalence values. Figure 2 is an example of such plots: the x axis presents the true class prevalence $p(\oplus)$, while the y axis displays the estimated prevalence $\hat{p}(\oplus)$. The dashed diagonal represents the ideal quantifier behaviour $\hat{p}(\oplus) = p(\oplus)$. A quantifier q is visualized as a curve of points $(p_S(\oplus), \hat{p}_S^q(\oplus))$: in our example q_2 is closer to the diagonal of the ideal quantifier, and thus superior to q_1.

2.2 Graph learning

We define a graph G by a set of nodes $\mathcal{V} = \{v_i : 1 \leq i \leq |\mathcal{V}|\}$ and a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of edges among them, encoded as an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ whose entries $\boldsymbol{A}_{ij}$ are 1 if $(v_i, v_j) \in \mathcal{E}$ and 0 otherwise. We define the set of neighbours of a node v_i as $\mathcal{N}(v_i) = \{v_j : (v_j, v_i) \in \mathcal{E}\}$, and the set of node features as $\boldsymbol{X} = \{\boldsymbol{x}_v \in \mathbb{R}^d : v \in \mathcal{V}\}$ for some $d \in \mathbb{N}$. On graphs defined as such, the task of *node classification* consists of learning a function $f : \mathcal{V} \to \mathcal{Y}$ that maps nodes to labels $y_v \in \mathcal{Y}$. Typically, node classification is a *transductive* task, meaning that the learning algorithm has access to the entire structure of G but only to a subset $\mathcal{V}_{\text{labelled}} \subset \mathcal{V}$ of the node labels, and the goal is to predict the labels on the unlabelled subset $\mathcal{V}_{\text{unlabelled}} = \mathcal{V} \backslash \mathcal{V}_{\text{labelled}}$.

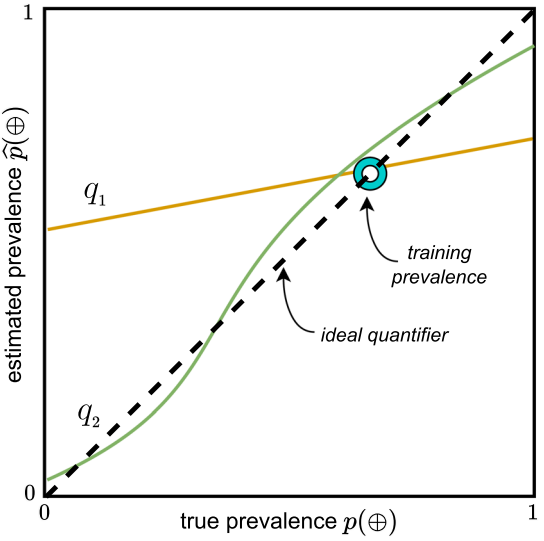


Fig. 2 In the binary case, diagonal plots provide a visual tool to compare quantifiers. Here, q_2 is closer to the ideal quantifier behaviour (dashed diagonal), and thus superior to q_1

2.2.1 Graph neural networks

Learning from graph data poses unique challenges such as handling variable-sized topologies, capturing potentially non-linear connectivity patterns, and managing cycles. Graph neural networks (GNNs) (Bacciu et al., 2020; Wu et al., 2021) facilitate the adaptive processing of graphs by iteratively refining node representations (*embeddings*) through neighbourhood aggregations, thereby progressively increasing the receptive field of the nodes (Micheli, 2009). In practice, a GNN computes the embedding of a node v by taking as input its features $\boldsymbol{x}_v$ and those of its neighbours $\{\boldsymbol{x}_u : u \in \mathcal{N}(v)\}$, returning as output a node embedding $\boldsymbol{h}_v^{(L)} \in \mathbb{R}^{d'}$ (for some $d' \in \mathbb{N}$) obtained after $L \geq 1$ local message-passing iterations. Once computed, node embeddings can be used to learn downstream tasks.

Within the GNN family, convolutional approaches compute node embeddings by stacking multiple message-passing layers, allowing to learn tasks in an end-to-end fashion. However, training convolutional GNNs poses several challenges, including a bias towards graphs with high homophily (Zhu et al., 2020) and the issue of over-smoothing, which causes the node embeddings to become indistinguishable as the number of layers increases (Chen et al., 2020). Addressing heterophily, i.e., a prevalence of inter-class edges between nodes (the opposite of homophily), is related to over-smoothing, as successive message-passing iterations make the embeddings of neighbouring nodes more similar (Yan et al., 2022).

In contrast, recursive GNN approaches (Scarselli et al., 2009) frame the embedding computation as an iterative map akin to the state transition function of a dynamical system. Instead of having different message-passing layers each with its own weights, recursive GNNs apply the same message-passing function within a layer for L iterations. For graph-level tasks using a pooled representation of the entire graph, the recursive map is required to possess contractive dynamics, i.e., a Lipschitz constant smaller than 1 (Tortorella et al., 2022), while node-level tasks usually take advantage of the opposite (Micheli & Tortorella, 2023). *Graph Echo State Networks* (**GESN**) belong to this class of graph models, adopting

additionally the reservoir computing paradigm (Lukoševičius & Jaeger, 2009; Nakajima & Fischer, 2021), meaning that the weights of the recursive map are specifically initialized to satisfy constraints on the Lipschitz constant and frozen, while only the task prediction layer is trained (Gallicchio & Micheli, 2010, 2020).

3 Related work

Tang et al. (2010) introduced two distinct NQ approaches. The first employs the *weighted vote Relational Neighbour* (**wvRN**) (Macskassy & Provost, 2003) algorithm to classify all nodes in the graph. It works by computing an initial prediction $\hat{y}_v^{(0)}$ for all nodes $v \in V$ using a base classifier, and then updating the predictions as:

$$\hat{y}_v^{(t)} \leftarrow \arg\max_y \sum_{u \in \mathcal{N}(v)} w_{uv} \mathbb{1}[\hat{y}_u^{(t-1)} = y] \qquad t \geq 1 \tag{3}$$

which corresponds to propagating to the current node the most frequent label among its neighbours, weighting their vote by the strength of the connection $w_{uv} \in \mathbb{R}$. Once the propagation has converged, a quantification algorithm is applied. The second approach by the same authors, called Link-Based Quantification (LBQ), is out of the scope of this study since it is only suited for graph-level quantification and is non-aggregative, i.e., it provides class prevalence estimations without an intermediate classification step.

Milli et al. (2015) proposed two other methods, *Community Discovery for Quantification* (**CDQ**) and *Ego Networks for Quantification* (**ENQ**). Both methods initially assign a class to each unlabelled node $v \in \mathcal{V}_{\text{unlabelled}}$ and then apply quantification on top. CDQ employs a community discovery algorithm to group nodes based on the network's topological structure. An unlabelled node $v \in \mathcal{V}_{\text{unlabelled}}$ is assigned the most frequent class within its community. If v belongs to multiple communities, two strategies are considered: a frequency-based strategy, which assigns the class label with the highest relative frequency, and a density-based strategy, which assigns the most frequent class of the denser community. The ENQ labeling process utilizes the concept of ego networks. The ego network of radius r of a node v is the sub-network that includes v, referred to as the ego, and all nodes within the r-hop neighbourhood of v. The class of v is then the most frequent class within its ego network. In the case of isolated nodes, both algorithms assign the class following the training distribution. One drawback of the methods discussed above is that they are not designed to leverage node features. Another major limitation is that they assume homophily within the graph, since in the intermediate classification step they assign labels to nodes based on the labels of their neighbours.

4 Method

We start this section by restating the objective of NQ for clarity. Given a partially labelled graph G with $\mathcal{V} = \mathcal{V}_{\text{labelled}} \cup \mathcal{V}_{\text{unlabelled}}$, our goal is to produce an estimate $\hat{p}_S(\oplus)$ of the proportion of positive nodes in any unlabelled subset $S \subseteq \mathcal{V}_{\text{unlabelled}}$. To tackle NQ the problem, we develop the *eXtreme Network Quantifier* (**XNQ**) model, referring to its

enhanced efficiency and effectiveness. At a high level, XNQ is composed of three modules applied sequentially:

1. An unsupervised *node embedder* which computes node embeddings leveraging the node features and the graph structure;
2. An intermediate *readout classifier* which takes the node embeddings as input and computes the node class posterior predictions as output;
3. A downstream *aggregative quantifier* which aggregates the classifier's posterior predictions and estimates the class prevalence values.

The three components are shown visually in Fig. 3. In the following, we describe each component in detail.

