Unlearning Graph Classifiers with Limited Data Resources

Chao Pan*
University of Illinois, Urbana-Champaign
Urbana, Illinois, USA
chaopan2@illinois.edu

Eli Chien*
University of Illinois, Urbana-Champaign
Urbana, Illinois, USA
ichien3@illinois.edu

Olgica Milenkovic
University of Illinois, Urbana-Champaign
Urbana, Illinois, USA
milenkov@illinois.edu

ABSTRACT

As the demand for user privacy grows, controlled data removal (machine unlearning) is becoming an important feature of machine learning models for data-sensitive Web applications such as social networks and recommender systems. Nevertheless, at this point it is still largely unknown how to perform efficient machine unlearning of graph neural networks (GNNs); this is especially the case when the number of training samples is small, in which case unlearning can seriously compromise the performance of the model. To address this issue, we initiate the study of unlearning the Graph Scattering Transform (GST), a mathematical framework that is efficient, provably stable under feature or graph topology perturbations, and offers graph classification performance comparable to that of GNNs. Our main contribution is the first known nonlinear approximate graph unlearning method based on GSTs. Our second contribution is a theoretical analysis of the computational complexity of the proposed unlearning mechanism, which is hard to replicate for deep neural networks. Our third contribution are extensive simulation results which show that, compared to complete retraining of GNNs after each removal request, the new GST-based approach offers, on average, a 10.38x speed-up and leads to a 2.6% increase in test accuracy during unlearning of 90 out of 100 training graphs from the IMDB dataset (10% training ratio). Our implementation is available online at https://doi.org/10.5281/zenodo.7613150.

CCS CONCEPTS

• **Security and privacy → Human and societal aspects of security and privacy**; • **Computing methodologies → Machine learning**.

KEYWORDS

Graph neural networks, graph scattering transforms, graph unlearning, machine unlearning, small data sample regime

ACM Reference Format:
Chao Pan, Eli Chien, and Olgica Milenkovic. 2023. Unlearning Graph Classifiers with Limited Data Resources. In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3543507.3583547

*Both authors contributed equally to this research.

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583547

1 INTRODUCTION

Graph classification is a learning task that frequently arises in real-world Web applications related to social network analysis [16, 31, 35, 42], recommendation system development [57, 60], medical studies [38, 40], and drug design [13, 23]. While the availability of large sets of user training data has contributed to the deployment of modern deep learning models – such as Graph Neural Networks (GNNs) – for solving graph classification problems, deep learners mostly fail to comply with new data privacy regulations. Among them is the right of users to remove their data from the dataset and eliminate their contribution to all models trained on it. Such unlearning regulations[1] ensure the *Right to be Forgotten,* and have to be taken into consideration during the model training process. We consider for the first time the problem of removing nodes from possibly different training graphs used in graph classification models. Examples where such requests arise include users (nodes) withdrawing from some interest groups (subgraphs) in social networks, and users deleting their browsing histories for online shopping, which corresponds to the removal of an entire graph from the training set. Note that only removing user data from a dataset is insufficient to guarantee the desired level of privacy, since the training data can be potentially reconstructed from the trained models [19, 53]. Naively, one can always retrain the model from scratch with the remaining dataset to ensure complete privacy; however, this is either computationally expensive or infeasible due to the high frequency of data removal requests [24]. Another possible approach is to use differential privacy based methods to accommodate data removal during training. However, most state-of-the-art DP-GNNs focus on node classification instead of graph classification tasks, and it remains an open problem to design efficient GNNs for graph classification problem while preserving node-level privacy. Recently, the data removal problem has also motivated a new line of research on *machine unlearning* [5], which aims to efficiently eliminate the influence of certain data points on a model. While various kinds of unlearning algorithms exist, most of the existing literature, especially the one pertaining to deep neural networks, does not discuss model training and unlearning in the limited training data regime; still, this is a very important data regime for machine learning areas such as few-shot learning [49, 54]. Unlearning in this case may degrade the model performance drastically, and as a result, efficient

[1]Existing laws on user data privacy include the European Union General Data Protection Regulation (GDPR), the California Consumer Privacy Act (CCPA), the Canadian Consumer Privacy Protection Act (CPPA), Brazilian General Data Protection Law (LGPD), and many others.

and accurate graph classification and unlearning with limited training data is still an open problem. Throughout this paper, we use "data removal" and "unlearning" interchangeably.

Concurrently, a mathematical approach known as the Graph Scattering Transform (GST) has been used as a nontrainable counterpart of GNNs that can be applied to any type of trainable classification. GST iteratively applies graph wavelets and nonlinear activations on the input graph signals which may be viewed as forward passes of GNNs without trainable parameters. It has been shown that GSTs can not only remain stable under small perturbations of graph features and topologies [21], but also compete with many types of GNNs on solving different graph classification problems [20, 22, 32, 45]. Furthermore, since all wavelet coefficients in GSTs are constructed analytically, GST is computationally more efficient and requires less training data compared to GNNs when combined with the same trainable classifiers. As a result, GSTs are frequently used in practice [2, 36, 41, 45], especially in the limited training data regime. Furthermore, it also allows for deriving theoretical analysis on nonlinear models for graph embedding, and the related conclusions could potentially shed some light on the design of GNNs that are more suitable for data removals.

Our contributions. We introduce the first *nonlinear* graph learning framework based on GSTs that accommodates an *approximate* unlearning mechanism with provable performance guarantees. Here, "approximate" refers to the fact that unlearning is not exact (as it would be for completely retrained models) but more akin to the parametrized notion of differential privacy [15, 28, 50] (see Section 3 for more details). With the adoption of GSTs, we show that our nonlinear framework enables provable data removal (similar results are currently only available for linear models [8, 9, 28]) and provides theoretical unlearning complexity guarantees. These two problems are hard to tackle for deep neural network models like GNNs [58]. Our experimental results reveal that when trained with only 10% samples in the dataset, our GST-based approach offers, on average, a 10.38x speed-up and leads to a 2.6% increase in test accuracy during unlearning of 90 out of 100 training graphs from the IMDB dataset, compared to complete retraining of a standard GNN baseline, Graph Isomorphism Network (GIN) [58], after each removal request. Moreover, we also show that nonlinearities consistently improve the model performance on all five tested datasets, with an average increase of 3.5% in test accuracy, which emphasizes the importance of analysis on nonlinear graph learning models.

2 RELATED WORKS

Machine unlearning. The concept of machine unlearning was introduced in [5]. Two unlearning criteria have been considered in the literature: *Exact* and *approximate unlearning*. Exact unlearning requires eliminating all information pertaining to the removed data, so that the unlearned model parameters have exactly the same "distribution" as the ones obtained by retraining from scratch. Examples of exact unlearning methods include sharding-based techniques [3] and quantization-based methods specialized for k-means clustering problems [24, 46]. On the other hand, approximate unlearning only requires the distribution of the unlearned model parameters to be similar to retraining from scratch. One recent work [28] introduced a probabilistic definition of unlearning motivated by differential privacy (DP) [15], while [50] performs a theoretical study of generalizations of unlearning and [26] addresses heuristic methods for deep learning models. Approximate unlearning offers a trade-off between model performance and privacy, as one is allowed to retain more relevant information compared to complete retraining so as not to cause serious performance degradation. This makes it especially desirable for limited training data regimes.

Graph unlearning and DP-GNNs. Only a handful of works have taken the initial steps towards machine unlearning of graphs. [7] proposes a sharding-based method for exact graph unlearning, while [8, 9] introduces approximate graph unlearning methods that come with theoretical (certified) guarantees. However, these works only focus on node classification tasks and are in general not directly applicable to graph classification. Moreover, the latter method [9] only considers linear model while the work described herein includes nonlinear graph transformations (i.e., GSTs). Note that although DP-GNNs [10, 47] can also be used to achieve graph unlearning, they not only focus on node classification, but also require a high "privacy cost" to unlearn even *one single node or edge* without causing significant performance degradation [9]. Similar observations regarding the relationship between DP and approximate unlearning were made in the context of unstructured unlearning [28, 50]. Only one recent line of work considered differential privacy for graph classification [44], but the edit distance defined therein relates to the entire training graph instead of one or multiple nodes within the training graph. This approach hence significantly differs from our proposed data removal approach on graph classification tasks, and a more detailed comparative analysis is available in Section 3.

Scattering transforms. Convolutional neural networks (CNNs) use nonlinearities coupled with trained filter coefficients and are well-known to be hard to analyze theoretically [1]. As an alternative, [4, 39] introduced *scattering transforms*, nontrainable versions of CNNs. Under certain conditions, scattering transforms offer excellent performance for image classification tasks and more importantly, they are analytically tractable. The idea of using specialized transforms has percolated to the graph neural network domain as well [20, 22, 45, 61]. Specifically, the graph scattering transform (GST) proposed in [20] iteratively performs graph filtering and applies element-wise nonlinear activation functions to input graph signals to obtain embeddings of graphs. It is computationally efficient compared to GNNs, and performs better than standard Graph Fourier Transform (GFT) [48] due to the adoption of nonlinearities.

Few-shot learning. Few-shot learning [54] is a machine learning paradigm for tackling the problem of learning from a limited number of samples with supervised information. It was first proposed in the context of computer vision [17], and has been successfully applied to many other areas including graph neural networks [49] and social network analysis [37]. At this point, it has not been addressed under the umbrella of unlearning, as in this case unlearning can drastically degrade the model performance. Consequently, efficient and accurate graph classification and unlearning within the few-shot learning setting remains an open problem.

