Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 338–350
Marseille, 20-25 June 2022

Privacy-Preserving Graph Convolutional Networks for Text Classification

Timour Igamberdiev, Ivan Habernal
Trustworthy Human Language Technologies
Department of Computer Science
Technical University of Darmstadt
{timour.igamberdiev,ivan.habernal}@tu-darmstadt.de
www.trusthlt.org

Abstract

Graph convolutional networks (GCNs) are a powerful architecture for representation learning on documents that naturally occur as graphs, e.g., citation or social networks. However, sensitive personal information, such as documents with people's profiles or relationships as edges, are prone to privacy leaks, as the trained model might reveal the original input. Although differential privacy (DP) offers a well-founded privacy-preserving framework, GCNs pose theoretical and practical challenges due to their training specifics. We address these challenges by adapting differentially-private gradient-based training to GCNs and conduct experiments using two optimizers on five NLP datasets in two languages. We propose a simple yet efficient method based on random graph splits that not only improves the baseline privacy bounds by a factor of 2.7 while retaining competitive F_1 scores, but also provides strong privacy guarantees of $\varepsilon = 1.0$. We show that, under certain modeling choices, privacy-preserving GCNs perform up to 90% of their non-private variants, while formally guaranteeing strong privacy measures.

Keywords: graph convolutional networks, text node classification, differential privacy

1. Introduction

Many text classification tasks naturally occur in the form of graphs where nodes represent text documents and edges are task specific, such as articles citing each other or health records belonging to the same patient. When learning node representations and predicting their categories, models benefit from exploiting information from the neighborhood of each node, as shown in graph neural networks, and graph convolutional networks (GCNs) in particular (Kipf and Welling, 2017), making them superior to other models (Xu et al., 2019; De Cao et al., 2019).

While GCNs are powerful for a variety of NLP problems, like other neural models they are prone to privacy attacks. Adversaries with extensive background knowledge and computational power might reveal sensitive information about the training data from the model, such as reconstructing information about the original classes of a model (Hitaj et al., 2017) or even auditing membership of an individual's data in a model (Song and Shmatikov, 2019). In order to preserve privacy for graph NLP data, models have to protect both the textual nodes and the graph structure, as both sources carry potentially sensitive information.

Privacy-preserving techniques, such as differential privacy (DP) (Dwork and Roth, 2013), prevent information leaks by adding 'just enough' noise during model training while attaining acceptable performance. Recent approaches to DP in neural models attempt to trade off between noise and utility, with differentially private stochastic gradient descent (SGD-DP) (Abadi et al., 2016) being a prominent example. However, SGD-DP's design expects i.i.d. data examples to form batches and 'lots', therefore its suitability for graph neural networks remains an open question.

In this work, we propose a methodology for applying differentially private stochastic gradient descent and its variants to GCNs, allowing to maintain strict privacy guarantees and performance. Our approach consists of applying an easy-to-implement graph splitting algorithm to GCNs in the DP setting, partitioning a graph into subgraphs while avoiding additional queries on the original data. We adapt SGD-DP (Abadi et al., 2016) to GCNs as well as propose a differentially-private version of Adam (Kingma and Ba, 2015), Adam-DP. We hypothesize that Adam's advantages, i.e. fewer training epochs, would lead to a better privacy/utility trade-off as opposed to SGD-DP.

We conduct experiments on five datasets in two languages (English and Slovak) covering a variety of NLP tasks, including research article classification in citation networks, Reddit post classification, and user interest classification in social networks, where the latter ones inherently carry potentially sensitive information calling for privacy-preserving models. Our main contributions are twofold. First, we show that DP training can be applied to the case of GCNs, with graph splitting and proper optimization recovering a lot of the dropped performance due to DP noise. Second, we show that more sophisticated text representations further mitigate the performance drop due to DP noise, resulting in a relative performance of 90% of the non-private variant, while keeping strict privacy ($\varepsilon = 1.0$ when using graph splits). To the best of our knowledge, this is the first study that brings differentially private gradient-based training to graph neural networks.[1]

[1]Code available at https://github.com/trusthlt/privacy-preserving-gcn

2. Theoretical background in DP

As DP does not belong to the mainstream methods in NLP, here we shortly outline the principles and present the basic terminology from the NLP perspective. Foundations can be found in (Dwork and Roth, 2013; Desfontaines and Pejó, 2020).

The main idea of DP is that if we query a database of N individuals, the result of the query will be almost indistinguishable from the result of querying a database of $N-1$ individuals, thus preserving each single individual's privacy to a certain degree. The difference of results obtained from querying any two databases that differ in one individual has a probabilistic interpretation.

Dataset D consists of $|D|$ documents where each document is associated with an individual whose privacy we want to preserve. A document can be any arbitrary natural language text, such as a letter, medical record, tweet, personal plain text passwords, or a paper review. Let D' differ from D by one document, so either $|D'| = |D| \pm 1$, or $|D'| = |D|$ with i-th document replaced. D and D' are called *neighboring* datasets.

Let $A : D \mapsto y \in \mathbb{R}$ be a randomized function applied to a dataset D; for example a function returning the average document length or the number of documents in the dataset. This function is also called a *query* which is not to be confused with queries in NLP, such as search queries.[2] In DP, this query function is a continuous random variable associated with a probability density $p_t(A(D) = y)$. Once the function $A(D)$ is applied on the dataset D, the result is a single draw from this probability distribution. This process is also known as a *randomized algorithm*. For example, a randomized algorithm for the average document length can be a Laplace density such that $p_t(A(D) = y) = \frac{1}{2b} \exp\left(-\frac{|\mu - y|}{b}\right)$, where μ is the true average document length and b is the scale (the 'noisiness' parameter). By applying this query to D, we obtain $y \in \mathbb{R}$, a single draw from this distribution.