4.1 Unsupervised node embedder

Differently from existing NQ methods, XNQ leverages the node representations computed by a GNN model in order to exploit information on input node features as well as the comprehensive graph topology. To satisfy the requirement of efficiency, XNQ exploits a reservoir computing GNN such as GESN to embed nodes in an *unsupervised* and *untrained* fashion, allowing it to scale to larger networks without requiring a too large fraction of annotated nodes: as opposed to end-to-end trained GNNs, target nodes are not used for learning node representations, but only for training the classifier readout. GESN-based models have proven effective in solving node classification tasks, reaching state-of-the-art accuracy on several heterophilic graph benchmarks, while also reducing computation time compared to fully-trained graph neural networks (Micheli & Tortorella, 2023). Specifically, in XNQ node embeddings $\boldsymbol{h}_v^{(L)}$ are recursively computed by the following dynamical system, called the *reservoir*, which implements an untrained GNN layer, i.e.,

$$\boldsymbol{h}_v^{(0)} \leftarrow \mathbf{0}, \qquad \boldsymbol{h}_v^{(\ell)} \leftarrow \tanh\left(\boldsymbol{W}_{\text{in}}\,\boldsymbol{x}_v + \sum_{u \in \mathcal{N}(v)} \hat{\boldsymbol{W}}\,\boldsymbol{h}_u^{(\ell-1)} + \boldsymbol{b}_{\text{in}}\right) \tag{4}$$

where $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{d' \times d}$ and $\hat{\boldsymbol{W}} \in \mathbb{R}^{d' \times d'}$ are the input-to-reservoir and the reservoir recurrent weights, respectively ($\boldsymbol{b}_{\text{in}} \in \mathbb{R}^{d'}$ is the input bias). The embedding dimension $d' \in \mathbb{N}$ is a hyperparameter chosen by model selection, and its value is typically much larger than the input dimension d. Reservoir weights are randomly initialized from a uniform distribution, and then rescaled to the desired input range and reservoir spectral radius (also chosen via model selection), without requiring any training. The number of message-passing iterations

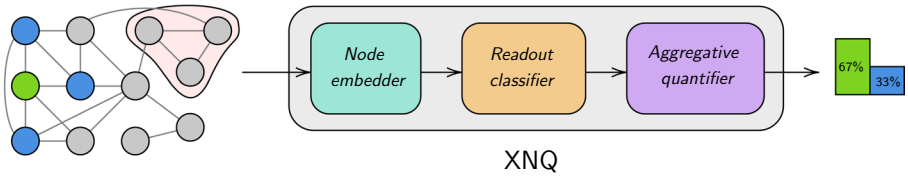


Fig. 3 XNQ is composed of three modules applied sequentially

$1 \leq \ell < L$ is set to be larger than the graph diameter, so as to have a comprehensive receptive field. Of crucial importance is the Lipschitz constant of the recursive map defined in Eq. (4), which is controlled by setting the spectral radius of $\hat{\boldsymbol{W}}$, i.e., the largest eigenvalue modulus $\rho(\hat{\boldsymbol{W}})$. Initializing the recurrent matrix with $\rho(\hat{\boldsymbol{W}}) < 1/\rho(\boldsymbol{A})$, where $\rho(\boldsymbol{A})$ is the graph spectral radius, implies that the map is contractive and that the sensitivity of the node embeddings to long-range interactions is exponentially vanishing (Micheli & Tortorella, 2023). Since this setting leads to over-smoothing, node-level tasks frequently benefit from recurrent matrix initializations with $\rho(\hat{\boldsymbol{W}}) > 1/\rho(\boldsymbol{A})$. This holds particularly true for heterophilic graphs, where sensitivity only to the immediate neighbours may lead to misleading representations. After being iteratively computed by Eq. (4) on the whole graph G, the node embeddings $\boldsymbol{h}_v^{(L)}, \forall v \in \mathcal{V}$ are passed to the readout classifier.

4.2 Intermediate readout classifier

XNQ uses a trained logistic regression readout module to compute the node predictions. The use of logistic regression is tightly coupled with choosing a GESN-based embedder, since the high-dimensional expansion performed by the reservoir usually results in a linear separation of the embeddings. Consequently, a linear model can be used to learn the classification rule, further contributing to making the approach extremely efficient. Specifically, our readout module takes the node embeddings $\boldsymbol{h}_v^{(L)}$ as input, and computes a raw posterior probability $\bar{y}_v \in [0, 1]$ as

$$\bar{y}_v \leftarrow \text{sigmoid}\left(\boldsymbol{w}_{\text{out}}^{\top} \boldsymbol{h}_v^{(L)} + b_{\text{out}}\right) \tag{5}$$

where $\boldsymbol{w}_{\text{out}} \in \mathbb{R}^{d'}$ are learnable weights and $b_{\text{out}} \in \mathbb{R}$ is a learnable bias. Once the readout has been learned, the output probabilities are calibrated and passed to the downstream quantifier. Calibration entails adjusting the output probabilities such that $\Pr(Y = \oplus \,|\, \bar{Y} = \bar{y}_v) \approx \bar{y}_v$, where $\bar{Y}$ is a random variable that ranges over [0, 1]. In other words, by calibrating the output of the readout we are adjusting the predicted probabilities to approximately match the observed class frequencies. In our implementation, calibration is achieved via Platt scaling (Platt, 2000), i.e., by transforming the raw posterior probabilities as

$$\hat{y}_v \leftarrow \frac{1}{1 + e^{(a\,\bar{y}_v + b)}} \tag{6}$$

where $a, b \in \mathbb{R}$ are parameters learned with maximum likelihood. In preliminary experiments we also tested, as an alternative calibration method, Bias-Corrected Temperature Scaling (BCTS), which in related work (Alexandari et al., 2020) had shown to improve quantification performance; however, in our experiments, Platt scaling revealed superior, and we thus decided to stick to it. The readout classifier is trained and calibrated using the labelled node embeddings $\boldsymbol{h}_v^{(L)}$, $\forall v \in \mathcal{V}_{\text{labelled}}$. Then, we use it to predict the unlabelled nodes in $\mathcal{V}_{\text{unlabelled}}$, obtaining a set of calibrated posterior probabilities $\hat{P}_{\text{tr}}(\oplus \,|\, \boldsymbol{h}_v^{(L)})$, for all $v \in \mathcal{V}_{\text{unlabelled}}$.

Note that the calibration property is defined upon a set (or a distribution), meaning that a classifier that has been calibrated for a training distribution will hardly generate well-calibrated posteriors for a different test distribution, especially when the training and test distributions

are related by PPS, since its underlying assumptions imply $P_{\text{tr}}(\oplus \mid \boldsymbol{h}_v^{(L)}) \neq P_{\text{te}}(\oplus \mid \boldsymbol{h}_v^{(L)})$. How to properly adapt the (estimated) posteriors to the (unknown) test prior is the subject of the next section.

4.3 Downstream aggregative quantifier

The goal of XNQ's downstream quantifier is to output the desired estimate $\hat{p}_S(\oplus)$ in an unlabelled subset $S \subseteq \mathcal{V}_{\text{unlabelled}}$ using the observed training proportion $p_{\text{labelled}}(\oplus)$ and the estimated posterior probabilities $\hat{P}_{\text{tr}}(\oplus \mid \boldsymbol{h}_v^{(L)})$, $\forall v \in S$ as inputs. We do so by adapting the *Saerens-Latinne-Decaestecker* method (Saerens et al., 2002) (**SLD**) to our setting. The rationale behind the choice are its strong performance in the challenge of quantification on data affected by PPS and its desirable theoretical guarantees. Indeed, SLD is proven to be *Fisher-consistent* under PPS (Tasche, 2017), i.e., its class prevalence estimates are guaranteed correct under PPS if computed on the whole populations of interest (instead of the limited samples $\mathcal{V}_{\text{unlabelled}}$ and S). Basically, SLD is an instance of the Expectation-Maximization (EM) algorithm. Initially, the prevalence estimates are set to $\hat{p}_S^{(0)}(\oplus) \leftarrow p_{\text{labelled}}(\oplus)$. Then, two mutually recursive steps are iterated (for $k \geq 1$):

- the *E-step*: The posterior probability $P_{\text{te}}(\oplus \mid \boldsymbol{h}_v^{(L)})$ is approximated by scaling the estimated posterior $\hat{P}_{\text{tr}}(\oplus \mid \boldsymbol{h}_v^{(L)})$ by the ratio between the previous estimate $\hat{p}_S^{(k-1)}(\oplus)$ and the initial estimate $\hat{p}_S^{(0)}(\oplus)$, and re-normalized. This tunes the posterior probabilities towards the current class prevalence estimate.
- the *M-step*: The current estimate $\hat{p}_S^{(k)}(\oplus)$ is updated as the average of the current posterior estimates obtained in the *E-step*. This tunes the class prevalence estimate towards the rescaled posterior probabilities.

The process is repeated until the estimate remains stable through successive iterations, at which point the final estimate $\hat{p}_S(\oplus)$ is returned.

5 Experiments and discussion

We compare our proposed XNQ against the previous literature methods described in Sect. 3 (wvRN, CDQ, ENQ) with the exception of LBQ as it cannot operate quantification at the node level. For each baseline, we optimize the downstream quantification method using the same experimental protocol as XNQ.