3 PRELIMINARIES

We reserve bold-font capital letters such as $\mathbf{X}$ for matrices, bold-font lowercase letters such as $\mathbf{x}$ for vectors, and calligraphic capital

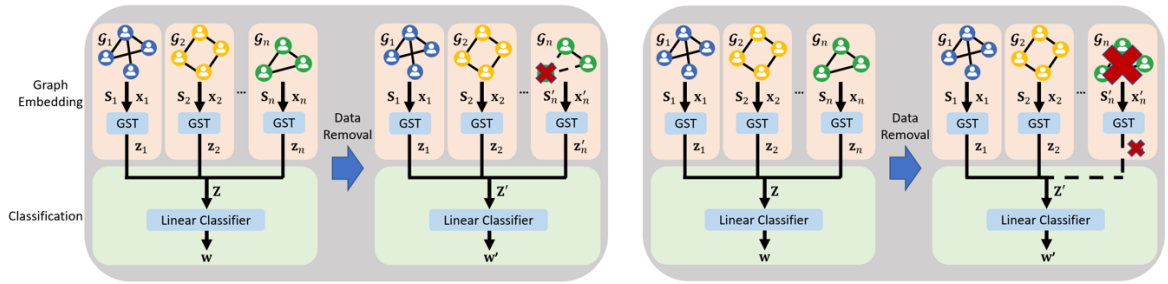

Figure 1: The difference between removing one node from one training graph and completely removing one training graph, in the context of graph classification and when the *graph embedding procedure* is nontrainable. Without loss of generality, we assume that the removal requests arise in the training graph $\mathcal{G}_n$. (a) The embedding of $\mathcal{G}_n$ changes from $\mathbf{z}_n$ to $\mathbf{z}'_n$ and is used to train the new classifier $\mathbf{w}'$. (b) The embedding of $\mathcal{G}_n$ is no longer used for training, and the underlying unlearning problem reduces to the one studied in [28]. Since GST is a nontrainable feature extractor, the embeddings of all other graphs $\mathbf{z}_i, i \neq n$ remain the same for both (a) and (b).

letters $\mathcal{G}_i = (V_i, E_i)$ for graphs, where the index $i \in [n]$ stands for the i-th training graph with vertex set V_i and edge set E_i. Furthermore, $[\mathbf{x}]_i$ denotes the i-th element of vector $\mathbf{x}$. The norm $\|\cdot\|$ is by default the ℓ_2 norm for vectors and the operator norm for matrices. We consider the graph classification problem as in [58], where we have n graphs with possibly different sizes for training (e.g., each graph $\mathcal{G}_i$ may represent a subgraph within the Facebook social networks). For a graph $\mathcal{G}_i$, the "graph shift" operator is denoted as $\mathbf{S}_i$. Graph shift operators include the graph adjacency matrix, graph Laplacian matrix, their normalized counterparts and others. For simplicity, this work focuses on $\mathbf{S}_i$ being the symmetric graph adjacency matrices. The graph node features are summarized in $\mathbf{X}_i \in \mathbb{R}^{g_i \times d_i}$, where $g_i = |V_i|$ denotes the number of nodes in $\mathcal{G}_i$ and d_i the corresponding feature vector dimension. We also assume that $d_i = 1$ for $\forall i \in [n]$ so that $\mathbf{X}_i$ reduces to a vector $\mathbf{x}_i$.

For each $\mathcal{G}_i$, we obtain an embedding vector $\mathbf{z}_i$ either via nontrainable, graph-based transforms such as PageRank [25], GFT [51] and GST [21], or geometric deep learning methods like GNNs [29, 55, 58]. For GST, the length of $\mathbf{z}_i \in \mathbb{R}^d$ is determined by the number of nodes in the L-layer J-ary scattering tree; specifically, $d = \sum_{l=0}^{L-1} J^l, J, L \in \mathbb{N}^+$. All $\mathbf{z}_i$ are stacked vertically to form the data matrix $\mathbf{Z}$, which is used with the labels $\mathbf{y}$ to form the dataset $\mathcal{D} = (\mathbf{Z}, \mathbf{y})$ for training a linear graph classifier. The loss equals

$$L(\mathbf{w}, \mathcal{D}) = \sum_{i=1}^{n} \left(\ell(\mathbf{w}^T \mathbf{z}_i, y_i) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \right), \tag{1}$$

where $\ell(\mathbf{w}^T \mathbf{z}, y)$ is a convex loss function that is differentiable everywhere. We also write $\mathbf{w}^\star = \text{argmin}_{\mathbf{w}} L(\mathbf{w}, \mathcal{D})$, and observe that the optimizer is unique whenever $\lambda > 0$ due to strong convexity. For simplicity, we only consider the binary classification problem and point out that multiclass classification can be solved using well-known one-versus-all strategies.

Graph embedding via GST. GST is a convolutional architecture including three basic units: 1) a bank of multiresolution wavelets $\{h_j\}_{j=1}^J$ (see Appendix E for a list of common choices of graph wavelets); 2) a pointwise nonlinear activation function ρ (i.e., the absolute value function); 3) A low-pass operator U (i.e., averaging operator $\frac{1}{g_i}\mathbf{1}$). Note that the graph wavelets used in GST should form a frame [30, 52] such that there exist $A \leq B$ and $\forall \mathbf{x}$, we have

$$A^2 \|\mathbf{x}\|^2 \leq \sum_{j=1}^{J} \left\| \mathbf{H}_j(\mathbf{S}) \mathbf{x} \right\|^2 \leq B^2 \|\mathbf{x}\|^2, \tag{2}$$

where the graph shift operator $\mathbf{S}$ has the eigenvalue decomposition $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ and $\mathbf{H}_j(\mathbf{S}) = \mathbf{V}\mathbf{H}_j(\mathbf{\Lambda})\mathbf{V}^T = \mathbf{V}\text{diag}[h_j(\lambda_1), \ldots, h_j(\lambda_n)]\mathbf{V}^T$. These elements are combined sequentially to generate a vector representation $\Phi(\mathbf{S}, \mathbf{x})$ of the input graph signal $\mathbf{x}$, as shown in Figure 2. More specifically, the original signal $\mathbf{x}$ is positioned as the root node (0-th layer), and then J wavelets are applied to each of the nodes from the previous layer, generating J^l new nodes in the l-th layer to which the nonlinearity ρ is applied. The scalar scattering coefficient $\phi_{p_j(l)}(\mathbf{S}, \mathbf{x})$ of node $p_j(l)$ is obtained by applying U to the filtered signal $\Phi_{p_j(l)}(\mathbf{S}, \mathbf{x}) = \mathbf{H}_{p_j(l)}(\mathbf{S})\mathbf{x}$, where the path $p_j(l) = (j_1, \ldots, j_l)$ denotes the corresponding node in the scattering tree. The coefficients are concatenated to form the overall representation of the graph $\Phi(\mathbf{S}, \mathbf{x})$, which is then used as the embedding of the graph $\mathbf{z}$. For a scattering transform with L layers, the length of $\Phi(\mathbf{S}, \mathbf{x})$ is $d = \sum_{l=0}^{L-1} J^l$, which is independent of the number of nodes in the graph. For a more detailed description of GSTs, the reader is referred to [21].

Removal requests. The unlearning problem considered is depicted in Figure 1(a). Without loss of generality, we assume that the removal request arises for the g_n-th node in the training graph $\mathcal{G}_n$. We consider one removal request at a time, pertaining either to the removal of node features of one node ($\mathbf{x}_n \to \mathbf{x}'_n$) or to the removal of the entire node ($\mathbf{S}_n \to \mathbf{S}'_n, \mathbf{x}_n \to \mathbf{x}'_n$) from one training graph (instead of the removal of one whole graph). In this case, the number of (training) graphs will remain the same (n), and we need to investigate the influence of the node removal on the graph embedding process (i.e., GST) to determine if removal significantly changes the embeddings of the affected graphs. Note that since GST is a nontrainable feature extractor, the embeddings of all unaffected training graphs will remain the same. On the other hand, if we are asked to unlearn the entire graph, the unlearning problem (see Figure 1(b)) reduces to the one studied in [28].

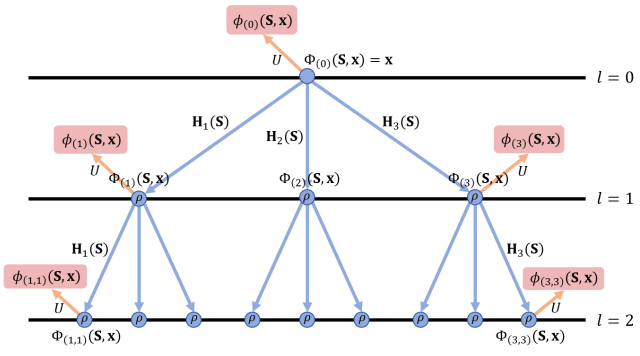

Figure 2: The embedding procedure of GST with $L=J=3$. The scalar scattering coefficients $\phi_{p_j(l)}(\mathrm{S},\mathbf{x})$ (red blocks) are concatenated to form the vector representation z of a graph.

Certified approximate removal. Let A be a (randomized) learning algorithm that trains on $\mathcal{D}$, the set of data points before removal, and outputs a model $h \in \mathcal{H}$, where $\mathcal{H}$ represents a chosen space of models. The data removal requests leads to a change from $\mathcal{D}$ to $\mathcal{D}'$. Given a pair of parameters (ϵ,δ), an unlearning algorithm M applied to $A(\mathcal{D})$ is said to guarantee an (ϵ,δ)-certified approximate removal for A, where $\epsilon,\delta>0$ and $\mathcal{X}$ denotes the space of possible datasets, if $\forall \mathcal{T}\subseteq\mathcal{H}, \mathcal{D}\subseteq\mathcal{X}$:

$$\mathbb{P}\left(M(A(\mathcal{D}),\mathcal{D},\mathcal{D}')\in\mathcal{T}\right)\leq e^{\epsilon}\mathbb{P}\left(A(\mathcal{D}')\in\mathcal{T}\right)+\delta,$$
$$\mathbb{P}\left(A(\mathcal{D}')\in\mathcal{T}\right)\leq e^{\epsilon}\mathbb{P}\left(M(A(\mathcal{D}),\mathcal{D},\mathcal{D}')\in\mathcal{T}\right)+\delta. \quad (3)$$

This definition is related to (ϵ,δ)-DP [15] except that we are now allowed to update the model based on the updated dataset $\mathcal{D}'$ under the certified approximate removal criterion. An (ϵ,δ)-certified approximate removal method guarantees that the updated model $M(A(\mathcal{D}),\mathcal{D},\mathcal{D}')$ is approximately the same from a probabilistic point of view as the model $A(\mathcal{D}')$ obtained by retraining from scratch on $\mathcal{D}'$. Thus, any information about the removed data is approximately (but with provable guarantees) eliminated from the model. Note that exact removal corresponds to $(0,0)$-certified approximate removal. Ideally, we would like to design M so that it satisfies Equation (3) with a predefined (ϵ,δ) pair and has a complexity that is significantly smaller than that of complete retraining.