Now we can formalize the backbone idea of DP. Having two neighboring datasets D, D', *privacy loss* is defined as

$$\ln \frac{p(A(D) = y)}{p(A(D') = y)}. \tag{1}$$

DP bounds this privacy loss by design. Given $\varepsilon \in \mathbb{R} : \varepsilon \geq 0$ (the *privacy budget* hyper-parameter), all values of y, and all neighboring datasets D and D', we must ensure that

$$\max_{\forall y} \left| \ln \frac{p(A(D) = y)}{p(A(D') = y)} \right| \leq \varepsilon \, . \tag{2}$$

In other words, the allowed privacy loss of any two neighboring datasets is upper-bounded by ε, also denoted as $(\varepsilon, 0)$-DP.[3] The privacy budget ε controls the amount of preserved privacy. If $\varepsilon \to 0$, the query outputs of any two datasets become indistinguishable, which guarantees almost perfect privacy but provides very little utility. Similarly, higher ε values provide less privacy but better utility. Finding the sweet spot is thus the main challenge in determining the privacy budget for a particular application (Lee and Clifton, 2011; Hsu et al., 2014). An important feature of (ε, δ)-DP is that once we obtain the result y of the query $A(D) = y$, any further computations with y cannot weaken the privacy guaranteed by ε and δ.

The desired behavior of the randomized algorithm is therefore adding as little noise as possible to maximize utility while keeping the privacy guarantees given by Eq. 2. The amount of noise is determined for each particular setup by the *sensitivity* of the query ΔA, such that for any neighboring datasets D, D' we have

$$\Delta A = \max_{\forall D, D'} \left(|A(D) - A(D')| \right) \, . \tag{3}$$

The sensitivity corresponds to the 'worst case' range of a particular query A, i.e., what is the maximum impact of changing one individual. The larger the sensitivity, the more noise must be added to fulfill the privacy requirements of ε (Eq. 2). For example, in order to be $(\varepsilon, 0)$-DP, the Laplace mechanism must add noise $b = (\Delta A)^{-1}$ (Dwork and Roth, 2013, p. 32). As the query sensitivity directly influences the required amount of noise, it is desirable to design queries with low sensitivity.

The so far described mechanisms consider a scenario when we apply the query only once. To ensure (ε, δ)-DP with multiple queries[4] on the same datasets, proportionally more noise has to be added.

3. Related work

A wide range of NLP tasks have been utilizing **graph neural networks** (GNNs), specifically graph convolutional networks (GCNs), including text summarization (Xu et al., 2020), machine translation (Marcheggiani et al., 2018) and semantic role labeling (Zheng and Kordjamshidi, 2020). Recent end-to-end approaches combine pre-trained transformer models with GNNs to learn graph representations for syntactic trees (Sachan et al., 2020). Rahimi et al. (2018) demonstrated the strength of GCNs on predicting geo-location of Twitter users where nodes are represented by users' tweets and edges by social connections, i.e. mentions of other Twitter users. However, for protecting user-level privacy, the overall social graph has to be taken into account.

Several recent works in the NLP area deal with **privacy** using arbitrary definitions. Li et al. (2018) propose

[2] In general, the query output is multidimensional $\mathbb{R}^k$; here we keep it scalar for the sake of simplicity.

[3] $(\varepsilon, 0)$-DP is a simplification of more general (ε, δ)-DP where δ is a negligible constant allowing relaxation of the privacy bounds (Dwork and Roth, 2013, p. 18).

[4] Queries might be different, for example querying the average document length first and then querying the number of documents in the dataset.

an adversarial-based approach to learning latent text representation for sentiment analysis and POS tagging. Although their privacy-preserving model performs on par with non-private models, they admit the lack of formal privacy guarantees. Similarly, Coavoux et al. (2018) train an adversarial model to predict private information on sentiment analysis and topic classification. The adversary's model performance served as a proxy for privacy strength but, despite its strengths, comes with no formal privacy guarantees. Similar potential privacy weaknesses can be found in a recent work by Abdalla et al. (2020) who replaced personal health information by semantically similar words while keeping acceptable accuracy of downstream classification tasks.

Abadi et al. (2016) pioneered the connection of DP and deep learning by proposing SGD-DP that bounds the query sensitivity using gradient clipping and formally guarantees the overall privacy budget. While originally tested on image recognition, they inspired subsequent work in language modeling using LSTMs (McMahan et al., 2018). However, to the best of our knowledge, training graph-based architectures with SGD-DP has not yet been explored. Two recent approaches utilize *local DP*, that is adding noise to each node before passing it to graph model training (Sajadmanesh and Gatica-Perez, 2020; Lyu et al., 2020), yet it is unclear whether it prevents leaking knowledge about edges. Our setup is different as we have access to the full dataset and preserve privacy of the entire graph.

As GCNs typically treat the entire graph as a single training example, Chiang et al. (2019) proposed a more efficient training using mini-batching methods. Despite the NP-hardness of the general graph splitting problem (Bui and Jones, 1992), they experimented with random partitioning and other clustering methods that take advantage of the graph structure (Karypis and Kumar, 1998). It remains an open question whether splitting the graph into disjoint i.i.d. examples would positively affect our DP-approach where mini-batches parametrize the required amount of noise.