5.1 Experimental protocol

5.1.1 Datasets

We select five publicly available datasets from the node classification literature, adapting them to NQ:

- *Cora*: a citation network first introduced in (McCallum et al., 2000). Nodes in the graph are scientific publications, links are citations, and the task consists of picking the scien-

tific field the publication belongs to from a set of 7 such fields (Yang et al., 2016). Each node has textual features extracted from the abstract.

- *Genius*: the social network from a crowd-sourced song annotation platform (Lim et al., 2021). Links represent friendship between users, whose accounts are classified as regular or spam.
- *Questions*: a network from the question-answering website Yandex Q where nodes are users, and links specify whether one user answered the other's questions within a certain time span (Platonov et al., 2023). User accounts are classified into active or inactive.
- *Tolokers*: a network from the Toloka platform (Likhobaba et al., 2023), where nodes are workers who participated in crowd-sourcing projects, and links specify mutual participation. The positive class corresponds to banned users (Platonov et al., 2023).
- *Twitch-DE*: a social network of German users from the Twitch streaming platform (Rozemberczki et al., 2021). The positive class corresponds to adult content profiles.

All datasets except Cora are real-world networks that range in size up to 400K nodes and 1 M edges, and present binary classification tasks characterized by a high degree of heterophily (i.e., a large fraction of inter-class edges). Cora is instead a multi-class dataset (with 7 classes); from it we have derived a binary 1-vs-rest classification task by picking class 2 as the "positive class" $\oplus$, to serve as a high-homophily control case. Table 1 reports the main statistics of these datasets, including the prevalence of the positive class and the class-adjusted homophily (Lim et al., 2021), which measures the prevalence of intra-class edges in the presence of class imbalance (higher values indicate higher homophily).

5.1.2 Evaluation setting

We use 5-fold cross-validation (CV) to estimate the out-of-sample AE. The set of nodes is divided in 5 equally-sized, disjoint partitions. In turn, 4 folds are used as development set, while the remaining fold is reserved for test evaluation. In each development set, 25% of the data is held out as validation set to perform model selection via grid search; an additional 12.5% of the data is held out for calibration (which is required by quantification methods such as PCC, PACC, HDy, DyS, SLD); the remaining 62.5% of the development data is used for training. The model configuration achieving the lowest AE estimated from 100 instances of the artificial prevalence protocol (Sect. 2.1) applied on the validation set is then evaluated on the corresponding test set. The average AE score estimated via the APP protocol on each of the 5 test folds is reported, along with the standard deviation. We remark that all combinations of models, hyper-parameters and quantification methods are trained and evaluated on the same data splits to ensure fairness. Further information allowing to reproduce the experiments is provided in the code repository.

Table 1 Main characteristics of the datasets considered in this study

	Cora-2	Genius	Questions	Tolokers	Twitch-DE
Number of nodes	2,708	421,961	48,921	11,758	9,498
Number of edges	5,429	984,979	307,080	1,038,000	315,774
Node input features	1,433	12	301	10	128
Prevalence of the $\oplus$ class	0.15	0.20	0.03	0.02	0.61
Class-adjusted homophily	0.89	0.22	0.18	0.08	0.17

5.1.3 Hyperparameters

XNQ hyperparameters were optimized using randomized search. Specifically, we sampled and evaluated 100 configurations choosing among the following: embedding size (sampled uniformly from the set $\{512, 1024, 2048, 4096\}$ (except for Genius, in which we restricted the exploration to $\{512, 1024\}$ due to hardware limitations), recurrent scale (sampled log-uniformly from the interval (Alexandari et al., 2020; Lipton et al., 2018)), input scale (sampled log-uniformly from the interval [0.1, 1]), and readout regularization coefficient (sampled log-uniformly from the interval $[10^{-2}, 10^{3}]$).

The existing methods with which we compare to were tuned with grid search as follows. For CDQ, we tuned the merge strategy, choosing between frequency-based and density-based. These are the only hyperparameters of this method. For ENQ, we tuned the hops to extract the ego-networks from the nodes, choosing from the set $\{1, 2\}$, limiting the exploration to either 1 or 2 hops (higher values revealed computationally intractable). Since both methods only assign crisp (non-probabilistic) labels, they choose the quantifier from the set $\{\mathrm{CC}, \mathrm{ACC}\}$. For wvRN, we tuned the number of iterations choosing from the set $\{1, 2, 3, 4, 5\}$, while the quantifier was chosen from the set $\{\mathrm{CC}, \mathrm{ACC}, \mathrm{PCC}, \mathrm{PACC}, \mathrm{HDy}, \mathrm{DyS}, \mathrm{SLD}\}$.

The ablation baselines were tuned via grid search as follows. For LR, we optimized the regularization coefficient in a grid of 10 log-uniformly distributed values in the interval $[10^{-2}, 10^{3}]$. For the convolutional GNNs, we optimized the following hyperparameters: number of layers (from the set $\{2, 3, 4\}$) and embedding dimension (from the set $\{128, 256\}$). Both LR and the convolutional GNNs also optimized the quantifier choice by inspecting the set $\{\mathrm{CC}, \mathrm{ACC}, \mathrm{PCC}, \mathrm{PACC}, \mathrm{HDy}, \mathrm{DyS}, \mathrm{SLD}\}$.

In order to guarantee a fair comparison in terms of resource constraints, we granted a budget of 24 h of computation to every model for exploring the hyperparameter space. Therefore, we are confident that our model selection setup is sufficiently fair given the computational constraints.

5.2 Results

Table 2 reports the average AE across the 5 CV folds using the APP testing protocol. XNQ achieves the lowest AE average across all 5 datasets, outperforming all existing NQ methods to date. The most significant improvement with respect to previous state-of-the-art (a 90.48% reduction in AE) is observed on the Genius dataset, which has the largest number of nodes. On the Tolokers dataset, characterized by the highest number of edges and the least homophily, the improvement is 59.49%. For the Questions and Twitch-DE datasets, with a positive class prevalence of 0.03 and 0.61 respectively (the former highly unbalanced,

Table 2 Results of the evaluation (best performance in boldface, second-best performance underlined)

Method	Cora-2	Genius	Questions	Tolokers	Twitch-DE
wvRN	0.037 ± 0.020	<u>0.147</u> ± 0.005	0.214 ± 0.040	<u>0.079</u> ± 0.036	<u>0.060</u> ± 0.011
CDQ	0.100 ± 0.045	0.409 ± 0.017	0.354 ± 0.098	0.358 ± 0.141	0.270 ± 0.067
ENQ	<u>0.029</u> ± 0.008	0.476 ± 0.001	<u>0.159</u> ± 0.007	0.099 ± 0.036	0.089 ± 0.036
XNQ	**0.015** ± 0.011	**0.015** ± 0.001	**0.034** ± 0.007	**0.032** ± 0.010	**0.040** ± 0.010
Improvement	− 48.27%	− 90.48%	− 84.11%	− 59.49%	− 33.33%

Reported results are the 5-fold CV AE averages on the test folds. Row "Improvement" reports the improvement (error reduction) of XNQ with respect to the second-best performance

the latter mostly balanced), the relative improvements are 84.11% and 33.33%. Even on the Cora-2 dataset, whose extremely high homophily should be congenial for the baselines, XNQ demonstrates a 48.27% error reduction with respect to the runner-up method. These results underscore XNQ's robust performance across graphs of varying sizes, training prevalence values, and heterophily levels. The Bayesian analysis reported in Appendix C confirms XNQ as the significantly best method for NQ.

For completeness, in Table 3 we also report the results of the evaluation in terms of *relative absolute error* (RAE). In general, the two measures may paint two different pictures, since it is often the case that method A is better than method B at AE while the opposite is true at RAE (Esuli et al., 2022). orEven using a different evaluation measure, XNQ manages to outperform all the other methods in almost all cases, the only exception being the Twitch-DE dataset, where it achieves the second-best result, with ENQ the best performer. The statistical significance analysis of Appendix C confirms this picture.

Figure 4 shows the diagonal plots on the four heterophilic datasets (Cora-2 is omitted as all methods have similarly looking diagonal plots). It can be noticed that while all baselines have an erratic behaviour depending on the dataset, XNQ consistently constantly maintains strong performance, regardless of the characteristics of the dataset to which it is being applied, approaching the ideal performance line that bisects the plotting space.

5.3 Ablation study

To gain deeper insights into XNQ's performance and to validate our design, we conduct ablation experiments by modifying the underlying components of XNQ, following the exact setup described in Sect. 5.1. Specifically, we study how performance is affected by varying the component of XNQ that computes the node embeddings, and the component corresponding to the downstream quantification method.