Nonlinearities in the graph learning framework. Previous work [9] on analyzing approximate unlearning of linearized GNNs focuses on node classification tasks within a single training graph. There, the training samples are node embeddings and the embeddings are correlated because of the graph convolution operation. The analysis of node classification proved to be challenging since propagation on graphs "mixes" node features, and thus the removal of even one feature/edge/node could lead to the change of embeddings for multiple nodes. The same technical difficulty exists for GNNs tackling graph classification tasks (graph embeddings are correlated), which limits the scope of theoretical studies of unlearning to GNNs that do not use nonlinear activations, such as SGC [56]. Meanwhile, if we use nontrainable graph feature extractors (i.e., GSTs) for graph classification tasks to obtain the graph embeddings, the removal requests arising for one graph will not affect the embeddings of other graphs. This feature of GSTs not only makes the analysis of approximate unlearning tractable, but also allows one to introduce nonlinearities into the graph embedding procedure (i.e., the nonlinear activation function used in GSTs), which significantly improves the model performance in practice.

4 UNLEARNING GRAPH CLASSIFIERS

We now turn our attention to describing how the inherent stability properties of GSTs, which are nonlinear graph embedding methods, can aid in approximate unlearning without frequent retraining (a detailed discussion regarding how to replace GST with GNNs is available in Section 5).

Motivated by the unlearning approach described in [28] for unstructured data, we design an unlearning mechanism M that updates the trained model from $\mathbf{w}^\star$ to $\mathbf{w}'$, the latter of which represents an approximation of the unique optimizer of $L(\mathbf{w},\mathcal{D}')$. Denote the Hessian of $L(\cdot,\mathcal{D}')$ at $\mathbf{w}^\star$ as $\mathbf{H}_{\mathbf{w}^\star}=\nabla^2 L(\mathbf{w}^\star,\mathcal{D}')$ and denote the gradient difference by $\Delta=\nabla L(\mathbf{w}^\star,\mathcal{D})-\nabla L(\mathbf{w}^\star,\mathcal{D}')$. The update rule is $\mathbf{w}'=\mathbf{w}^\star+\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta$, which can be intuitively understood as follows. Our goal is to achieve $\nabla L(\mathbf{w}',\mathcal{D}')=0$ for the updated model. Using a Taylor series expansion we have

$$\nabla L(\mathbf{w}',\mathcal{D}')\approx\nabla L(\mathbf{w}^\star,\mathcal{D}')+\nabla^2 L(\mathbf{w}^\star,\mathcal{D}')(\mathbf{w}'-\mathbf{w}^\star)=0.$$

Therefore, we have

$$\mathbf{w}'-\mathbf{w}^\star=\left[\nabla^2 L(\mathbf{w}^\star,\mathcal{D}')\right]^{-1}\left[0-\nabla L(\mathbf{w}^\star,\mathcal{D}')\right]$$
$$\mathbf{w}'=\mathbf{w}^\star+\left[\nabla^2 L(\mathbf{w}^\star,\mathcal{D}')\right]^{-1}\left[\nabla L(\mathbf{w}^\star,\mathcal{D})-\nabla L(\mathbf{w}^\star,\mathcal{D}')\right]. \quad (4)$$

The last equality holds due to the fact that $\nabla L(\mathbf{w}^\star,\mathcal{D})=0$. When $\nabla L(\mathbf{w}',\mathcal{D}')=0$, $\mathbf{w}'$ is the unique optimizer of $L(\cdot,\mathcal{D}')$ due to strong convexity. If $\nabla L(\mathbf{w}',\mathcal{D}')\neq 0$, some amount of information about the removed data point remains. One can show that the gradient residual norm $\|\nabla L(\mathbf{w}',\mathcal{D}')\|$ determines the error of $\mathbf{w}'$ when used to approximate the true minimizer of $L(\cdot,\mathcal{D}')$ again via Taylor series expansion.

We would like to point out that the update approach from [28] originally designed for unstructured unlearning can be viewed as a special case of Equation (4) when the graph $\mathcal{G}_n$ needs to be unlearned completely. In this case, we have $\Delta=\nabla L(\mathbf{w}^\star,\mathcal{D})-\nabla L(\mathbf{w}^\star,\mathcal{D}')=\lambda\mathbf{w}^\star+\nabla\ell((\mathbf{w}^\star)^T\mathbf{z}_n,y_n)$, which is the same expression as the one used in [28]. However, when we only need to unlearn part of the nodes in $\mathcal{G}_n$, Δ becomes $\Delta=\nabla\ell((\mathbf{w}^\star)^T\mathbf{z}_n,y_n)-\nabla\ell((\mathbf{w}^\star)^T\mathbf{z}_n',y_n)$, where $\mathbf{z}_n'$ is obtained via GST computed on the remaining nodes in $\mathcal{G}_n$. The unlearning mechanism shown in Equation (4) can help us deal with different types of removal requests within a unified framework, and the main analytical contribution of our work is to establish bounds of the gradient residual norm for the generalized approach in the context of graph classification.

As discussed above, Equation (4) is designed to minimize the gradient residual norm $\|\nabla L(\mathbf{w}',\mathcal{D}')\|$. However, the direction of the gradient residual $L(\mathbf{w}',\mathcal{D}')$ may leak information about the training sample that was removed, which violates the goal of approximate unlearning. To address this issue, [28] proposed to hide the real gradient residual by adding a linear noise term $\mathbf{b}^T\mathbf{w}$ to the training loss, a technique known as loss perturbation [6]. When taking the derivative of the noisy loss, the random noise $\mathbf{b}$ is supposed to "mask" the true value of the gradient residual so that one cannot infer information about the removed data. The corresponding

approximate unlearning guarantees for the proposed unlearning mechanism can be established by leveraging Theorem 4.1 below.

Theorem 4.1 (Theorem 3 from [28]). *Denote the noisy training loss by* $L_{\mathbf{b}}(\mathbf{w}, \mathcal{D}) = \sum_{i=1}^{n} \left(\ell(\mathbf{w}^T \mathbf{z}_i, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \right) + \mathbf{b}^T \mathbf{w}$, *and let A be the learning algorithm that returns the unique optimum of* $L_{\mathbf{b}}(\mathbf{w}, \mathcal{D})$. *Suppose that* $\mathbf{w}'$ *is obtained by the unlearning procedure M and that* $\|\nabla L(\mathbf{w}', \mathcal{D}')\| \leq \epsilon'$ *for some computable bound* $\epsilon' > 0$. *If* $\mathbf{b} \sim \mathcal{N}(0, c\epsilon'/\epsilon)^d$ *is normally distributed with some constant* $\epsilon, c > 0$, *then M satisfies Equation (3) with* (ϵ, δ) *for algorithm A applied to* $\mathcal{D}'$, *where* $\delta = 1.5 \cdot e^{-c^2/2}$.

Hence, if we can appropriately bound the gradient residual norm $\|\nabla L(\mathbf{w}', \mathcal{D}')\|$ for graph classification problems, we can show that the unlearning mechanism ensures an (ϵ, δ)-certified approximate removal. For the analysis, we need the following assumptions on the loss function ℓ. These assumptions naturally hold for commonly used linear classifiers such as linear regression and logistic regression (see Section 5).

Assumption 4.2. *There exist constants* $C_1, C_2, \gamma_1, \gamma_2$ *such that for* $\forall i \in [n]$ *and* $\mathbf{w} \in \mathbb{R}^d$: *1)* $\|\nabla \ell(\mathbf{w}^T \mathbf{z}_i, y_i)\| \leq C_1$; *2)* $|\ell'(\mathbf{w}^T \mathbf{z}_i, y_i)| \leq C_2$; *3)* ℓ' *is* γ_1*-Lipschitz; 4)* ℓ'' *is* γ_2*-Lipschitz; 5) it is always possible to rescale* $\mathbf{x}_i$ *for graph* $\mathcal{G}_i$ *so that* $|[\mathbf{x}_i]_j| \leq 1, \forall j \in [g_i]$.

We show next that the gradient residual norm for graph classification can be bounded for both types of removal requests.

Theorem 4.3. *Suppose that Assumptions 4.2 hold, and that the difference between the original dataset* $\mathcal{D} = (\mathbf{Z}, \mathbf{y})$ *and the updated dataset* $\mathcal{D}' = (\mathbf{Z}', \mathbf{y})$ *is in the embedding of the n-th training graph, which equals* $\mathbf{z}'_n = \Phi(\mathbf{S}_n, \mathbf{x}'_n)$ *with* $[\mathbf{x}'_n]_{g_n} = 0$. *Let B be the frame constant for the graph wavelets used in GST (see Equation (2)). Then*

$$\|\nabla L(\mathbf{w}', \mathcal{D}')\| \leq \frac{\gamma_2 F^3}{\lambda^2 n} \min\left\{4C_1^2, \frac{(\gamma_1 C_1 F^2 + \lambda C_2 F)^2}{\lambda^2 g_n}\right\}, \quad (5)$$

where $F = \sqrt{\sum_{l=0}^{L-1} B^{2l}}$. *For tight energy preserving wavelets we have* $B = 1, F = \sqrt{L}$.

The proof of Theorem 4.3 can be found in Appendix A. The key idea is to use the stability property of GSTs, as we can view the change of graph signal from $\mathbf{x}_n$ to $\mathbf{x}'_n$ as a form of signal perturbation. The stability property ensures that the new embedding $\Phi(\mathbf{S}_n, \mathbf{x}'_n)$ does not deviate significantly from the original one $\Phi(\mathbf{S}_n, \mathbf{x}_n)$, and allows us to establish the upper bound on the norm of gradient residual. Note that the second term on the RHS in Equation (5) decreases as the size g_n of the graph $\mathcal{G}_n$ increases. This is due to the averaging operator U used in GSTs, and thus the graph embedding is expected to be more stable under signal perturbations for large rather than small graphs.