4. Models

4.1. GCN as the underlying architecture

We employ the GCN architecture (Kipf and Welling, 2017) for enabling DP in the domain of graph-based NLP. GCN is a common and simpler variant to more complex types of GNNs which allows us to focus primarily on a comparison of the DP and non-DP models. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ model our graph data where each node $v_i \in \mathcal{V}$ contains a feature vector of dimensionality d. GCN aims to learn a node representation by integrating information from each node's neighborhood. The features of each neighboring node of v_i pass through a 'message passing function' (usually a transformation by a weight matrix Φ) and are then aggregated and combined with the current state of the node h_i^l to form the next state h_i^{l+1}. Edges are represented using an adjacency matrix $A \in \mathbb{R}^{n \times n}$, where n is the number of nodes in the graph. A is then multiplied by the matrix $H \in \mathbb{R}^{n \times f}$, f being the hidden dimension, as well as the weight matrix Φ responsible for message passing, learned during training. Additional tweaks by Kipf and Welling (2017) include adding the identity matrix to A to include self-loops in the computation $\hat{A} = A + \mathbf{I}$, as well as normalizing matrix A by the degree matrix D, specifically using a symmetric normalization $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. This results in the following equation for calculating the next state of the GCN for a given layer l, passing through a non-linearity function σ:

$$H^{l+1} = \sigma\left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} \Phi^{(l)}\right) \tag{4}$$

The final layer states for each node are then used for node-level classification, given output labels.

4.2. Baseline model: SGD-DP

SGD-DP (Abadi et al., 2016) modifies the standard stochastic gradient descent algorithm to be differentially private. The DP 'query' is the gradient computation at time step t: $g_t(x_i) \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, x_i)$, for each i in the training set. To ensure DP, the output of this query is distorted by random noise proportional to the sensitivity of the query, which is the range of values that the gradient can take. As gradient range is unconstrained, possibly leading to extremely large noise, Abadi et al. (2016) clip the gradient vector by its ℓ_2 norm, replacing each vector g with $\bar{g} = g / \max(1, \frac{||g||_2}{C})$, C being the clipping threshold. This clipped gradient is altered by a draw from a Gaussian: $\bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 I)$.

Instead of running this process on individual examples, Abadi et al. (2016) actually break up the training set into 'lots' of size L, being a slightly separate concept from that of 'batches'. Whereas the gradient computation is performed in batches, SGD-DP groups several batches together into lots for the DP calculation itself, which consists of adding noise, taking the average over a lot and performing the descent $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$.

Incorporating this concept, we obtain the overall core mechanism of SGD-DP:

$$\tilde{\mathbf{g}}_t = \frac{1}{L}\left(\sum_{i \in L} \frac{\mathbf{g}_t(x_i)}{\max\left(1, \frac{||\mathbf{g}_t(x_i)||_2}{C}\right)} + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right) \tag{5}$$

4.3. Our DP extension: Differentially-private Adam

In this paper, we also propose a DP version of Adam (Kingma and Ba, 2015), a widely-used default optimizer in NLP (Ruder, 2016). As Adam shares the core principle of gradient computing within SGD, to make it differentialy private we add noise to the gradient following Eq. 5, prior to Adam's moment estimates and parameter update. Adam-DP thus guarantees privacy like SGD-DP does, namely (1) by DP privatizing the query, that is the gradient, and (2) by a composition theorem, that is a sequence of DP mechanisms remains DP.

Despite their conceptual simplicity, both SGD-DP and Adam-DP have to determine the amount of noise to guarantee (ε, δ) privacy. Abadi et al. (2016) proposed the moments accountant which we present in detail in Appendix E.

4.4. Our approach: Graph cuts for improved DP performance

We propose a simple yet effective treatment of the discrepancy between GCN training (that is, taking the entire graph as a single example to maximally utilize the contextual information of each node) and DP-version of SGD and Adam (which requires a set of i.i.d. examples to form batches and 'lots' in order to distribute DP noise effectively).

The unit for which our method provides a DP guarantee is a full graph, including all of its nodes and edges, which contrasts with other notions of DP for graphs such as Edge DP and Node DP (Kasiviswanathan et al., 2013), which only protect edges, or nodes with all of their adjacent edges, respectively. As DP operates with the notion of 'neighboring datasets' (Desfontaines and Pejó, 2020, Sec. 4), training a GCN privately on the full graph means that *any other graph* is neighboring. It also implies that each individual's privacy in that graph is protected, which is *the* goal of differential privacy. The other extreme would be to completely ignore the graph structure and train GCN on individual nodes; using DP, it would again protect each individual's privacy, but any advantage of graph structure would be ignored.

We thus propose a sweet-spot approach, that is splitting the graph into disconnected subgraphs. We experiment with different numbers of subgraphs to find the best trade-off. In order to avoid any further dataset queries that might require a larger privacy budget, we utilize random masking of the adjacency matrix so no additional DP mechanism is required.

Our algorithm creates a random index tensor for all nodes in the training set, which is then split into s groups, corresponding to the number of desired subgraphs. If the number of nodes n is divisible by s, then all subgraphs have equal sizes of nodes ($\frac{n}{s}$). If n is not divisible by s, then $n \bmod s$ subgraphs have $\lfloor \frac{n}{s+1} \rfloor$ nodes, while the rest have $\lfloor \frac{n}{s} \rfloor$. These indexes are then used to mask the original graph during training. This step is performed once during data preprocessing and requires very little additional computational time or memory requirements.