5.3.1 Node embedder ablation

We replace the original node embedder of XNQ with multiple layers of Graph Convolutional Network (GCN) (Kipf & Welling, 2017), Graph Attention Network (GAT) (Velickovic et al., 2018), and Graph Isomorphism Network (GIN) (Xu et al., 2019), with the number of layers treated as a hyper-parameter. These methods are briefly recapped in Appendix B. Following best practices to evaluate GNNs (Errica et al., 2020), we also include a network-agnostic variant model which applies a Logistic Regression (LR) readout directly to the node features without considering the graph structure. The results, shown in the upper block of Table 4, clearly indicate that XNQ exploits the graph structure better than the baselines,

Table 3 Results of the evaluation (best performance in boldface, second-best performance underlined)

Method	Cora-2	Genius	Questions	Tolokers	Twitch-DE
wvRN	0.976 ± 1.758	$\underline{2.605} \pm 0.539$	12.718 ± 7.930	6.151 ± 8.407	3.465 ± 3.070
CDQ	1.679 ± 1.101	18.304 ± 2.047	25.152 ± 2.702	20.150 ± 12.42	11.234 ± 4.352
ENQ	$\underline{0.751} \pm 0.987$	13.246 ± 7.278	$\underline{4.775} \pm 2.473$	$\underline{2.903} \pm 1.398$	$\mathbf{2.316} \pm 0.949$
XNQ	$\mathbf{0.241} \pm 0.374$	$\mathbf{0.417} \pm 0.025$	$\mathbf{2.161} \pm 1.020$	$\mathbf{0.441} \pm 0.253$	$\underline{3.254} \pm 1.508$
Improvement	-67.95%	-83.99%	-54.74%	-84.80%	–

Reported results are the 5-fold CV RAE averages on the test folds. The row "Improvement" reports the improvement (error reduction) of XNQ with respect to the second-best performance

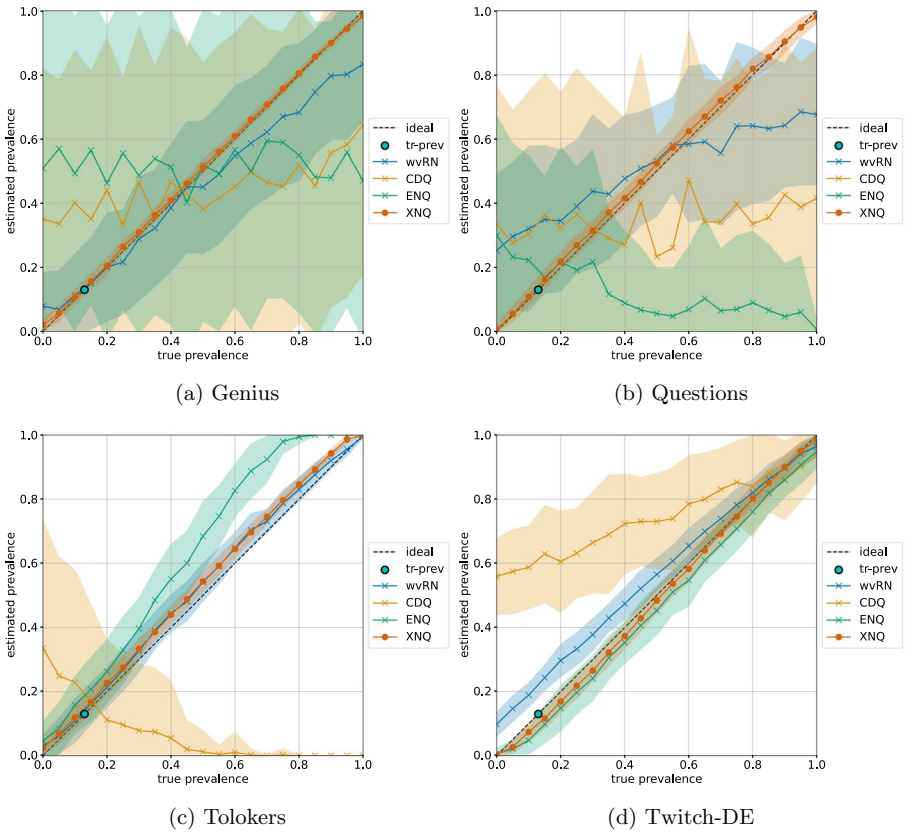


Fig. 4 Diagonal plots compare XNQ against other NQ baselines for different test class prevalence values generated via the APP. Shaded bands represent standard deviations

with superior performances. Surprisingly, the LR baseline performs better than the convolutional GNNs, which we attribute to the known issue of these models being difficult to calibrate properly (Teixeira et al., 2019), making them less suitable for quantification tasks.

5.3.2 Quantifier component ablation

We replace SLD with different aggregative quantification methods described in detail in Appendix A. According to the results in the bottom block of Table 4, XNQ achieves the best performance in 4 out of 5 cases and secures the second-best performance in the Genius dataset, with only a tiny margin separating it from the top methods (PACC, HDy, and DyS). This result confirms that CC is a completely inadequate method for NQ, which requires powerful downstream quantifiers. All the other results align with the existing literature on quantification, where SLD-based algorithms consistently rank among the top performers, thereby justifying our decision to integrate an EM quantifier into XNQ.

Table 4 Results of ablating the two main components of XNQ (best performance in boldface, second-best performance underlined)

Ablation	Method	Cora-2	Genius	Questions	Tolokers	Twitch-DE
Node	LR	.032 ± .023	.018 ± .001	.088 ± .013	.044 ± .011	.057 ± .010
	GCN	.071 ± .034	.050 ± .008	.106 ± .035	.057 ± .027	.045 ± .012
	GAT	.059 ± .023	.032 ± .004	.096 ± .041	.099 ± .111	.055 ± .016
	GIN	.075 ± .014	.111 ± .124	.107 ± .032	.051 ± ±.020	.064 ± .030
Quantifier	CC	.048 ± .011	.086 ± .002	.429 ± .010	.265 ± .008	.203 ± .007
	ACC	.022 ± .012	.017 ± .000	.108 ± .018	.045 ± .016	.056 ± .013
	PCC	.043 ± .012	.217 ± .000	.410 ± .009	.235 ± .002	.221 ± .004
	PACC	.026 ± .012	**.014** ± .000	.068 ± .021	.041 ± .010	.048 ± .011
	HDy	.037 ± .019	**.014** ± .000	.076 ± .043	.054 ± .030	.070 ± .012
	DyS	.023 ± .014	**.014** ± .000	.081 ± .039	**.032** ± .005	.044 ± .010
Reference	XNQ	**.015** ± .011	.015 ± .001	**.034** ± .007	**.032** ± .010	**.040** ± .010

Reported results are the 5-fold CV AE averages (\pm standard deviation) obtained on the test folds

The XNQ results are taken from Table 2

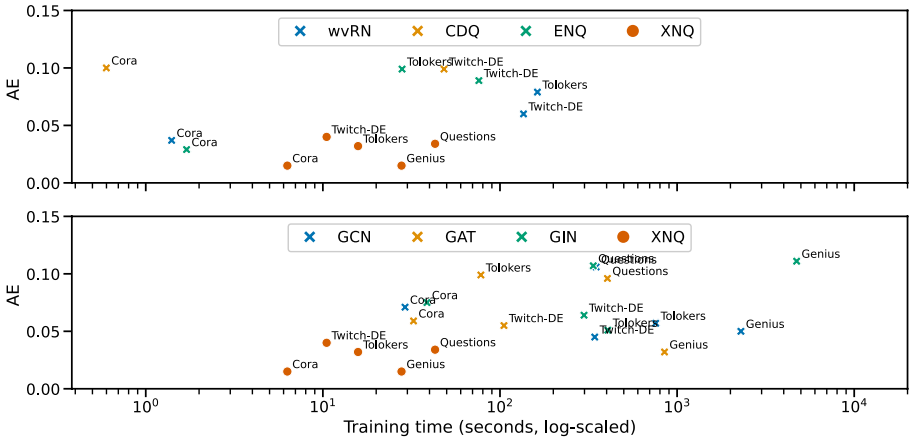


Fig. 5 The trade-off between error (AE, y axis) and average time (in seconds, log-scaled x axis) required to train the most expensive hyperparameter configuration of the models. Only results with AE ≤ 0.125 are shown to avoid cluttering. XNQ occupies the "sweet spot" close to the origin, where methods are both efficient and effective

5.4 Efficiency considerations

5.4.1 Computation time

XNQ turns out to be computationally more efficient than the other graph-learning methods. To support this claim, for each dataset we plot on the log-scaled y axis of Fig. 5 the average time (over 5 folds) required by each model to train the most expensive hyper-parameter configuration on a single NVidia V100 GPU. In all cases, XNQ is faster to train than the alternatives, up to more than one order of magnitude faster for the Genius dataset, taking less than a single minute to process the network's 400K nodes. Notice that while the base-

lines are faster than XNQ in the Cora-2 dataset (top figure), their AE is worse than the one obtained by XNQ.