Next, we consider a more common scenario where an entire node in $\mathcal{G}_n$ needs to be unlearned. This type of request frequently arises in graph classification problems for social networks, where unlearning one node corresponds to one user withdrawing from one or multiple social groups. In this case, we have the following bound on the gradient residual norm.

Theorem 4.4. *Suppose that Assumptions 4.2 hold, and that both the features and all edges incident to the* g_n*-th node in* $\mathcal{G}_n$ *have to be unlearned. Then*

$$\|\nabla L(\mathbf{w}', \mathcal{D}')\| \leq \frac{4\gamma_2 C_1^2 F^3}{\lambda^2 n}, F = \sqrt{\sum_{l=0}^{L-1} B^{2l}}. \quad (6)$$

Remark. *In this case, the norm* $\|\mathbf{z}'_n - \mathbf{z}_n\|$ *capturing the change in the graph embeddings obtained via GST is proportional to the norm of the entire graph signal* $\|\mathbf{x}_n\|$. *The second term within the* min *function is independent on* g_n *and likely to be significantly larger than the first term. Thus, we omit the second term in Equation (6). More details are available in Appendix B.*

Batch removal. The update rule in Equation (4) naturally supports removing multiple nodes from possibly different graphs at the same time. We assume that the number of removal requests m at one time instance is smaller than the minimum size of a training graph, i.e., $m < \min_i g_i$, to exclude the trivial case of unlearning an entire graph. In this setting, we have the following upper bound on the gradient residual norm, as described in Corollary 4.5 and 4.6. The proofs are delegated to Appendix C.

Corollary 4.5. *Suppose that Assumptions 4.2 hold, and that m nodes from n graphs have requested feature removal. Then*

$$\|\nabla L(\mathbf{w}', \mathcal{D}')\| \leq \frac{\gamma_2 m^2 F^3}{\lambda^2 n} \min\left\{4C_1^2, \frac{(\gamma_1 C_1 F^2 + \lambda C_2 F)^2}{\lambda^2 g_n}\right\}, \quad (7)$$

where $F = \sqrt{\sum_{l=0}^{L-1} B^{2l}}$.

Corollary 4.6. *Suppose that Assumptions 4.2 hold, and that m nodes from n graphs have requested entire node removal. Then*

$$\|\nabla L(\mathbf{w}', \mathcal{D}')\| \leq \frac{4\gamma_2 m^2 C_1^2 F^3}{\lambda^2 n}, F = \sqrt{\sum_{l=0}^{L-1} B^{2l}}. \quad (8)$$

Data-dependent bounds. The upper bounds in Theorems 4.3 and 4.4 contain a constant factor $1/\lambda^2$ which may be large when λ is small and n is moderate. This issue arises due to the fact that those bounds correspond to the worst case setting for the gradient residual norm. Following an approach suggested in [28], we also investigated data-dependent gradient residual norm bounds which can be efficiently computed and are much tighter than the worst-case bound. Note that these are the bounds we use in the online unlearning procedure of Algorithm 2 for simulation purposes.

Theorem 4.7. *Suppose that Assumptions 4.2 hold. For both single and batch removal setting, and for both feature and node removal requests, one has*

$$\left\|\nabla L\left(\mathbf{w}', \mathcal{D}'\right)\right\| \leq \gamma_2 F \left\|\mathbf{Z}'\right\| \left\|\mathbf{H}_{\mathbf{w}^\star}^{-1} \Delta\right\| \left\|\mathbf{Z}' \mathbf{H}_{\mathbf{w}^\star}^{-1} \Delta\right\|, \quad (9)$$

where $\mathbf{Z}'$ *is the data matrix corresponding to the updated dataset* $\mathcal{D}'$.

Algorithmic details. The pseudo-codes for training unlearning models, as well as the single node removal procedure are described below. During training, a random linear term $\mathbf{b}^T \mathbf{w}$ is added to the training loss. The choice of standard deviation α determines the privacy budget $\alpha\epsilon/\sqrt{2\log(1.5/\delta)}$ that is used in Algorithm 2. During unlearning, β tracks the accumulated gradient residual norm. If it exceeds the budget, then (ϵ, δ)-certified approximate removal for M is no longer guaranteed. In this case, we completely retrain the model using the updated dataset $\mathcal{D}'$.

Algorithm 1 Training Procedure

input: Training dataset $\mathcal{D} = \{\mathbf{z}_i \in \mathbb{R}^d, y_i\}_{i=1}^n$, loss ℓ, parameters $\alpha, \lambda > 0$.
Sample the noise vector $\mathbf{b} \sim \mathcal{N}(0, \alpha^2)^d$.
$\mathbf{w}^\star = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^n \left(\ell(\mathbf{w}^T\mathbf{z}_i, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|^2\right) + \mathbf{b}^T\mathbf{w}$.
return $\mathbf{w}^\star$.

Algorithm 2 Unlearning Procedure

input: Training graphs $\mathcal{G}_i$ with features $\mathbf{x}_i$, graph shift operator $\mathbf{S}_i$ and label y_i, loss ℓ, removal requests $\mathcal{R}_m = \{r_1, r_2, \ldots\}$, parameters $\epsilon, \delta, \gamma_2, \alpha, \lambda, F > 0$.
Compute the graph embeddings $\mathbf{z}_i$ via $\mathrm{GST}(\mathbf{x}_i, \mathbf{S}_i)$. Initiate the training set $\mathcal{D} = \{\mathbf{z}_i \in \mathbb{R}^d, y_i\}_{i=1}^n$.
Compute $\mathbf{w}$ using Algorithm 1 $(\mathcal{D}, \ell, \alpha, \lambda)$.
Set the accumulated gradient residual norm to $\beta = 0$.
for $r \in \mathcal{R}_m$ **do**
  In $\mathbf{x}'_r$, set the feature of a node to be removed to 0. Update $\mathbf{S}'_r$ if the entire node is to be removed.
  Compute the new graph embedding $\mathbf{z}'_r$ with $\mathrm{GST}(\mathbf{x}'_r, \mathbf{S}'_r)$.
  Update the training set $\mathcal{D}'$ and $\mathbf{Z}'$.
  Compute $\Delta = \nabla L(\mathbf{w}, \mathcal{D}) - \nabla L(\mathbf{w}, \mathcal{D}')$, $\mathbf{H} = \nabla^2 L(\mathbf{w}, \mathcal{D}')$.
  Update the accumulated gradient residual norm as $\beta = \beta + \gamma_2 F\|\mathbf{Z}'\|\|\mathbf{H}^{-1}\Delta\|\|\mathbf{Z}'\mathbf{H}^{-1}\Delta\|$.
  if $\beta > \alpha\epsilon/\sqrt{2\log(1.5/\delta)}$ **then**
    Recompute $\mathbf{w}$ using Algorithm 1 $(\mathcal{D}', \ell, \alpha, \lambda)$, $\beta = 0$.
  else
    $\mathbf{w} = \mathbf{w} + \mathbf{H}^{-1}\Delta$.
  end if
  $\mathcal{D} = \mathcal{D}'$.
end for
return $\mathbf{w}$.

5 DISCUSSION

Commonly used loss functions. For linear regression, the loss function is $\ell(\mathbf{w}^T\mathbf{z}_i, y_i) = (\mathbf{w}^T\mathbf{z}_i - y_i)^2$, while $\nabla^2\ell(\mathbf{w}^T\mathbf{z}_i, y_i) = \mathbf{z}_i\mathbf{z}_i^T$, which does not depend on $\mathbf{w}$. Therefore, it is possible to have $\|\nabla L(\mathbf{w}', \mathcal{D}')\| = 0$ based on the proof in Appendix A. This observation implies that our unlearning procedure M is a $(0, 0)$-certified approximate removal method when linear regression is used as the linear classifier module. Thus, the exact values of $C_1, C_2, \gamma_1, \gamma_2$ are irrelevant for the performance guarantees for M.

For binary logistic regression, the loss function is defined as $\ell(\mathbf{w}^T\mathbf{z}_i, y_i) = -\log(\sigma(y_i\mathbf{w}^T\mathbf{z}_i))$, where $\sigma(x) = 1/(1+\exp(-x))$ denotes the sigmoid function. As shown in [28], the assumptions (1) and (4) in 4.2 are satisfied with $C_1 = 1$ and $\gamma_2 = 1/4$. We only need to show that (2) and (3) of 4.2 hold as well. Observe that $\ell'(x, y) = \sigma(xy) - 1$. Since the sigmoid function $\sigma(\cdot)$ is restricted to lie in $[0, 1]$, $|\ell'|$ is bounded by 1, which means that our loss satisfies (2) in 4.2 with $C_2 = 1$. Based on the Mean Value Theorem, one can show that $\sigma(x)$ is $\max_{x\in\mathbb{R}}|\sigma'(x)|$-Lipschitz. With some simple algebra, one can also prove that $\sigma'(x) = \sigma(x)(1-\sigma(x)) \Rightarrow \max_{x\in\mathbb{R}}|\sigma'(x)| = 1/4$. Thus the loss satisfies assumption (3) in 4.2 as well, with $\gamma_1 = 1/4$. For multiclass logistic regression, one can adapt the one-versus-all strategy which leads to the same result.

Note that it is also possible to use other loss functions such as linear SVM in Equation (1) with regularization. Choosing appropriate loss function for different applications could be another interesting future direction of this work.

Reducing the complexity of recomputing graph embeddings. Assume that the removal request arises in graph $\mathcal{G}_n$. For the case of feature removal, since we do not need to update the graph shift operator $\mathbf{S}_n$, we can reuse the graph wavelets $\{\mathbf{H}_j(\mathbf{S}_n)\}_{j=1}^J$ computed before the removal to obtain the new embedding $\mathbf{z}'_n$, which is with complexity $O(dg_n^2)$.

For the case of complete node removal, we do need to update the graph wavelets $\{\mathbf{H}_j(\mathbf{S}'_n)\}_{j=1}^J$ based on the updated $\mathbf{S}'_n$. In general, the complexity of computing $\mathbf{z}'_n$ in this case equals $O(dg_n^3)$, as we need to compute the eigenvalue decomposition of $\mathbf{S}'_n$ and perform matrix multiplications multiple times. This computational cost may be too high when the size g_n of $\mathcal{G}_n$ is large. There are multiple methods to reduce this computational complexity, which we describe next.