Privacy guarantee of graph cuts We start by summarizing the main DP argument of SGD-DP (Abadi et al., 2016). In particular, any two gradient vectors are made 'indistinguishable' from each other up to factor $\exp(\varepsilon)$ and summand δ. Gradients are computed over mini-batches. This means that the presence or absence of an individual in the mini-batch is protected by SGD-DP. Furthermore, mini-batches are disjoint, such that each individual is associated with a unique mini-batch only. The disjoint requirement stems from DP being defined through the notion of neighboring datasets, where a given individual's record in the dataset cannot appear in any of its neighboring datasets (see Section 2). SGD-DP with mini-batches is (ε, δ)-DP (Abadi et al., 2016). In our graph scenario, we cut the graph into several disconnected subgraphs that are equivalent to 'mini-batches'. When trained by SGD-DP, the gradients are again computed over each 'mini-batch' (subgraph) and privatized. Now having each individual (a single node and all its edges) in one 'mini-batch', the presence or absence of that individual is again protected by DP as desired. Therefore SGD-DP on GNN with disjoint subgraphs is (ε, δ)-DP.

Finally, our graph splitting algorithm is completely random. It does not query (in the DP sense) any information about the graph and its output is independent of a presence or absence of any individual. As such, the random graph splitting algorithm does not consume any privacy budget. Overall, this makes our approach (ε, δ)-DP.

Our method is thus easy to implement, does not require much computational overhead and fits very well into the DP scenario. We extensively compare our results using different subgraph sizes in the non-private and private settings in Section 6.

5. Experiments

5.1. Datasets

We are interested in a text classification use-case where documents are connected via undirected edges, forming a graph. While structurally limiting, this definition covers a whole range of applications. We perform experiments on five single-label multi-class classification tasks. The **Cora**, **Citeseer**, and **PubMed** datasets (Yang et al., 2016; Sen et al., 2008; McCallum et al., 2000; Giles et al., 1998) are widely used citation networks of research papers where citing a paper i from paper j creates an edge $i - j$. The task is to predict the category of the particular paper.

The **Reddit** dataset (Hamilton et al., 2017) treats the 'original post' as a graph node and connects two posts by an edge if any user commented on both posts. Given the large size of this dataset (230k nodes; all posts from Sept. 2014) causing severe computational challenges, we sub-sampled 10% of posts (only few days of Sept. 2014). The gold label corresponds to one of the top Reddit communities to which the post belongs to.

Unlike the previous English datasets, the **Pokec** dataset (Takac and Zabovsky, 2012; Leskovec and Krevl, 2014) contains an anonymized social network in Slovak. Nodes represent users and edges their friendship relations. User-level information contains many attributes in natural language (e.g., 'music', 'perfect evening'). We set up the following binary task: Given the textual attributes, predict whether a user prefers dogs or cats.[5]

[5]We decided against user profiling, namely age prediction

Pokec's personal information including friendship connections shows the importance of privacy-preserving methods to protect this potentially sensitive information. For the preparation details see Appendix D.

5.2. Experiment setup

We operate with three bench-marking scenarios. **Experiment A** is vanilla GCN without DP: The aim is to train the GCN without any privacy mechanism, evaluating also influence on performance with less training data. **Experiment B** is GCN with DP: We evaluate performance varying the amount of privacy budget as well as data size. We randomly sub-sample a certain percentage of nodes that then form a single training graph, as in standard GCN. The latter allows us to see the effects on performance of both adding noise and reducing training data. **Experiment C** is GCN with graph splits: Evaluating performance varying the number of graph splits in the non-DP and DP settings.

Implementation details. As the δ privacy parameter is typically kept 'cryptographically small' (Dwork and Roth, 2013) and, unlike the main privacy budget ε, has a limited impact on accuracy (Abadi et al., 2016, Fig. 4), we fixed its value to 10^{-5} for all experiments. The clipping threshold is set at 1. We validated our PyTorch implementation by fully reproducing the MNIST results from Abadi et al. (2016). We perform all experiments five times with different random seeds and report the mean and standard deviation. Early stopping is determined using the validation set. See Appendix A for more details on other hyperparameters.

6. Results and analysis

6.1. Experiment A: Non-private GCN

Table 1 shows the results on the left-hand side under 'Non-DP'. When trained with SGD, all datasets achieve fairly good results with the exception of PubMed, possibly due to PubMed having a much larger graph. The best of these is for Pokec, which could be due to its more expressive representations (BERT) and a simpler task (binary classification).

In comparison, in line with previous research (Ruder, 2016), Adam outperforms SGD in all cases, with Pokec showing the smallest gap (0.826 and 0.832 for SGD and Adam, respectively).

Figure 1 shows the non-DP results with increasing training data. We observe two contrasting patterns. First, there is a clear improvement as training data increases (e.g. CiteSeer, with 0.70 F1 score at 10% vs. 0.77 at 100%). Second, we observe the exact opposite pattern, with PubMed dropping from 0.57 at 10% to 0.49 at 100%, with a similar pattern for Pokec, or an early saturation effect for Reddit and Cora, where results do not increase beyond a certain point (at 20-30% for Reddit with approximately 0.69 F1 score, 50% for Cora at a score of 0.77). We speculate that, with a larger training size, a vanilla GCN has a harder time to learn the more complex input representations. In particular, for PubMed and Pokec, the increasing number of training nodes only partially increases the graph degree, so the model fails to learn expressive node representations when limited information from the node's neighborhood is available. By contrast, Reddit graph degree grows much faster, thus advantaging GCNs.