5.4.2 Data annotation

A particularly onerous challenge in dealing with large-scale networks is providing enough annotated data to feed to the learning methods, as it often requires human intervention in real-world scenarios. Such an example may be users in a social network answering a preference survey, which is then used as the annotated data to quantify user preferences within particular communities. In Fig. 6, we show that the AE of XNQ on Genius (the largest network) stays low for increasingly smaller fractions of annotated training data (down to less than 2% of nodes), showcasing its scalability to scarcer annotated data scenarios.

5.5 Extension to the multi-class setting

To demonstrate the versatility of XNQ beyond the scope of binary quantification, we here present the results of an evaluation in a multi-class quantification setting. From an operational point of view, adapting from the binary to the multi-class scenario amounts to replacing the readout module with a multinomial LR classifier, and replacing the downstream quantifiers with their multi-class counterparts. Crucially, most quantification methods we have discussed are natively multi-class,[2] making the extension trivial. We thus conducted an evaluation on the following datasets:

- *Cora-Multi*: this is the same Cora dataset we described in Sect. 5.1.1, but this time we handle all 7 classes simultaneously. The class-adjusted homophily is 0.77.
- *Amazon*: a product co-purchasing network where nodes are products, and edges connect co-occurring purchases (Platonov et al., 2023). The task is to predict the average rating (4 classes) given to a product by reviewers. The graph has 29,492 nodes (with 300 features each) and 182,100 edges, and a class-adjusted homophily of 0.13.
- *Flickr*: a graph where nodes are images uploaded to the Flickr online community. Edges connect nodes where respective images share some common properties such as same geographic location (McAuley & Leskovec, 2012). Node features are 500-dimensional bag-of-word vectors extracted from image annotations. The task is to classify each image into a semantic category (one of 7 classes). The dataset has 89,250 nodes and

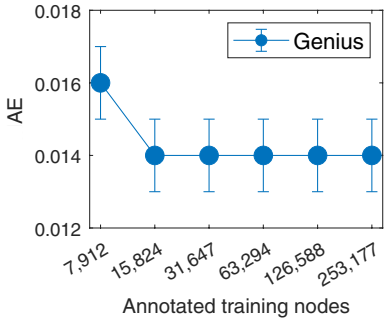


Fig. 6 The quantification error of XNQ remains consistently low on Genius as the fraction of annotated data is reduced up to $\approx 2\%$

[2] The only exceptions are DyS and HDy, although multi-class extensions have been proposed in the literature.

Table 5 Results of the evaluation on multi-class datasets (best performance in boldface, second-best performance underlined)

Method	Cora-Multi	Amazon	Flickr
wvRN	$\underline{0.023} \pm 0.006$	$\underline{0.108} \pm 0.017$	0.201 ± 0.066
CDQ	0.076 ± 0.015	0.111 ± 0.003	0.145 ± 0.001
ENQ	0.028 ± 0.004	0.116 ± 0.016	$\underline{0.106} \pm 0.006$
XNQ	$\mathbf{0.012} \pm 0.002$	$\mathbf{0.065} \pm 0.007$	$\mathbf{0.075} \pm 0.006$
Improvement	-47.82%	-39.81%	-29.24%

Reported results are the 5-fold CV AE averages on the test folds. The row "Improvement" reports the improvement (error reduction) of XNQ with respect to the second-best performance

Table 6 Results of the evaluation on multi-class datasets (best performance in boldface, second-best performance underlined)

Method	Cora-Multi	Amazon	Flickr
wvRN	$\underline{0.417} \pm 0.095$	$\underline{1.424} \pm 0.104$	$\underline{1.710} \pm 0.112$
CDQ	1.189 ± 0.210	1.576 ± 0.360	2.986 ± 0.521
ENQ	0.459 ± 0.067	1.521 ± 0.188	1.884 ± 0.193
XNQ	$\mathbf{0.312} \pm 0.025$	$\mathbf{1.330} \pm 0.273$	$\mathbf{1.588} \pm 0.082$
Improvement	-25.17%	-6.60%	-7.13%

Reported results are the 5-fold CV RAE averages on the test folds. The row "Improvement" reports the improvement (error reduction) of XNQ with respect to the second-best performance

899,756 edges, with a class-adjusted homophily of 0.07.

We compare XNQ to the same baselines as in Table 2, using the same experimental setup described in Sect. 5. The results are presented in Table 5 (for the AE metric) and Table 6 (for the RAE metric): similarly to what observed in the binary quantification setup, XNQ consistently outperforms all the existing baselines in the three datasets. These results underscore the effectiveness of XNQ and broaden its applicability to NQ tasks having node label cardinality > 2. A Bayesian analysis to assess the significance of the results is presented in Appendix C, confirming the above trend.

5.6 Application scenarios

In many real-world scenarios, users are not interested in classifying individual items, but in estimating the prevalence of some characteristics in a subset of the general population. In this section, we present some application scenarios where XNQ can be employed to address some real-world tasks.

5.6.1 Quantifying bots and malicious users in social networks

Social networks have become central to modern communication, shaping public discourse and influencing societal trends. However, they also present significant challenges, including the spread of misinformation (Del Vicario et al., 2016), the formation of echo chambers (Cinelli et al., 2021), and the manipulation of online narratives through automated bots (Stella et al., 2019). The proliferation of echo chambers and automated bots on social media has significantly altered the landscape of information dissemination, leading to widespread misinformation and polarization. Echo chambers arise when individuals interact primar-

ily within networks that reinforce their pre-existing beliefs, limiting exposure to diverse perspectives and reducing critical engagement with opposing viewpoints. Social media algorithms exacerbate this phenomenon by curating content that aligns with user preferences, ultimately reinforcing biases and facilitating the spread of misinformation. Simultaneously, the deployment of bots—automated accounts programmed to manipulate online discourse—amplifies deceptive narratives by artificially boosting engagement and creating false consensus. These bots can be strategically employed to distort public opinion, influence elections, and undermine trust in credible sources (Stella et al., 2019). The combined effects of these mechanisms contribute to a fragmented and distorted information ecosystem, weakening democratic processes and informed decision-making. Such tactics may be employed as a concerted effort by a malicious actor, posing a serious security threat (Hellman, 2024). In Fig. 7 we present an example of a social network containing real users and bot accounts. Both types of accounts may interact and form relationships with each other. In addition, accounts may follow certain social pages that publish posts and other content that users can comment on and share with their contacts. Quantifying the fraction of bot accounts that subscribe to a social page can be an indication of whether the interactions are artificially inflated to construct an echo chamber by a malicious actor. In our example, Page B presents an unusual fraction of bot accounts compared to other social pages such as Page A. XNQ can enable social media to efficiently identify such pages or user groups, resulting in increased effectiveness in content moderation. This application scenario represents a slight variation of the tasks of Sect. 5, which mostly have dealt with quantifying classes of users in social networks.

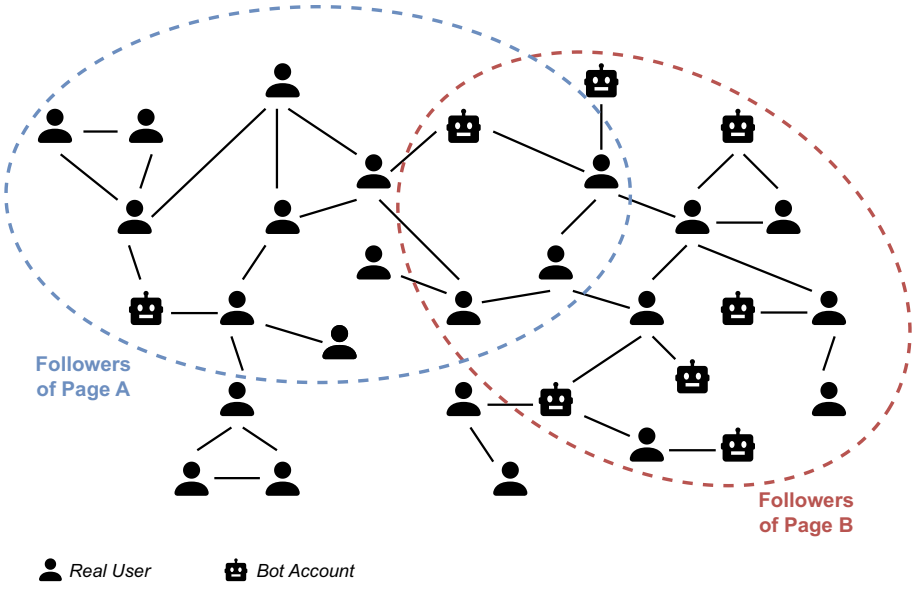


Fig. 7 Quantifying bots among the followers of a social network page. An anomalous prevalence of bots may be an indication that Page B has been artificially constructed to be an echo chamber for the spread of misinformation

5.6.2 Network quantification in the social sciences and political science

As recalled in the introduction, quantification finds many applications in disciplines, such as political science, the social sciences, and epidemiology, that have an inherent focus on populations, rather than on individuals. For instance, in political science, a researcher may want to predict, for an upcoming election, the prevalence of votes for the different parties (say, Democratic vs. Republican) among the population of a certain nation, and how these prevalence values vary geographically (e.g., across different municipalities, provinces, and regions of the country of interest). Here, individuals belonging to the population may each be represented by a set of covariates assumed to predict the phenomenon under study. The above inference can straightforwardly be obtained via quantification at the desired level of granularity, e.g., quantifying on a specific province or on a specific municipality of that province. An interesting observation is that (see Fig. 8 for an example)

the distribution of electoral votes tends to vary *smoothly* in space. In other words, the distributions that characterize two neighbouring municipalities are unlikely to be radically different, and their difference is likely to be smaller than the difference between two far-away municipalities (the same holds at province level, region level, etc.). This suggests that NQ may be more effective than standard quantification, once geographic proximity (i.e., the relation "x and y are neighbouring geographical units") is represented as an undirected graph. (In this case, the resulting graph is likely characterized by a high degree of homophily.)