If the wavelet kernel function $h(\lambda)$ is a polynomial function, we can avoid the computation of the eigenvalue decomposition of $\mathbf{S}'_n$ by storing the values of all $\{\mathbf{S}_n^k\}_{k=1}^K$ in advance during initial training, where K is the degree of $h(\lambda)$. For example, if $h(\lambda) = \lambda - \lambda^2$, we have $\mathbf{H}(\mathbf{S}'_n) = \mathbf{V}'(\mathbf{\Lambda}' - \mathbf{\Lambda}'^2)\mathbf{V}'^T = \mathbf{S}'_n - \mathbf{S}_n'^2$. Note that we can write the new graph shift operator as $\mathbf{S}'_n = \mathbf{S}_n + \mathbf{E}\mathbf{S}_n + \mathbf{S}_n\mathbf{E}$, where $\mathbf{E}$ is a diagonal matrix (i.e., if we remove the g_n-th node in $\mathcal{G}_n$, we have $\mathbf{E} = \mathrm{diag}[0, \ldots, 0, -1]$). In this case, $\mathbf{S}_n'^2$ can be found as

$$\mathbf{S}_n'^2 = \mathbf{S}_n^2 + 2\mathbf{S}_n\mathbf{E}\mathbf{S}_n + \mathbf{S}_n^2\mathbf{E} + \mathbf{E}\mathbf{S}_n^2 + \mathbf{E}\mathbf{S}_n\mathbf{E}\mathbf{S}_n + \mathbf{E}\mathbf{S}_n^2\mathbf{E} + \mathbf{S}_n\mathbf{E}^2\mathbf{S}_n + \mathbf{S}_n\mathbf{E}\mathbf{S}_n\mathbf{E}.$$

Thus, if we can store the values of all $\{\mathbf{S}_n^k\}_{k=1}^K$ in advance during initial training, we can reduce the complexity of computing $\{\mathbf{S}_n'^k\}_{k=1}^K$ to $O(g_n^2)$, due to the fact that whenever $\mathbf{E}$ is involved in a matrix multiplication (i.e., $\mathbf{S}_n\mathbf{E}\mathbf{S}_n$), the computation essentially reduces to matrix-vector multiplication which is of complexity $O(g_n^2)$. Therefore, the complexity of computing $\{\mathbf{S}_n'^k\}_{k=1}^K$ is $O(g_n^2)$ and the overall computational complexity of obtaining $\mathbf{z}'_n$ is $O(dg_n^2)$.

Lastly, if $h(\lambda)$ is an arbitrary function, and we need to recompute the eigenvalue decomposition of $\mathbf{S}'_n$, the problem is related to a classical problem termed "downdating of the singular value decomposition" of a perturbed matrix [27]. The overall complexity of obtaining $\mathbf{z}'_n$ then becomes $O(g_n^2(\log^2\xi + d))$, where ξ is a parameter related to machine precision.

It is worth pointing out that $O(g_n^2)$ is order-optimal with respect to the unlearning complexity of removing nodes from a graph $\mathcal{G}_n$, since the complexity of the basic operation, graph convolution (i.e., $\mathbf{S}\mathbf{x}$), is $O(g_n^2)$. As we will show in Section 6, the unlearning complexity of using nontrainable GSTs is significantly smaller than that of using GNNs when constructing graph embeddings in the worst case. This is due to the fact that we may need to retrain GNNs frequently to eliminate the effect of removed nodes on the embedding procedure; on the other hand, we only need to recompute the embeddings of affected training graphs when using GSTs. The GSTs for different training graphs are computed independently, which may be seen as a form of sharding with small components. However, unlike traditional sharding-based methods [3, 7], we do not need to carefully select the partition, and the sizes of the shards do not affect the performance of the final model.

Using differentially-private GNNs for graph embeddings. To ensure that the gradient residual norm does not grow excessively, we need to have control over the graph embedding procedure so that the embedding is stable with respect to small perturbations in the graph topology and features. The nontrainable GST is one choice, but DP-GNNs can also be used for generating the graph embeddings as they may improve the overall performance of the learner. Based on Theorem 5 from [28], the overall learning framework still satisfies the certified approximate removal criterion, and thus can be used as an approximate unlearning method as well. However, most DP-GNNs focus on node classification instead of graph classification tasks, and it remains an open problem to design efficient GNNs for graph classification problems while preserving node-level privacy. Moreover, it has been shown in [9] that DP-GNNs often require a high "privacy cost" ($\epsilon \geq 5$) (see Equation (3)) to unlearn one node without introducing significant negative effects on model performance. In contrast, we find that in practice, our proposed unlearning approach based on GSTs only requires $\epsilon = 1$. Therefore, using DP-GNNs for graph embedding in unlearning frameworks may not offer any advantages compared to alternatives.

6 EXPERIMENTAL RESULTS

Settings. We test our methods on five benchmarking datasets for graph classification, including two real social networks datasets IMDB, COLLAB [43], and three other standard graph classification benchmarking datasets MNIST, CIFAR10, PROTEINS [11, 12, 14, 34, 59]. As we focus on the limited training data regime, we use 10 random splits for all experiments with the training/validation/testing ratio 0.1/0.1/0.8. Following [28], we use LBFGS as the optimizer for all non-GNN methods due to its high efficiency on strongly convex problems. We adopt the Adam [33] optimizer for GNNs following the implementation of Pytorch Geometric library benchmarking examples [18]. We compare our unlearning approach (Figure 1 (a)) with a naive application of [28] (Figure 1 (b)) as well as complete retraining. The tested backbone graph learning models include GST, GFT, linear-GST (i.e., GST without nonlinear activations) and GIN [58]. For all approximate unlearning methods we use $(\epsilon, \delta) = (1, 10^{-4})$ and noise $\alpha = 0.1$ as the default parameters unless specified otherwise. The shaded area in all plots indicates one standard deviation. Additional details are in Appendix F.

Performance of the backbone models. We first test the performance of all backbone graph learning models on the standard graph classification problem. The results are presented in Tables 1 and 2. We observe that GST has consistently smaller running times compared to GIN while offering matching or better accuracy. This validates the findings of [22] and confirms that GST is indeed efficient and effective in the limited training data regime. Compared to linear-GSTs, we find that the nonlinearity of GST is important to achieve better accuracy, with an average increase of 3.5% in test accuracy over five datasets. In general, GST also significantly outperforms GFT with respect to accuracy with a significantly smaller running time. This is due to the fact that GST (with polynomial wavelet kernel functions as in [22]) does not require an eigenvalue decomposition as GFT does, which is computationally expensive to perform for large datasets.

Performance of different unlearning methods. Next, we test various unlearning schemes combined with GST. In this set of experiments, we sequentially unlearn one node from each of the selected 10% training graphs. We compare our Algorithm 2 with the unstructured unlearning method [28] and complete retraining (Retrain). Note that in order to apply unstructured unlearning to the graph classification problem, we have to remove the entire training graph whenever we want to unlearn even one single node from it. The results are depicted in Figure 3. We can see that our unlearning scheme with GSTs has accuracy comparable to that of complete retraining but much lower time complexity. Also, note that a naive application of [28] (indexed "UU" for unstructured unlearning) results in complete retraining in almost all the cases (see Table 3). In addition, the method requires removing the entire training graph instead of just one node as requested, thus the accuracy can drop significantly when unlearning many requests (Figure 4).

Table 1: Test accuracy (%) of the backbone graph learning methods for standard graph classification in the limited training data regime. The results report the mean accuracy and standard deviation. Bold numbers indicate the best results.

	IMDB	PROTEINS	COLLAB	MNIST	CIFAR10
GST	**68.56 ± 3.52**	**68.26 ± 2.28**	**74.42 ± 0.81**	47.59 ± 0.25	**33.12 ± 0.40**
linear-GST	**68.30 ± 3.67**	62.79 ± 4.67	73.84 ± 0.70	38.52 ± 0.26	31.07 ± 0.21
GFT	50.81 ± 1.32	49.67 ± 1.45	34.58 ± 0.79	10.13 ± 0.22	10.00 ± 0.15
GIN	66.63 ± 4.29	65.12 ± 1.55	73.11 ± 1.43	**48.17 ± 0.45**	30.05 ± 0.59

Table 2: Running time (s) of the backbone graph learning methods for standard graph classification in the limited training data regime. The results report the mean accuracy and standard deviation. Bold numbers indicate the best results.

	IMDB	PROTEINS	COLLAB	MNIST	CIFAR10
GST	6.47 ± 0.89	**7.57 ± 1.79**	**10.89 ± 1.08**	**82.94 ± 5.75**	**75.36 ± 1.17**
linear-GST	6.92 ± 1.50	**7.59 ± 1.25**	**10.94 ± 1.40**	**82.23 ± 0.83**	**74.98 ± 1.07**
GFT	**4.43 ± 1.04**	9.00 ± 0.96	137.69 ± 1.29	1307.43 ± 1.10	4240.62 ± 2.56
GIN	23.13 ± 1.32	21.94 ± 0.92	949.06 ± 63.68	1279.26 ± 30.92	1239.03 ± 33.06

Table 3: Number of retraining from scratch during sequential unlearning one node from 10% of training graphs.