6.2. Experiment B: GCN with DP

The middle columns of Table 1 show results for a privacy budget of $\varepsilon = 2.0$. As discussed further below, without splitting the graph into subgraphs, it is impossible to reach the lower ε value of 1.0, since computations become very unstable due to the large amount of noise. We do not report values larger than $\varepsilon = 2.0$ as their privacy protection diminishes exponentially. We note four main patterns in this experiment.

First, **SGD-DP results stay the same, regardless of the noise value added.** This is quite unexpected, since higher added noise values would be anticipated to lead to lower results. One explanation for this pattern is that the gradients in vanilla SGD are already quite noisy, which may even help in generalization for the model, so the additional DP noise does not pose much difficulty beyond the initial drop in performance.

Second, **Adam-DP results outperform SGD-DP and can reach results close to the non-DP settings.** It is worth noting that, when using default hyperparameters with a moderate learning rate of 0.01, Adam-DP results are very low, usually worse than SGD-DP. It is only when optimizing this learning rate that we see substantial improvements, with the best-performing learning rates being very large, in the case of Reddit as high as 100. In contrast, SGD-DP does not see much benefit from additional hyperparameter optimization.

Third, we see **bigger drops in performance in the DP setting for datasets with simpler input representations.** Datasets of simpler input features can have results drop by more than half in comparison to the non-DP implementation. In comparison to SGD-DP, Adam-DP is able to retain better performance even with the simpler input features for Cora and PubMed (e.g., drop $0.79 \rightarrow 0.54$ for PubMed). Reddit and Pokec show the smallest drops from the non-DP to DP setting ($0.88 \rightarrow 0.72$ and $0.83 \rightarrow 0.66$ with Adam, for each dataset, respectively). In fact, even for SGD-DP, Pokec results are very close to the non-DP counterpart ($0.83 \rightarrow 0.75$ F1 score). Hence, Citeseer, Cora and PubMed, all using one-hot textual representations, show far greater drops in performance for the DP setting. The datasets utilizing GloVe (Reddit) and BERT (Pokec) representations perform far better. Since this effect of feature complexity on DP performance is shown only through different datasets, we perform additional experiments on Pokec using BoWs, fastText (Grave et al., 2018) and BERT features for a proper 'apples-to-apples'

for ad targeting (Perozzi and Skiena, 2015), for ethical reasons. Our task still serves well the demonstration purposes of text classification of social network data.

Non-DP				DP		DP split	
Maj.	SGD	Adam	ε	SGD	Adam	SGD	Adam
CiteSeer			1	-	-	0.35	0.36
0.18	0.77	0.79	2	**0.36**	**0.36**	0.35	**0.36**
Cora			1	-	-	0.55	0.56
0.32	0.77	0.88	2	0.39	0.52	0.55	**0.57**
PubMed			1	-	-	0.54	0.52
0.40	0.49	0.79	2	0.38	**0.54**	**0.54**	0.51
Pokec			1	-	-	0.62	0.72
0.50	0.83	0.83	2	**0.75**	0.66	0.64	0.73
Reddit			1	-	-	0.65	0.79
0.15	0.68	0.88	2	0.46	0.72	0.67	**0.82**

Table 1: F_1 results for experiments A, B and C: full dataset without DP (first three columns, including a majority baseline), with DP and varying ε (middle two columns), with DP using graph splits (right-most two columns). Best DP results are bold. Lower ε corresponds to better privacy.

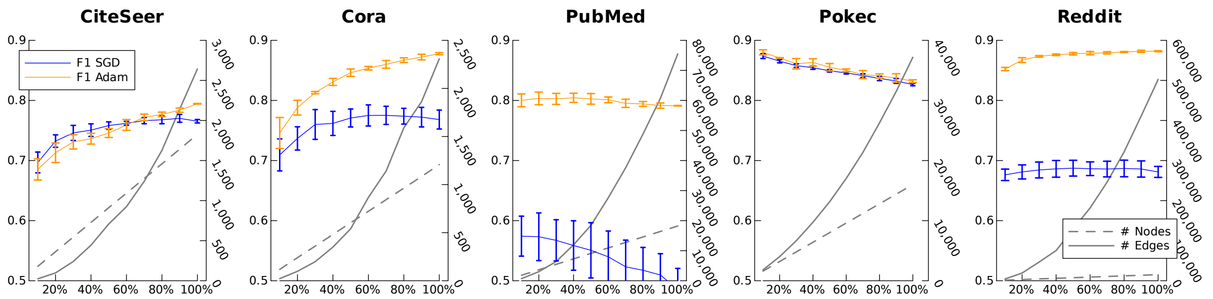

Figure 1: Experiment A: F_1 wrt. training data size (in %), without DP.