5.6.3 Network quantification in epidemiology

A similar case can be made for the geographic distribution of many other phenomena, be they of interest for political science, the social sciences, epidemiology, etc. For example, when studying the spread of infectious diseases (e.g., HIV, or COVID-19), both social and geographical proximity are core factors (Kuchler et al., 2022; Rothenberg et al., 2005). In this scenario, estimating the fraction of infected people within a certain geographical area may be achieved via quantification, and NQ techniques may deliver more accurate results than standard quantification techniques by leveraging, as in the case of election result prediction, proximity information. As noted above, here the notion of "proximity" may be different from the previous case; e.g., two towns may be considered "neighbour-

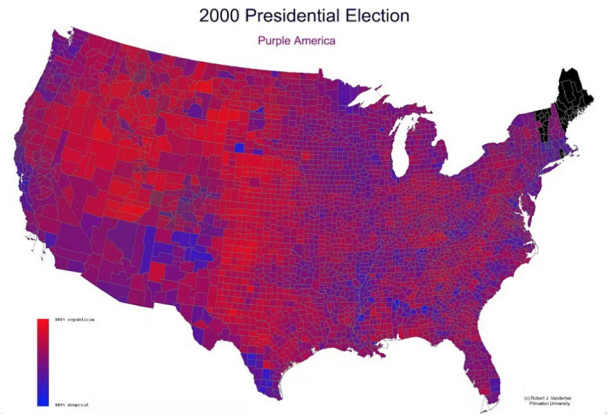


Fig. 8 2000 Presidential Election results in the US, at county level. (Image from https://tinyurl.com/2nxy8rzw, © Robert Vanderbei, Princeton University; reprinted with permission of the author.)

ing geographical units" not only if they lie next to each other, but also if a high-frequency flight connects them. Estimating the prevalence of infections in different geographic areas or within specific groups is extremely useful for planning an effective response to the health crisis and for the informed decision-making process concerning health policies (Madhav et al., 2017). An accurate quantification of the infected populations can improve: the resource allocation of medical supplies and personnel; the containment measures, such as lockdowns and travel restrictions; and the planning of vaccination campaigns.

6 Conclusions

We have presented XNQ, a novel model tailored to the many challenges of NQ, which integrates randomized recursive graph neural networks, a customized calibrated readout for quantification, and a downstream powerful quantifier based on the EM algorithm. Our extensive evaluation shows that XNQ improves at this task by effectively exploiting the graph structure of the data while scaling seamlessly to hundreds of thousands of nodes without being impaired by common issues of large-scale networks such as heterophily. These results place XNQ at the forefront of NQ research and pave the way for its application to further real-world case studies. In future research, we plan to extend our approach to the multi-class case, and to investigate *non-aggregative* NQ methods, i.e., methods that estimate class priors without assigning labels to (or computing posterior probabilities for) individual nodes. Unlike the aggregative methods, non-aggregative ones have the additional advantage that no inference at the individual level is performed; this is desirable in applications such as measuring the fairness (i.e., absence of bias) of a model with respect to sensitive attributes (Fabris et al., 2023).

Appendix A: Quantification methods

Given a hard classifier $\mathfrak{h}$ and a sample set S, *Classify and Count (CC)* is defined as:

$$\hat{p}_S^{\mathrm{CC}}(\oplus) = \frac{1}{|S|} \sum_{\boldsymbol{x}_v \in S} \mathbb{1}\left[\mathfrak{h}(\boldsymbol{x}_v) = \oplus\right] \tag{A1}$$

i.e., the prevalence of class $\oplus$ is estimated as the number of times it is predicted by $\mathfrak{h}$ in S divided by the number of samples in S. *Adjusted Classify and Count (ACC)* attempts to correct the estimates returned by CC. It is defined as:

$$\hat{p}_S^{\mathrm{ACC}}(\oplus) = \frac{\hat{p}_S^{\mathrm{CC}}(\oplus) - \hat{\mathrm{fpr}}_{\mathfrak{h}}}{\hat{\mathrm{tpr}}_{\mathfrak{h}} - \hat{\mathrm{fpr}}_{\mathfrak{h}}} \tag{A2}$$

where $\hat{\mathrm{tpr}}_{\mathfrak{h}}$ and $\hat{\mathrm{fpr}}_{\mathfrak{h}}$ are estimates of the true positive and false positive rates obtained by hold-out or k-fold cross-validation on the training set $\mathcal{V}_{\mathrm{labelled}}$. *Probabilistic Classify and Count (PCC)* is a probabilistic counterpart of CC, which replaces the hard estimates with expected counts computed from the posterior probabilities of a calibrated probabilistic classifier $\mathfrak{s}$:

$$\hat{p}_S^{\mathrm{PCC}}(\oplus) = \frac{1}{|S|}\sum_{\boldsymbol{x}_v \in S} \Pr(Y = \oplus | X = \boldsymbol{x}_v) = \frac{1}{|S|}\sum_{\boldsymbol{x}_v \in S} \mathfrak{s}(\boldsymbol{x}_v) \tag{A3}$$

Similarly, *Probabilistic Adjusted Classify and Count (PACC)* is a probabilistic counterpart of ACC where the CC estimate is replaced with the PCC estimate, with $\hat{\mathrm{tpr}}_{\mathfrak{s}}$ and $\hat{\mathrm{fpr}}_{\mathfrak{s}}$ as probabilistic counterparts of $\hat{\mathrm{tpr}}_{\mathfrak{h}}$ and $\hat{\mathrm{fpr}}_{\mathfrak{h}}$, respectively:

$$\hat{p}_S^{\mathrm{PACC}}(\oplus) = \frac{\hat{p}_S^{\mathrm{PCC}}(\oplus) - \hat{\mathrm{fpr}}_{\mathfrak{s}}}{\hat{\mathrm{tpr}}_{\mathfrak{s}} - \hat{\mathrm{fpr}}_{\mathfrak{s}}} \tag{A4}$$

HDy is a probabilistic binary quantification method that views quantification as the problem of minimizing the divergence (measured in terms of the Hellinger Distance) between two distributions of posterior probabilities returned by a soft classifier $\mathfrak{s}$, one coming from the unlabelled examples and the other coming from a validation set. HDy relies on histograms for modelling the class-conditional distributions of the validation posteriors coming from positive instances ($H_\oplus$), and those coming from negative instances ($H_\ominus$). HDy then looks for the mixture parameter α that yields the best match between the validation distribution (the mixture $\alpha H_\oplus + (1-\alpha)H_\ominus$) and the test distribution (a histogram of the test posteriors), returning α as the estimated prevalence for class $\oplus$. The *DyS* method is basically HDy with the Topsøe distance used in place of the Hellinger distance.

Appendix B: Convolutional graph neural networks

In the ablation experiments, we use different convolutional GNNs to study their performances in comparison with XNQ. In Table 7, we briefly describe them for completeness, using the notation developed in Sect. 2. We use $1 \leq \ell \leq L$ to indicate the number of convolutional layers, $\boldsymbol{h}_v^{(\ell)}$ to denote the embedding of node v computed at the ℓ^{th} layer, and $\boldsymbol{W}^{(\ell)}$ as the ℓ^{th} matrix of learnable weights specific for each layer. In the table, $\tilde{\mathcal{N}}(v) = \mathcal{N}(v) \cup \{v\}$ is the closed neighbourhood of v, α_{ij} are attention coefficients computed by comparing pairs of neighbouring embeddings, ϵ is a small learnable constant and MLP is a multilayer perceptron.