	IMDB	PROTEINS	COLLAB	MNIST	CIFAR10
GST	3.3	7.2	7.7	65.9	113.0
GST UU	10.0	11.0	50.0	550.0	450.0
GST Retrain	10	11	50	550	450
linear-GST	3.0	6.8	6.3	54.1	91.6
linear-GST UU	10.0	11.0	49.6	532.5	450.0

We further examine the performance of these unlearning approaches in the "extreme" graph unlearning setting: Now we unlearn one node from each of the 90% training graphs sequentially. Due to the high time complexity of baseline methods, we conduct this experiment only on one small dataset, IMDB, and one medium dataset, COLLAB. We also compare completely retraining GIN on IMDB. The results are shown in Figure 4. We observe that retraining GIN is indeed prohibitively complex in practice. A naive application of [28] (indexed "UU") leads to a huge degradation in accuracy despite the high running complexity. This is due to the fact that one has to remove the entire training graph for each node unlearning request, which is obviously wasteful. Overall, our proposed strategy

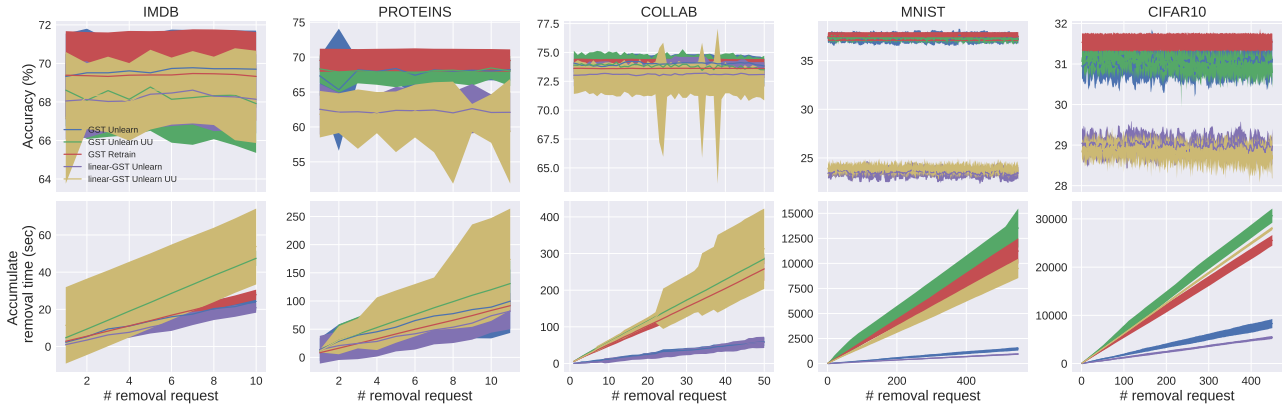

Figure 3: Sequential unlearning results. We unlearn one node in each of the 10% of the selected training graphs. The shaded area indicates one standard deviation. All approximate unlearning methods satisfy $(1, 10^{-4})$-certified approximate removal.

combined with GST gives the best results regarding time complexity and offers comparable test accuracy and affordable privacy costs.

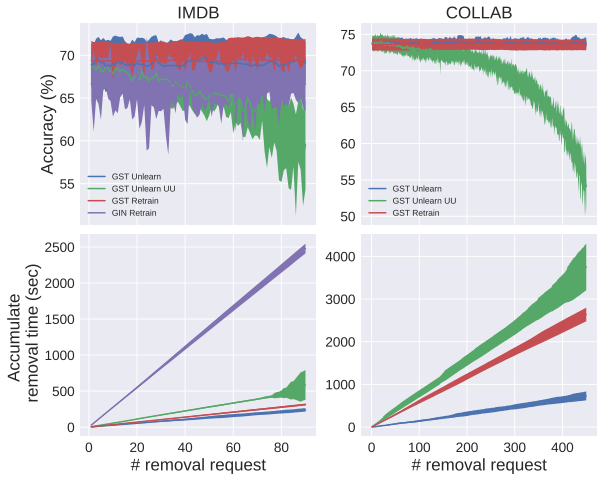

Figure 4: Unlearning results for the extreme setting. We unlearn one node in each of the 90% training graphs.

Performance of the proposed method with abundant training data. Next, we use the IMDB dataset as an example to demonstrate the performance of our unlearning method compared to complete retraining; the training/validation/testing ratio is set to 0.6/0.2/0.2, and out of 600 training graphs, we unlearn 500 samples in terms of each removing a single node. The results (see Figure 5) show that our approach still offers roughly a 10-fold decrease in unlearning time complexity compared to retraining GIN, with comparable test accuracy. Note that due to the high complexity of retraining GIN in this setting, we only performed complete retraining for the number of removal requests indicated using green marks in Figure 5. The green lines correspond to the interpolated results.

Bounds on the gradient residual norm. We also examine the *worst-case* bounds (Theorem 4.4) and the *data-dependent* bounds (Theorem 4.7) of Algorithm 2 computed during the unlearning process, along with the true value of the gradient residual norm (True norm) to validate our theoretical findings from Section 4. For

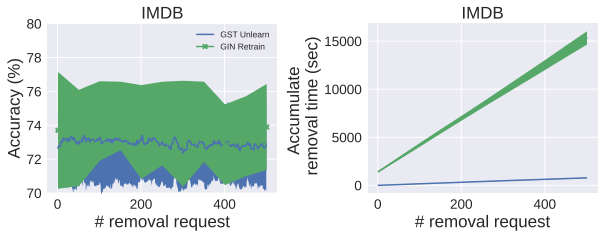

Figure 5: Unlearning results when training data is abundant. We unlearn one node in each of the 83% training graphs.

simplicity, we set $\alpha = 0$ during training. Figure 6 confirms that the worst-case bounds are looser than the data-dependent bounds.

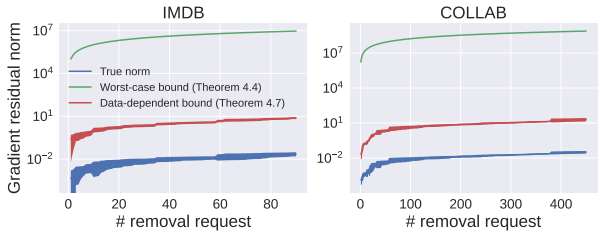

Figure 6: Comparison of the worst-case and data-dependent bounds on the gradient residual norm, and the true value of the gradient residual norm.

7 CONCLUSION

We studied the first nonlinear graph learning framework based on GSTs that accommodates an approximate unlearning mechanism with provable performance guarantees on computational complexity. With the adoption of the mathematically designed transform GSTs, we successfully extended the theoretical analysis on linear models to nonlinear graph learning models. Our experimental results validated the remarkable unlearning efficiency improvements compared to complete retraining.

ACKNOWLEDGMENTS

This work was funded by NSF grants 1816913 and 1956384.

REFERENCES

[1] Martin Anthony and Peter L Bartlett. 2009. *Neural network learning: Theoretical foundations*. cambridge university press.

[2] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2021. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[3] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 141–159.

[4] Joan Bruna and Stéphane Mallat. 2013. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1872–1886.

[5] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 463–480.

[6] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12, 3 (2011).

[7] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 499–513.

[8] Eli Chien, Chao Pan, and Olgica Milenkovic. 2022. Certified Graph Unlearning. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*.

[9] Eli Chien, Chao Pan, and Olgica Milenkovic. 2023. Efficient Model Updates for Approximate Unlearning of Graph-Structured Data. In *International Conference on Learning Representations*.

[10] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. 2021. Node-Level Differentially Private Graph Neural Networks. *arXiv preprint arXiv:2111.15521* (2021).

[11] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[12] Paul D Dobson and Andrew J Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology* 330, 4 (2003), 771–783.

[13] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems* 28 (2015).

[14] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2020. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982* (2020).

[15] Cynthia Dwork. 2011. Differential privacy. Encyclopedia of cryptography and security.

[16] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[17] Li Fei-Fei, Robert Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence* 28, 4 (2006), 594–611.

[18] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[19] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333.

[20] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. 2019. Diffusion Scattering Transforms on Graphs. In *International Conference on Learning Representations*.

[21] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. 2019. Stability of graph scattering transforms. *Advances in Neural Information Processing Systems* 32 (2019).

[22] Feng Gao, Guy Wolf, and Matthew Hirn. 2019. Geometric scattering for graph data analysis. In *International Conference on Machine Learning*. 2122–2131.

[23] Thomas Gaudelet, Ben Day, Arian R Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy BR Hayter, Richard Vickers, Charles Roberts, Jian Tang, et al. 2021. Utilizing graph machine learning within drug discovery and development. *Briefings in bioinformatics* 22, 6 (2021), bbab159.

[24] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. 2019. Making ai forget you: Data deletion in machine learning. *Advances in Neural Information Processing Systems* 32 (2019).

[25] David F Gleich. 2015. PageRank beyond the Web. *siam REVIEW* 57, 3 (2015), 321–363.

[26] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9304–9312.

[27] Ming Gu and Stanley C Eisenstat. 1995. Downdating the singular value decomposition. *SIAM J. Matrix Anal. Appl.* 16, 3 (1995), 793–810.

[28] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. 2020. Certified Data Removal from Machine Learning Models. In *International Conference on Machine Learning*. PMLR, 3832–3842.

[29] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[30] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.

[31] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xia, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware coupled graph neural network for social recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4115–4122.

[32] Vassilis N Ioannidis, Siheng Chen, and Georgios B Giannakis. 2020. Pruned graph scattering transforms. In *International Conference on Learning Representations*.

[33] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[34] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[35] Jia Li, Yu Rong, Hong Cheng, Helen Meng, Wenbing Huang, and Junzhou Huang. 2019. Semi-supervised graph classification: A hierarchical graph perspective. In *The World Wide Web Conference*. 972–982.

[36] Maosen Li, Siheng Chen, Zihui Liu, Zijing Zhang, Lingxi Xie, Qi Tian, and Ya Zhang. 2021. Skeleton graph scattering networks for 3d skeleton-based human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 854–864.

[37] Ruirui Li, Xian Wu, Xian Wu, and Wei Wang. 2020. Few-shot learning for new user recommendation in location-based social networks. In *Proceedings of The Web Conference 2020*. 2472–2478.

[38] Xiaoxiao Li, Nicha C Dvornek, Yuan Zhou, Juntang Zhuang, Pamela Ventola, and James S Duncan. 2019. Graph neural network for interpreting task-fmri biomarkers. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 485–493.

[39] Stéphane Mallat. 2012. Group invariant scattering. *Communications on Pure and Applied Mathematics* 65, 10 (2012), 1331–1398.

[40] Chengsheng Mao, Liang Yao, and Yuan Luo. 2019. Medgcn: Graph convolutional networks for multiple medical tasks. *arXiv preprint arXiv:1904.00326* (2019).