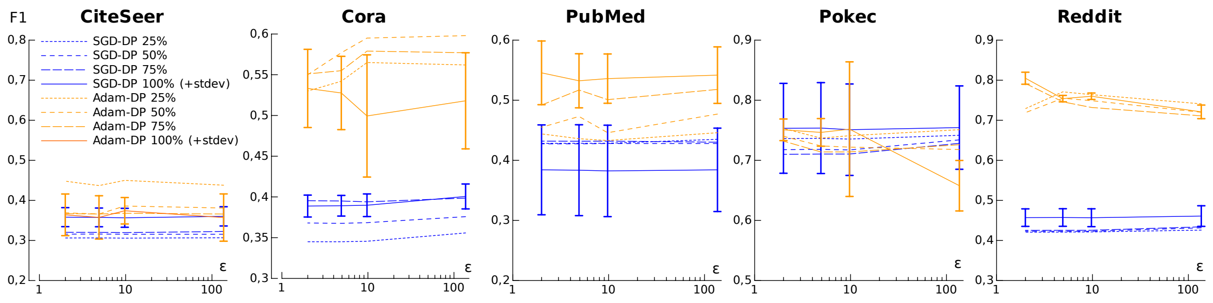

Figure 2: Experiment B: F_1 with varying training data size (in %) wrt. privacy budget ε, with DP.

comparison described in Section 6.4.

Learning Curves with DP Figure 2 shows the DP results both for varying ε and with different training sub-samples (25%, 50%, 75% and the full 100%). First, generally observed patterns are not the same for the learning curves in the non-DP setup (Experiment A). For instance, Adam exhibits the opposite pattern, e.g. Citeseer and Cora increase with more data for Adam without DP, but decrease for Adam-DP.

Second, we can see that **increasing the amount of data does not necessarily help in the DP setting**. For instance, while there is an improvement for SGD-DP with the Citeseer, Cora and Reddit datasets, results mostly get worse for Adam-DP, with the exception of PubMed. Hence, increasing training data generally does not act as a solution to the general drop in performance introduced by DP.

6.3. Experiment C: Graph splitting approach

First we highlight main results for the **non-DP graph splitting** approach. For all datasets, increasing the number of subgraph divisions yields better results in the SGD setting. This is especially notable in a case such as for PubMed, where there is an increase from 0.47 with no splits to 0.79 with a splits size of 100. Overall, splitting the graph is shown to be quite effective in a setting where the model may struggle more in the learning process, such as with SGD. Furthermore, at the higher subgraph split sizes, there is a slight drop-off for Adam

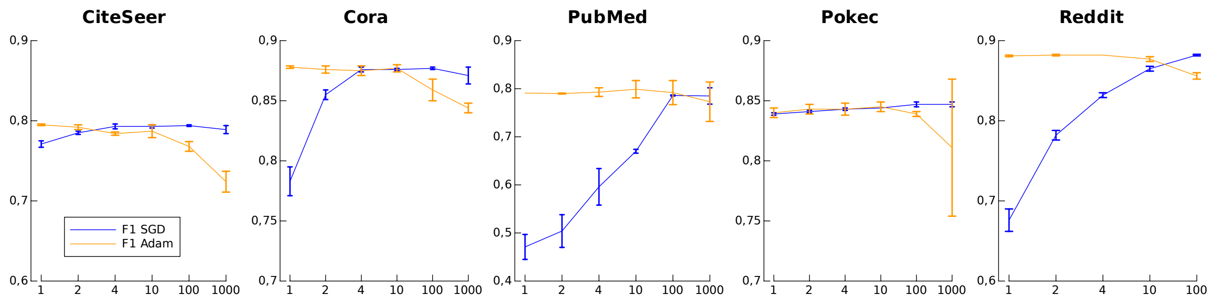

Figure 3: Experiment C, no DP: F_1 wrt. number of subgraphs.

but not for SGD, where increasing subgraph split size never improves performance beyond vanilla Adam, as shown in Figure 3.

Figure 4 shows the results for the **graph splitting setting with DP**, varying the privacy budget. First, increasing the number of **subgraph splits generally improves results** for SGD-DP (e.g. $0.36 \rightarrow 0.55$ for Cora at $\varepsilon = 2.0$, with 10 subgraphs vs. the full graph, respectively). We see a difference across datasets, where for instance Reddit shows the best results at 10 splits for all three ε values, while Citeseer or Pokec do not particularly benefit beyond four splits.

Second, this pattern of improvement is also noticeable in the case of Adam-DP, in contrast to the non-private vanilla Adam results. This is clearly seen in the Reddit results, where the very best result is with 10 splits with $\varepsilon = 1.0$ at an F-score of 0.79 with Adam-DP, being just 0.09 points lower than the non-DP version. As in Experiment B, it is notable that Adam-DP does not perform particularly well without using very high learning rates, with less graph splits requiring larger learning rate values. Overall, the best-performing number of subgraphs seems to be specific to the particular dataset used and can thus be treated as another hyperparameter to optimize on.

Third, with more subgraphs allowing for less noise to be added to obtain a lower ε value, it is possible to **reach the strong privacy guarantee** of $\varepsilon = 1.0$. For comparison, the randomized response, a DP mechanism extensively used by social scientists for decades (Warner, 1965), has $\varepsilon \approx 1.1$. Thus our graph splitting approach not only helps mitigate the difficulties of training with added DP noise, but also allows us to reach stronger privacy guarantees with about the same performance. Without the mini-batching that becomes possible by splitting the graph, it is impossible to carry out a stable computation during the moment accounting process to achieve $\varepsilon = 1.0$. A comparison of the DP setting with graph splitting (using our initial setup of 10 graph splits) and the regular DP setting is summarized in Table 1.