Table 7 Message passing variants used in the ablation studies

Message passing	$\boldsymbol{h}_v^{(\ell)}$
GCN	$\mathrm{sigmoid}\left(\boldsymbol{W}^{(\ell)} \sum_{u \in \tilde{\mathcal{N}}(v)} \frac{1}{\lvert\tilde{\mathcal{N}}(v)\rvert} \boldsymbol{h}_u^{(\ell 1)}\right)$
GAT	$\mathrm{ReLU}\left(\alpha_{vv}\boldsymbol{W}^{(\ell)}\boldsymbol{h}_v^{(\ell-1)} + \sum_{u \in \mathcal{N}(v)} \alpha_{vu}\boldsymbol{W}^{(\ell)}\boldsymbol{h}_u^{(\ell-1)}\right)$
GIN	$\mathrm{MLP}\left((1+\epsilon)\,\boldsymbol{h}_v^{(\ell-1)} + \sum_{u \in \mathcal{N}(v)} \boldsymbol{h}_v^{(\ell-1)}\right)$

Appendix C: Statistical significance

In order to assess whether the differences in performance between our proposed XNQ and the alternative methods are statistically significant, we compare the different models to XNQ across datasets using Bayesian analysis, following the work of Benavoli et al. (2017).

The test first computes the posterior distribution of the differences in performance between the two models on multiple datasets. Then, it draws samples from the posterior distribution to estimate the number of times one model outperforms the other, or how many times they perform equivalently (where equivalence is established if performances fall in a predefined region around the posterior mean, called *rope*). Here, we set the rope to the AE/RAE standard deviation of XNQ across the different datasets.

The results are presented in Table 8. From the table, it is possible to see that, irrespective of the model compared to XNQ, the posterior probability that the alternative method would perform better than XNQ is always 0 (except the XNQ-ENQ comparison with RAE on the binary datasets, where however the probability that ENQ is better than XNQ is only of 3.2%). There are some situations where ties are likely, though: for example, if the downstream quantifier is ACC rather than SLD, there is approximately a 50% chance of a performance tie. However, we argue in favor of SLD as our quantifier of choice, which is able to achieve top-tier performance with solid theoretical guarantees (in particular, Fisher consistency).

Table 8 Bayesian test comparison of XNQ's performances versus the different baselines and ablation models used in this study

XNQ vs.	p(XNQ is worse)	p(XNQ is equal)	p(XNQ is better)
Baselines on binary datasets (Tables 2, 3)			
wvRN	0.000 (0.000)	0.010 (0.001)	0.990 (0.999)
CDQ	0.000 (0.000)	0.003 (0.001)	0.997 (0.999)
ENQ	0.000 (0.032)	0.001 (0.001)	0.999 (0.967)
Baselines on multi-class datasets (Tables 5, 6)			
wvRN	0.000 (0.000)	0.395 (0.009)	0.605 (0.991)
CDQ	0.000 (0.000)	0.092 (0.143)	0.908 (0.857)
ENQ	0.000 (0.000)	0.844 (0.038)	0.156 (0.962)
Node embedder ablation (Table 4)			
LR	0.000	0.259	0.741
GCN	0.000	0.015	0.985
GAT	0.000	0.009	0.991
GIN	0.000	0.003	0.997
Quantifier ablation (Table 4)			
CC	0.000	0.001	0.999
ACC	0.000	0.541	0.449
PCC	0.000	0.001	0.999
PACC	0.000	0.789	0.211
HDy	0.000	0.141	0.859
DyS	0.000	0.789	0.211

Each row expresses the posterior probability that XNQ performs better than the competitor. Highest probability is underlined. Results using RAE as main metric are shown within parentheses

Author contributions A.Micheli and F.S. contributed to the study conception and design. Material preparation, data collection and analysis were performed by W.S., D.T., A.Moreo, and M.P. The initial manuscript was written by W.S., using feedback from all the other authors. All authors have contributed to the revision of the initial manuscript, and have read and approved the final version.

Funding This work was partially supported by: Project DEEP-GRAPH, funded by the Italian Ministry of University and Research (MUR) PRIN 2022 (project code: 2022YLRBTT, CUP: I53C24002440006); project PNRR, PE00000013, "FAIR - Future Artificial Intelligence Research", Spoke 1, funded by European Commission under NextGeneration EU programme (CUP B53D22000980006); project "Quantification under Dataset Shift" (QuaDaSh), funded by the European Union under the NextGenerationEU funding scheme (CUP B53D23026250001); Project PAN-HUB, funded by the Italian Ministry of Health (POS 2014–2020, project ID: T4-AN-07, CUP: I53C22001300001).

Data availability Data used in this study are publicly available. The Cora dataset is available at https://github.com/kimiyoung/planetoid/tree/master/data/. The Genius dataset is available at https://github.com/CUAI/Non-Homophily-Large-Scale/raw/master/data/. The Questions dataset is available at https://github.com/yandex-research/heterophilous-graphs/raw/main/data/. The Tolokers dataset is available at https://github.com/yandex-research/heterophilous-graphs/raw/main/data/. The Twitch dataset is available at http://graphmining.ai/datasets/ptg/twitch/. The Amazon dataset is available at https://github.com/yandex-research/heterophilous-graphs/raw/main/data. The Flickr dataset is downloadable through the Python Geometric library in Python.

Code availability The code is available at https://github.com/marcopodda/network-quantification.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Ethical approval and consent to participate Not applicable.

Consent for publication Not applicable.

References

Alexandari, A., Kundaje, A., & Shrikumar, A. (2020). Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In *Proceedings of the 37th international conference on machine learning (ICML 2020)*, Virtual Event (pp. 222–232).

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis. *Journal of Machine Learning Research, 18*(77), 1–36.

Bacciu, D., Errica, F., Micheli, A., & Podda, M. (2020). A gentle introduction to deep learning for graphs. *Neural Networks, 129*, 203–221. https://doi.org/10.1016/j.neunet.2020.06.006

Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M. J. (2010). Quantification via probability estimators. In *Proceedings of the 11th IEEE international conference on data mining (ICDM 2010)* (pp. 737–742). https://doi.org/10.1109/icdm.2010.75

Cinelli, M., De Francisci Morales, G., Galeazzi, A., Quattrociocchi, W., & Starnini, M. (2021). The echo chamber effect on social media. *Proceedings of the National Academy of Sciences of the United States of America, 118*(9), 2023301118. https://doi.org/10.1073/pnas.2023301118

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., & Sun, X. (2020). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the 34th AAAI conference on artificial intelligence (AAAI 2020)*, New York (pp. 3438–3445). https://doi.org/10.1609/AAAI.V34I04.5747

Del Vicario, M., Bessi, A., Zollo, F., Petroni, F., Scala, A., Caldarelli, G., Stanley, H. E., & Quattrociocchi, W. (2016). The spreading of misinformation online. *Proceedings of the National Academy of Sciences of the United States of America, 113*(3), 554–559. https://doi.org/10.1073/pnas.1517441113

Esuli, A., Fabris, A., Moreo, A., & Sebastiani, F. (2023). *Learning to quantify*. Springer. https://doi.org/10.1007/978-3-031-20467-8

Errica, F., Podda, M., Bacciu, D., & Micheli, A. (2020). A fair comparison of graph neural networks for graph classification. In *Proceedings of the 8th international conference on learning representations (ICLR 2020)*, Addis Ababa, ET.

Esuli, A., Moreo, A., Sebastiani, F., & Sperduti, G. (2022). A detailed overview of LeQua 2022: Learning to quantify. In *Working notes of the 13th conference and labs of the evaluation forum (CLEF 2022)*.

Fabris, A., Esuli, A., Moreo, A., & Sebastiani, F. (2023). Measuring fairness under unawareness of sensitive attributes: A quantification-based approach. *Journal of Artificial Intelligence Research, 76*, 1117–1180. https://doi.org/10.1613/jair.1.14033

Forman, G. (2008). Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery, 17*(2), 164–206. https://doi.org/10.1007/s10618-008-0097-y

Gallicchio, C., & Micheli, A. (2010). Graph echo state networks. In *Proceedings of the 2010 international joint conference on neural networks (IJCNN 2010)*, Barcelona, ES (pp. 3967–3974). https://doi.org/10.1109/IJCNN.2010.5596796

Gallicchio, C., & Micheli, A. (2020). Fast and deep graph neural networks. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020), New York, US, 3898–3905. https://doi.org/10.1609/AAAI.V34I04.5803

González, P., Castaño, A., Chawla, N. V., & Coz, J. J. (2017). A review on quantification learning. *ACM Computing Surveys, 50*(5), 1–40. https://doi.org/10.1145/3117807

González, P., Moreo, A., & Sebastiani, F. (2024). Binary quantification and dataset shift: An experimental investigation. *Data Mining and Knowledge Discovery, 38*(4), 1670–1712. https://doi.org/10.1007/s10618-024-01014-1

González-Castro, V., Alaiz-Rodríguez, R., & Alegre, E. (2013). Class distribution estimation based on the Hellinger distance. *Information Sciences, 218*, 146–164. https://doi.org/10.1016/j.ins.2012.05.028

Hellman, M. (2024). Disinformation as a security problem. In *Security, disinformation and harmful narratives: RT and Sputnik News coverage about Sweden* (Chap. 1, pp. 1–27). https://doi.org/10.1007/978-3-031-58747-4_1

Jaenich, T., Moreo, A., Fabris, A., McDonald, G., Esuli, A., Ounis, I., & Sebastiani, F. (2025). Quantifying query fairness under unawareness. arXiv:2506.04140 [cs.IR]

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the international conference on learning representations (ICLR 2017)*, Toulon, FR.