[41] Yimeng Min, Frederik Wenkel, and Guy Wolf. 2020. Scattering gcn: Overcoming oversmoothness in graph convolutional networks. *Advances in Neural Information Processing Systems* 33 (2020), 14498–14508.

[42] Mohammadreza Mohammadrezaei, Mohammad Ebrahim Shiri, and Amir Masoud Rahmani. 2018. Identifying fake accounts on social networks based on graph analysis and classification algorithms. *Security and Communication Networks* 2018 (2018).

[43] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663* (2020).

[44] Tamara T Mueller, Johannes C Paetzold, Chinmay Prabhakar, Dmitrii Usynin, Daniel Rueckert, and Georgios Kaissis. 2022. Differentially Private Graph Classification with GNNs. *arXiv preprint arXiv:2202.02575* (2022).

[45] Chao Pan, Siheng Chen, and Antonio Ortega. 2021. Spatio-Temporal Graph Scattering Transform. In *International Conference on Learning Representations*.

[46] Chao Pan, Jin Sima, Saurav Prakash, Vishal Rana, and Olgica Milenkovic. 2023. Machine Unlearning of Federated Clusters. In *International Conference on Learning Representations*.

[47] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. 2022. GAP: Differentially Private Graph Neural Networks with Aggregation Perturbation. *arXiv preprint arXiv:2203.00949* (2022).

[48] Aliaksei Sandryhaila and José MF Moura. 2013. Discrete signal processing on graphs: Graph Fourier transform. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 6167–6170.

[49] Victor Garcia Satorras and Joan Bruna Estrach. 2018. Few-shot learning with graph neural networks. In *International conference on learning representations*.

[50] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. 2021. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems* 34 (2021).

[51] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine* 30, 3 (2013), 83–98.

[52] David I Shuman, Christoph Wiesmeyr, Nicki Holighaus, and Pierre Vandergheynst. 2015. Spectrum-adapted tight graph wavelet and vertex-frequency frames. *IEEE Transactions on Signal Processing* 63, 16 (2015), 4223–4235.

[53] Michael Veale, Reuben Binns, and Lilian Edwards. 2018. Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376, 2133 (2018), 20180083.

[54] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)* 53, 3 (2020), 1–34.
[55] Max Welling and Thomas N Kipf. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.
[56] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.
[57] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)* (2020).
[58] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.
[59] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1365–1374.
[60] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.
[61] Dongmian Zou and Gilad Lerman. 2020. Graph convolutional neural networks via scattering. *Applied and Computational Harmonic Analysis* 49, 3 (2020), 1046–1074.

A PROOF OF THEOREM 4.3

Our proof is a generalization of the proof in [28]. Due to space limitations, we show proof sketches here and delegate the details to a full version of this paper. Let $G(\mathbf{w}) = \nabla L(\mathbf{w}, \mathcal{D}')$ denote the gradient of the empirical risk on the updated dataset $\mathcal{D}'$ at $\mathbf{w}$. By Taylor's expansion theorem, there exists some $\eta \in [0, 1]$ such that

$$\begin{aligned} G(\mathbf{w}') &= G(\mathbf{w}^\star + \mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta) = G(\mathbf{w}^\star) + \nabla G(\mathbf{w}^\star + \eta\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta)\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta \\ &= (\mathbf{H}_{\mathbf{w}_\eta} - \mathbf{H}_{\mathbf{w}^\star})\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta, \end{aligned} \tag{10}$$

where $\mathbf{H}_{\mathbf{w}_\eta} \triangleq \nabla G(\mathbf{w}^\star + \eta\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta)$, which is the Hessian at $\mathbf{w}_\eta \triangleq \mathbf{w}^\star + \eta\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta$. By the Cauchy-Schwartz inequality, we have

$$\|G(\mathbf{w}')\| \le \|\mathbf{H}_{\mathbf{w}_\eta} - \mathbf{H}_{\mathbf{w}^\star}\|\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\|. \tag{11}$$

Note that since ℓ'' is γ_2-Lipschitz, for $\forall i \in [n]$ we have

$$\left\|\nabla^2\ell\left(\mathbf{w}_\eta^T\mathbf{z}_i, y_i\right) - \nabla^2\ell\left(\left(\mathbf{w}^\star\right)^T\mathbf{z}_i, y_i\right)\right\| \le \gamma_2 F^3\left\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\right\|, \tag{12}$$

where $F = \sqrt{\sum_{l=0}^{L-1} B^{2l}}$. The upper bound comes from the frame property of graph wavelets. More specifically, the sum of energy for the l-th layer in the scattering tree is upper bounded by $B^{2l}\|\mathbf{x}_i\|^2$, where $\mathbf{x}_i$ is the input graph signal at the 0-th layer and $\|\mathbf{x}_i\|^2$ is the corresponding energy. Suppose that we are using the low-pass averaging operator for U_i in GST (i.e., $U_i = \frac{1}{g_i}\mathbf{1}^T$, $\|U_i\| = \frac{1}{\sqrt{g_i}}$). Then

$$\|\mathbf{z}_i\|^2 = \sum_{p_j(l)} \phi_{p_j(l)}^2(\mathbf{S}_i, \mathbf{x}_i) \le \frac{\sum_{l=0}^{L-1} B^{2l}\|\mathbf{x}_i\|^2}{g_i} \le \sum_{l=0}^{L-1} B^{2l}, \tag{13}$$

since $\|\mathbf{x}_i\|^2 \le g_i$ based on Assumption 4.2. Therefore, we have $\|\mathbf{z}_i\|^3 \le \left(\sum_{l=0}^{L-1} B^{2l}\right)^{3/2} = F^3$. Summing the expression in Equation (12) over the updated dataset $\mathcal{D}'$, we conclude that

$$\|\mathbf{H}_{\mathbf{w}_\eta} - \mathbf{H}_{\mathbf{w}^\star}\| \le \gamma_2 n F^3\left\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\right\|.$$

Replacing the above equality into Equation (11), we arrive at

$$\|G(\mathbf{w}')\| \le \gamma_2 n F^3\left\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\right\|^2.$$

Next, we bound $\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\|$. Since $L(\cdot, \mathcal{D}')$ is (λn)-strongly convex, we have $\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\| \le \frac{1}{\lambda n}$. For Δ, by definition,

$$\Delta = \nabla\ell\left(\left(\mathbf{w}^\star\right)^T\mathbf{z}_n, y_n\right) - \nabla\ell\left(\left(\mathbf{w}^\star\right)^T\mathbf{z}_n', y_n\right).$$

There are two ways to bound $\|\Delta\|$. First, based on Assumption 4.2,

$$\|\Delta\| \le \left\|\nabla\ell\left(\left(\mathbf{w}^\star\right)^T\mathbf{z}_n, y_n\right)\right\| + \left\|\nabla\ell\left(\left(\mathbf{w}^\star\right)^T\mathbf{z}_n', y_n\right)\right\| \le 2C_1.$$

Second, we can also bound $\|\Delta\|$ by

$$\begin{aligned} \|\Delta\| \le& \left|\ell'\left(\left(\mathbf{w}^\star\right)^T\mathbf{z}_n, y_n\right) - \ell'\left(\left(\mathbf{w}^\star\right)^T\mathbf{z}_n', y_n\right)\right|\|\mathbf{z}_n\| \\ &+ \left|\ell'\left(\left(\mathbf{w}^\star\right)^T\mathbf{z}_n', y_n\right)\right|\|\mathbf{z}_n - \mathbf{z}_n'\| \\ \le& \gamma_1\|\mathbf{z}_n - \mathbf{z}_n'\|\|\mathbf{w}^\star\|\|\mathbf{z}_n\| + C_2\|\mathbf{z}_n - \mathbf{z}_n'\|. \end{aligned}$$

For $\|\mathbf{w}^\star\|$, we have

$$0 = \nabla L(\mathbf{w}^\star, \mathcal{D}) = \sum_{i=1}^{n}\nabla\ell\left(\left(\mathbf{w}^\star\right)^T\mathbf{z}_i, y_i\right) + \lambda n\mathbf{w}^\star,$$

which leads to

$$\|\mathbf{w}^\star\| \le \frac{\sum_{i=1}^{n}\|\nabla\ell\left(\left(\mathbf{w}^\star\right)^T\mathbf{z}_i, y_i\right)\|}{\lambda n} \le \frac{C_1}{\lambda}.$$

From Theorem 1 of [45], by setting $T = 1$, we have

$$\|\mathbf{z}_n - \mathbf{z}_n'\| \le \sqrt{\frac{\sum_{l=0}^{L-1} B^{2l}}{g_n}}\|\mathbf{x}_n - \mathbf{x}_n'\| \le \frac{F}{\sqrt{g_n}}.$$

Combining the above results we obtain

$$\|\Delta\| \le \frac{\gamma_1 C_1 F^2 + \lambda C_2 F}{\lambda\sqrt{g_n}}.$$

Therefore,

$$\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\| \le \frac{1}{\lambda n}\cdot\min\left\{2C_1, \frac{\gamma_1 C_1 F^2 + \lambda C_2 F}{\lambda\sqrt{g_n}}\right\}. \tag{14}$$

Replacing Equation (14) into the bound on $\|G(\mathbf{w}')\|$, we arrive at

$$\|G(\mathbf{w}')\| \le \frac{\gamma_2 F^3}{\lambda^2 n}\min\left\{4C_1^2, \frac{(\gamma_1 C_1 F^2 + \lambda C_2 F)^2}{\lambda^2 g_n}\right\},$$

which completes the proof.