Summary and take-aways We summarize the key observations as follows:

1. SGD-DP is fairly robust to noise for these datasets and settings, even at $\varepsilon = 1.0$.
2. Adam-DP works even better than SGD-DP, however it needs to be tuned very carefully, using very high learning rates.
3. More complex representations are better for the DP setting, showing a smaller performance drop from the non-DP results.
4. Increasing training data does not necessarily mitigate negative performance effects of DP.
5. Graph splitting improves both performance and allows for a stronger privacy guarantee of $\varepsilon = 1.0$, resolving the mini-batching problem of GCNs in the DP setting.

We provide an additional error analysis in Appendix B, where we show that failed predictions in Reddit and CiteSeer are caused by 'hard cases', i.e. examples and classes that are consistently misclassified regardless of training data size or privacy budget. Moreover, in Appendix C we describe results on the MNIST dataset with varying lot sizes, showing how this hyperparameter affects model results.

6.4. Pokec Feature Comparison

As mentioned in Section 6.2, we perform additional experiments on the Pokec dataset in order to further investigate the hypothesis that input feature complexity has an effect on the degree of performance drop in the DP setting. We originally noticed this across separate datasets, with Cora, Citeseer and PubMed, using one-hot input features, having a greater performance drop than Reddit and Pokec, which use GloVe and BERT, respectively. In order to more properly evaluate this under the same conditions, we prepare two additional types of input features for the Pokec dataset, namely Bag of Words (BoWs) and fastText (Joulin et al., 2016), altogether having three levels of word representations, ranging from the simpler (BoWs) to more complex (BERT).

The BoWs embeddings were prepared by taking the same 20,000 user profiles as in the BERT preprocessing methodology described above. Tokens were split on whitespace, with additional steps such as punctuation removal and lowercasing. In order to reduce the embedding dimensionality, we filtered tokens by frequency in the interval $[15, 15000]$. Each user profile was thus

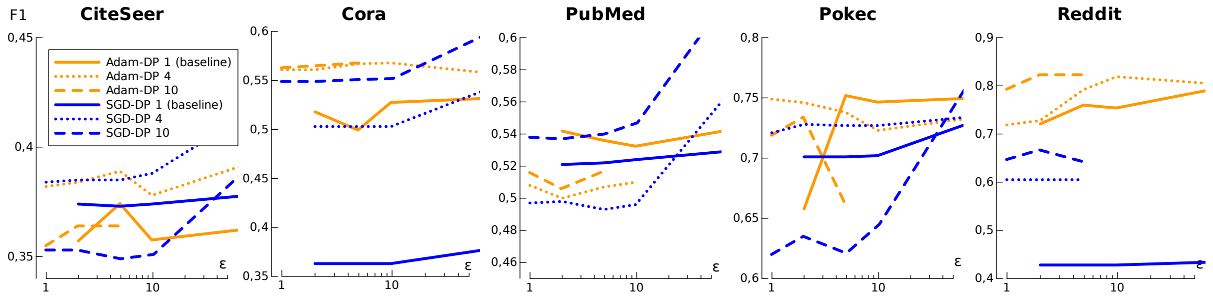

Figure 4: Experiment C, with DP: F_1 with varying number of subgraphs wrt. privacy budget ε.

represented with a 9447-dimensional vector of binary values.

For the fastText embeddings, we use a pre-trained model for Slovak from Grave et al. (2018). Using the same set of user profiles, we preprocess the data in the same manner as described by the authors. In order to have one vector per user profile, we average all fastText embeddings for a given user to have a final embedding dimension of 300.

The results of this experiment can be seen in Table 2. We notice that, in line with our hypothesis, the most effective input features in the DP setting are the BERT embeddings, with the smallest performance drop from the non-DP setting (e.g. $0.84 > 0.70$ for SGD-DP at $\varepsilon = 2.0$). Interestingly, the best method overall without DP is with the BoWs representation. One explanation for this is that a lot of slang vocabulary and unusual tokens are used in the social network data, which a fastText or BERT model may struggle with more, while BoWs would simply treat them equally as any other token in the vocabulary. As expected, the BoWs embeddings have a larger drop when trained with DP ($0.88 > 0.62$ for SGD and SGD-DP with $\varepsilon = 2.0$, respectively). The fastText results show the lowest performance both in the non-DP and DP settings, possibly due to the model struggling to maintain useful representations after averaging many token vectors for a user, which a more powerful model such as mBERT has an easier time with. Our original hypothesis is thus verified that more sophisticated input features such as BERT would show a smaller performance drop in the DP setting, compared to simpler representations such as BoWs, with this effect shown in an 'apples-to-apples' setting on the same dataset.

7. Conclusion

We have explored differentially-private training for GCNs, showing the nature of the privacy-utility trade-off. We show that an expected drop in results for the DP models can be mitigated by graph partitioning, utilizing Adam-DP, as well as having more complexity in the input representations. Our approach achieves strong privacy guarantees of $\varepsilon = 1.0$, yet reaching up to 87% and 90% of non-private F_1 scores for Pokec and Reddit datasets, respectively. An interesting line of future work is to explore further options for graph splitting that may take advantage of the graph structure instead of uninformed random splitting. By adapting global DP to a challenging class of deep learning networks, we are thus a step closer to flexible and effective privacy-preserving NLP.

Non-DP F_1 scores		**DP F_1 scores**		
SGD	Adam	ε	SGD	Adam
BoWs		2	0.62	0.63
0.88	0.87	5	0.62	0.61
		10	0.62	0.61
		137	0.63	0.63
fastText		2	0.59	0.63
0.71	0.73	5	0.59	0.57
		10	0.59	0.57
		137	0.61	0.60
BERT		2	0.70	0.66
0.84	0.84	5	0.70	0.75
		10	0.70	0.75
		137	0.74	0.75

Table 2: F_1 scores for the Pokec dataset, comparing different input feature representations and privacy budgets.