Kuchler, T., Russel, D., & Stroebel, J. (2022). JUE insight: The geographic spread of COVID-19 correlates with the structure of social networks as measured by Facebook. *Journal of Urban Economics, 127*, Article 103314. https://doi.org/10.1016/j.jue.2020.103314

Likhobaba, D., Pavlichenko, N., & Ustalov, D. (2023). Toloker graph: Interaction of crowd annotators. https://zenodo.org/records/7620796. https://doi.org/10.5281/zenodo.7620795

Lim, D., Hohne, F., Li, X., Huang, S. L., Gupta, V., Bhalerao, O., & Lim, S.-N. (2021). Large-scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Proceedings of the 35th conference on neural information processing systems (NeurIPS 2021)*, Virtual Event (pp. 20887–20902).

Lipton, Z. C., Wang, Y., & Smola, A. J. (2018). Detecting and correcting for label shift with black box predictors. In *Proceedings of the 35th international conference on machine learning (ICML 2018)*, Stockholm, SE (pp. 3128–3136).

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review, 3*(3), 127–149. https://doi.org/10.1016/j.cosrev.2009.03.005

Macskassy, S. A., & Provost, F. (2003). A simple relational classifier. In *Proceedings of the SIGKDD multi-relational data mining workshop (MRDM 2003)*, Washington.

Madhav, N., Oppenheim, B., Gallivan, M., Mulembakani, P., Rubin, E., & Wolfe, N. (2017). Pandemics: risks, impacts, and mitigation. In *Disease control priorities*: Improving health and reducing poverty (Vol. 9, 3rd ed., Chap. 17, pp. 315–345). The World Bank. https://doi.org/10.1596/978-1-4648-0527-1_ch17

Maletzke, A., Reis, D., Cherman, E., & Batista, G (2019). DyS: A framework for mixture models in quantification. In (*Proceedings of the 33rd AAAI conference on artificial intelligence (AAAI 2019)* (pp. 4552–4560). https://doi.org/10.1609/aaai.v33i01.33014552

McAuley, J., & Leskovec, J. (2012). Image labeling on a network: Using social-network metadata for image classification. In *Proceedings of the 12th European conference on computer vision (ECCV 2012)*, Firenze, IT (pp. 828–841). https://doi.org/10.1007/978-3-642-33765-9_59

McCallum, A. K., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of Internet portals with machine learning. *Information Retrieval, 3*(2), 127–163. https://doi.org/10.1023/a:1009953814988

Micheli, A. (2009). Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks, 20*(3), 498–511. https://doi.org/10.1109/TNN.2008.2010350

Micheli, A., & Tortorella, D. (2023). Addressing heterophily in node classification with graph echo state networks. *Neurocomputing, 550*, Article 126506. https://doi.org/10.1016/j.neucom.2023.126506

Milli, L., Monreale, A., Rossetti, G., Pedreschi, D., Giannotti, F., & Sebastiani, F. (2015). Quantification in social networks. In *Proceedings of the 2nd IEEE international conference on data science and advanced analytics (DSAA 2015)*, Paris, FR (pp. 596–605). https://doi.org/10.1109/dsaa.2015.7344845

Moreo, A. (2025). On the interconnections of calibration, quantification, and classifier accuracy prediction under dataset shift. arXiv:2505.11380 [cs.LG]

Nakajima, K., & Fischer, I. (Eds.). (2021). *Reservoir computing: Theory, physical implementations, and applications*. Springer.

Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., & Prokhorenkova, L. (2023). A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *Proceedings of the 11th international conference on learning representations (ICLR 2023)*, Kigali, RW.

Platt, J. C. (2000). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in large margin classifiers* (pp. 61–74). MIT.

Rozemberczki, B., Allen, C., & Sarkar, R. (2021). Multi-scale attributed node embedding. *Journal of Complex Networks*. https://doi.org/10.1093/comnet/cnab014

Rothenberg, R., Muth, S. Q., Malone, S., Potterat, J. J., & Woodhouse, D. E. (2005). Social and geographic distance in HIV risk. *Sexually Transmitted Diseases, 32*(8), 506–512. https://doi.org/10.1097/01.olq.0000161191.12026.ca

Saerens, M., Latinne, P., & Decaestecker, C. (2002). Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation, 14*(1), 21–41. https://doi.org/10.1162/089976602753284446

Sebastiani, F. (2020). Evaluation measures for quantification: An axiomatic approach. *Information Retrieval Journal, 23*(3), 255–288. https://doi.org/10.1007/s10791-019-09363-y

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks, 20*(1), 61–80. https://doi.org/10.1109/TNN.2008.2005605

Stella, M., Cristoforetti, M., & De Domenico, M. (2019). Influence of augmented humans in online interactions during voting events. *PLOS ONE, 14*(5), 0214210. https://doi.org/10.1371/journal.pone.0214210

Storkey, A. (2009). When training and test sets are different: Characterizing learning transfer. In J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, & N. D. Lawrence (Eds.), *Dataset shift in machine learning* (pp. 3–28). MIT.

Tang, L., Gao, H., & Liu, H. (2010). Network quantification despite biased labels. In *Proceedings of the 8th workshop on mining and learning with graphs (MLG 2010)* (pp. 147–154). https://doi.org/10.1145/1830252.1830271

Tasche, D. (2017). Fisher consistency for prior probability shift. *Journal of Machine Learning Research, 18*, 95–19532.

Teixeira, L., Jalaian, B., & Ribeiro, B. (2019). Are graph neural networks miscalibrated? In (*Proceedings of the ICML workshop on learning and reasoning with graph-structured representations*, Long Beach, USA.

Tortorella, D., Gallicchio, C., & Micheli, A. (2022). Spectral bounds for graph echo state network stability. In *Proceedings of the 2022 international joint conference on neural networks (IJCNN 2022)*, Padova, IT. https://doi.org/10.1109/IJCNN55064.2022.9892102

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y. (2018). Graph attention networks. In: *Proceedings of the 6th international conference on learning representations (ICLR 2018)*, Vancouver, CA

Volpi, L., Moreo, A., Sebastiani, F. (2024). A simple method for classifier accuracy prediction under prior probability shift. In *Proceedings of the 27th international conference on discovery science (DS 2024)*, Pisa, IT (pp. 267–283). https://doi.org/10.1007/978-3-031-78980-9_17

Volpi, L., Moreo, A., & Sebastiani, F. (2025). QuAcc: Using quantification to predict classifier accuracy under prior probability shift. *Intelligenza Artificiale* (forthcoming). https://doi.org/10.1177/17248035251338347

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems, 32*(1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In *Proceedings of the 7th international conference on learning representations (ICLR 2019)*, New Orleans, USA.

Yan, Y., Hashemi, M., Swersky, K., Yang, Y., & Koutra, D. (2022). Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *Proceedings of the 22nd international conference on data mining (ICDM 2022)*, Orlando, USA (pp. 1287–1292). https://doi.org/10.1109/ICDM54844.2022.00169

Yang, Z., Cohen, W. W., & Salakhutdinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd international conference on machine learning (iCML 2016)*, New York, USA (pp. 40–48).

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., & Koutra, D. (2020). Beyond homophily in graph neural networks: Current limitations and effective designs. In *Proceedings of the 34th conference on neural information processing systems (NeurIPS 2020)*, Virtual Event (pp. 7793–7804).

Authors and Affiliations

Alessio Micheli[1] · Alejandro Moreo[2] · Marco Podda[1] · Fabrizio Sebastiani[2] · William Simoni[1] · Domenico Tortorella[1]

✉ Domenico Tortorella
domenico.tortorella@unipi.it

Alessio Micheli
alessio.micheli@unipi.it

Alejandro Moreo
alejandro.moreo@isti.cnr.it

Marco Podda
marco.podda@unipi.it

Fabrizio Sebastiani
fabrizio.sebastiani@isti.cnr.it

William Simoni
wilsimoni@gmail.com

[1] Dipartimento di Informatica, Università di Pisa, Largo Bruno Pontecorvo, 3, 56127 Pisa, Italy

[2] Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Via Giuseppe Moruzzi 1, 56124 Pisa, Italy