B PROOF OF THEOREM 4.4

Following the same proof approach as described in Appendix A, for the case of single node removal we have

$$\|G(\mathbf{w}')\| \le \gamma_2 n F^3\left\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\right\|^2.$$

The main difference between feature removal and node removal proof is that the norm of the change in the graph embeddings $\|\mathbf{z}_n' - \mathbf{z}_n\|$ obtained by GST with respect to structural perturbation is proportional to the norm of the entire graph signal $\|\mathbf{x}_n\|$. More specifically, when the magnitude of the structural perturbation is controlled and the wavelet kernel functions satisfy certain mild conditions, from Theorem 2 of [45] we have that (for $T = 1$)

$$\begin{aligned} \|\mathbf{z}_n' - \mathbf{z}_n\| &= \left\|\Phi(\mathbf{S}_n', \mathbf{x}_n') - \Phi(\mathbf{S}_n, \mathbf{x}_n)\right\| \\ &= \left\|\Phi(\mathbf{S}_n', \mathbf{x}_n') - \Phi(\mathbf{S}_n', \mathbf{x}_n) + \Phi(\mathbf{S}_n', \mathbf{x}_n) - \Phi(\mathbf{S}_n, \mathbf{x}_n)\right\| \\ &\le \frac{F}{\sqrt{g_n}} + O\left(\sqrt{\frac{\sum_{l=0}^{L-1} l^2(B^2J)^l}{B^2 g_n}}\right)\|\mathbf{x}_n\| \\ &\le \frac{F}{\sqrt{g_n}} + O\left(\sqrt{\frac{\sum_{l=0}^{L-1} l^2(B^2J)^l}{B^2}}\right), \end{aligned}$$

where $\|\mathbf{x}_n\| \le \sqrt{g_n}$ and $F = \sqrt{\sum_{l=0}^{L-1} B^{2l}}$. In this case, the second term in the upper bound of $\|\mathbf{z}_n' - \mathbf{z}_n\|$ does not decrease when g_n increases, and $\|\Delta\| \le 2C_1$ is very likely a tighter upper bound than the other option. Therefore, we have $\|G(\mathbf{w}')\| \le \frac{4\gamma_2 C_1^2 F^3}{\lambda^2 n}$, which completes the proof.

C PROOF OF COROLLARIES 4.5 AND 4.6

The proof of Corollaries 4.5 and 4.6 follows along similar lines as that in Appendix A and B. With the same argument, we can show that $\|G(\mathbf{w}')\| \le \gamma_2 n F^3\left\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\right\|^2$. The only difference arises from the fact that there could be now at most $2m$ terms in Δ instead of

just 2 terms, and the worst case arises when each of these m nodes that requested removal comes from a different graph. In this case, $\|\Delta\| \leq 2mC_1$. Therefore, for batch feature removal we have

$$\|\nabla L(\mathbf{w}', \mathcal{D}')\| \leq \frac{\gamma_2 m^2 F^3}{\lambda^2 n} \min\left\{4C_1^2, \frac{(\gamma_1 C_1 F^2 + \lambda C_2 F)^2}{\lambda^2 g_n}\right\},$$

while for batch node removal, we have

$$\|\nabla L(\mathbf{w}', \mathcal{D}')\| \leq \frac{4\gamma_2 m^2 C_1^2 F^3}{\lambda^2 n}.$$

Note that we require $m < \min_i g_i$ not to unlearn the entire graph, otherwise the number of training samples in Equation (1) would be less than n and tighter bounds on gradient residual norm can be derived. Nevertheless, if we indeed need to unlearn multiple graphs completely, we can always unlearn them first based on the batch removal procedure in [28], and then perform our unlearning procedure based on Equation (4) for the remaining graphs.

D PROOF OF THEOREM 4.7

Based on the loss function defined in Equation (1), the Hessian of $L(\cdot, \mathcal{D}')$ at $\mathbf{w}$ takes the form $\mathbf{H}_\mathbf{w} = (\mathbf{Z}')^T \mathbf{D}_\mathbf{w} \mathbf{Z}' + \lambda n \mathbf{I}_d$, where $\mathbf{Z}' \in \mathbb{R}^{n \times d}$ is the data matrix corresponding to $\mathcal{D}'$ and $\mathbf{D}_\mathbf{w}$ denotes the diagonal matrix with diagonal values $(\mathbf{D}_\mathbf{w})_{ii} = \ell''(\mathbf{w}^T \mathbf{z}_i, y_i)$.

From the proof of Theorem 4.4 we know that

$$\begin{aligned}\left\|\nabla L\left(\mathbf{w}', \mathcal{D}'\right)\right\| &\leq \left\|\left(\mathbf{H}_{\mathbf{w}_\eta} - \mathbf{H}_{\mathbf{w}^\star}\right)\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\right\| \\ &= \left\|\left(\mathbf{Z}'\right)^T\left(\mathbf{D}_{\mathbf{w}_\eta} - \mathbf{D}_{\mathbf{w}^\star}\right)\mathbf{Z}'\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\right\| \\ &\leq \|\mathbf{Z}'\|\|\mathbf{D}_{\mathbf{w}_\eta} - \mathbf{D}_{\mathbf{w}^\star}\|\|\mathbf{Z}'\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\|, \qquad (15)\end{aligned}$$

where $\mathbf{w}_\eta \triangleq \mathbf{w}^\star + \eta \mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta$ for some $\eta \in [0, 1]$. Since $\mathbf{D}_{\mathbf{w}_\eta} - \mathbf{D}_{\mathbf{w}^\star}$ is a diagonal matrix, its operator norm corresponds to the maximum absolute value of the diagonal elements. In the proof of Theorem 4.4 we showed that for $\forall i \in [n]$,

$$\begin{aligned}\left|\ell''\left(\mathbf{w}_\eta^T \mathbf{z}_i, y_i\right) - \ell''\left(\left(\mathbf{w}^\star\right)^T \mathbf{z}_i, y_i\right)\right| &\leq \gamma_2 \|\mathbf{w}_\eta - \mathbf{w}^\star\|\|\mathbf{z}_i\| \\ &\leq \gamma_2 F \left\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\right\|.\end{aligned}$$

Thus we have that $\|\mathbf{D}_{\mathbf{w}_\eta} - \mathbf{D}_{\mathbf{w}^\star}\| \leq \gamma_2 F \left\|\mathbf{H}_{\mathbf{w}^\star}^{-1}\Delta\right\|$. Combining this result with Equation (15) completes the proof. Note that this analysis holds for both single-removal and batch-removal, as well as both feature and node removal requests, since it does not require an explicit upper bound on the norm of gradient change $\|\Delta\|$.

E CHOICES OF GRAPH WAVELETS

There are many off-the-shelf graph wavelets we can choose from. They are mainly used for extracting features from multiple frequency bands of the input signal spectrum. Some are listed below.

Monic Cubic wavelets. Monic Cubic wavelets [30] use a kernel function $h(\lambda)$ of the form

$$h(\lambda) = \begin{cases}\lambda & \text{for} \quad \lambda < 1; \\ -5 + 11\lambda - 6\lambda^2 + \lambda^3 & \text{for} \quad 1 \leq \lambda \leq 2; \\ 2/\lambda & \text{for} \quad \lambda > 2.\end{cases}$$

Different scalings of the filters are implemented by scaling and translating the above kernel function.

Itersine wavelets. Itersine wavelets define the kernel function at scale j as

$$h_j(\lambda) = \sin\left(\frac{\pi}{2}\cos^2\left(\pi\left(\lambda - \frac{j-1}{2}\right)\right)\right) \cdot \mathbb{I}\left[\frac{j}{2} - 1 \leq \lambda \leq \frac{j}{2}\right].$$

Itersine wavelets form tight and energy-preserving frames.

Diffusion scattering wavelets. A diffusion scattering wavelet filter bank [20] contains a set of filters based on a lazy diffusion matrix $\mathbf{S} = \frac{1}{2}(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})$, where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}$ is the corresponding degree matrix. The filters are defined as

$$\mathbf{H}_j(\mathbf{S}) = \begin{cases}\mathbf{I} - \mathbf{S}, & j = 0, \\ \mathbf{S}^{2^{j-1}} - \mathbf{S}^{2^j}, & j > 0.\end{cases}$$

Note that for diffusion scattering the low-pass operator U is defined as $\mathbf{d}/\|\mathbf{d}\|_1$, where $\mathbf{d}$ is the diagonal of $\mathbf{D}$.

Geometric scattering wavelets. The definition of geometric scattering [22] is similar as diffusion scattering, except that the lazy random walk matrix used in geometric scattering is defined as $\mathbf{S} = \frac{1}{2}(\mathbf{I} + \mathbf{A}\mathbf{D}^{-1})$. And geometric scattering will also record different moments of filtered signals $\mathbf{H}_j^q(\mathbf{x}), \forall q \in [Q]$ as features.

Note that one is also allowed to customize the graph wavelets, as long as they satisfy the constraint

$$A^2\|\mathbf{x}\|^2 \leq \sum_{j=1}^{J}\left\|\mathbf{H}_j\mathbf{x}\right\|^2 \leq B^2\|\mathbf{x}\|^2, \quad |\lambda h'(\lambda)| \leq \text{const } \forall\lambda,$$

where A, B are scalar constants and $h(\cdot)$ is the kernel function.

F ADDITIONAL EXPERIMENTAL DETAILS

Hyperparameters. We follow the PyG benchmarking code to preprocess the datasets. For datasets without node features, we generate synthetic node features based on node degrees. For all methods, we perform training with 500 epochs. For GIN, we tune the hyperparameters on the small dataset IMDB and subsequently use them on all other datasets. We use 2 layers, 64 hidden dimensions and a learning rate 10^{-4} in the Adam optimizer. We find that this setting works well in general. For GST, we use geometric scattering wavelets in the graph embedding procedure and fix the learning rate of the LBFGS optimizer to 0.5 for training the classifier. Other hyperparameters used for GST are described in Table 4. Here J represents the number of scales for graph wavelets $\{h_j\}_{j=1}^J$, L represents the number of layers in the scattering tree (with the root node at layer 0), and Q represents the number of moments computed for geometric scattering wavelets (see Appendix E).

Table 4: The hyperparameters of GST for all experiments.

	Standard graph classification				Unlearning graph classification				
	J	Q	L	λ	J	Q	L	λ	α
IMDB	5	4	3	10^{-4}	4	3	3	10^{-3}	10^{-1}
PROTEINS	5	4	3	10^{-4}	5	4	3	10^{-4}	10^{-1}
COLLAB	3	3	2	10^{-4}	3	3	2	10^{-4}	10^{-1}
MNIST	5	4	3	0	5	4	3	10^{-6}	10^{-1}
CIFAR10	5	4	3	0	5	4	3	10^{-6}	10^{-1}