Acknowledgments

This research work has been funded by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. Calculations were conducted on the Lichtenberg high performance computer of the TU Darmstadt.

8. Bibliographical References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, Vienna, Austria. ACM.

Abdalla, M., Abdalla, M., Rudzicz, F., and Hirst, G. (2020). Using word embeddings to improve the privacy of clinical notes. *Journal of the American Medical Informatics Association*, 27(6):901–907.

Bui, T. N. and Jones, C. (1992). Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters*, 42(3):153–159.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. (2019). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266.

Coavoux, M., Narayan, S., and Cohen, S. B. (2018). Privacy-preserving Neural Representations of Text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Brussels, Belgium. Association for Computational Linguistics.

De Cao, N., Aziz, W., and Titov, I. (2019). Question Answering by Reasoning Across Documents with Graph Convolutional Networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2306–2317, Minneapolis, Minnesota. Association for Computational Linguistics.

Desfontaines, D. and Pejó, B. (2020). SoK: Differential privacies. *Proceedings on Privacy Enhancing Technologies*, 2020(2):288–313.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dwork, C. and Roth, A. (2013). The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407.

Giles, C. L., Bollacker, K. D., and Lawrence, S. (1998). CiteSeer: An Automatic Citation Indexing System. In Ian H Witten, et al., editors, *Proceedings of the third ACM conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, USA. ACM Press.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Hamilton, W., Ying, R., and Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. In I. Guyon, et al., editors, *Advances in Neural Information Processing Systems 30*, pages 1024–1034, Long Beach, CA, USA. Curran Associates, Inc.

Hitaj, B., Ateniese, G., and Perez-Cruz, F. (2017). Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, page 603–618, Dallas, TX, USA. Association for Computing Machinery.

Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B. C., and Roth, A. (2014). Differential Privacy: An Economic Method for Choosing Epsilon. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 398–410. IEEE.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Karypis, G. and Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392.

Kasiviswanathan, S. P., Nissim, K., Raskhodnikova, S., and Smith, A. (2013). Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*, pages 457–476. Springer.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In Yoshua Bengio et al., editors, *Proceedings of the 3rd International Conference for Learning Representations*, San Diego, CA.

Kipf, T. N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of International Conference on Learning Representations (ICLR 2017)*, pages 1–14, Toulon, France.

Lee, J. and Clifton, C. (2011). How Much Is Enough? Choosing ϵ for Differential Privacy. In Xuejia Lai, et al., editors, *Proceedings of the 14th Information Security Conference (ISC 2011)*, pages 325–340, Xi'an, China. Springer Berlin / Heidelberg.

Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June.

Li, Y., Baldwin, T., and Cohn, T. (2018). Towards Robust and Privacy-preserving Text Representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 25–30, Melbourne, Australia. Association for Computational Linguistics.

Lyu, L., Li, Y., He, X., and Xiao, T. (2020). Towards Differentially Private Text Representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1813–1816, Virtual conference. ACM.

Marcheggiani, D., Bastings, J., and Titov, I. (2018). Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans, Louisiana. Association for Computational Linguistics.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore,

K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. (2018). Learning Differentially Private Recurrent Language Models. In *Proceedings of the 6th International Conference on Learning Representations*, pages 1–14, Vancouver, BC, Canada.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Perozzi, B. and Skiena, S. (2015). Exact Age Prediction in Social Networks. In *Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion*, pages 91–92, Florence, Italy. ACM Press.

Rahimi, A., Cohn, T., and Baldwin, T. (2018). Semi-supervised User Geolocation via Graph Convolutional Networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2009–2019, Melbourne, Australia. Association for Computational Linguistics.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3980–3990, Hong Kong, China. Association for Computational Linguistics.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Sachan, D. S., Zhang, Y., Qi, P., and Hamilton, W. (2020). Do syntax trees help pre-trained transformers extract information? *arXiv preprint arXiv:2008.09084*.

Sajadmanesh, S. and Gatica-Perez, D. (2020). When Differential Privacy Meets Graph Neural Networks. *arXiv preprint*.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective Classification in Network Data. *AI Magazine*, 29(3):93.

Song, C. and Shmatikov, V. (2019). Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–206.

Takac, L. and Zabovsky, M. (2012). Data analysis in public social networks. In *International scientific conference and international workshop present day trends of innovations*, Łomża, Poland.

Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How Powerful are Graph Neural Networks? In *Proceedings of International Conference on Learning Representations (ICLR 2019)*, pages 1–17, New Orleans, Louisiana.

Xu, J., Gan, Z., Cheng, Y., and Liu, J. (2020). Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031, Online. Association for Computational Linguistics.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. (2016). Revisiting Semi-Supervised Learning with Graph Embeddings. In Maria Florina Balcan et al., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 40–48, New York, NY, USA. PMLR.

Zheng, C. and Kordjamshidi, P. (2020). SRLGRN: Semantic role labeling graph reasoning network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8881–8891, Online. Association for Computational Linguistics.

A. Hyperparameter Configuration

Our GCN model consists of 2 layers, with ReLU non-linearity, a hidden size of 32 and dropout of 50%, trained with a learning rate of 0.01 (apart from Adam-DP, which required far higher learning rates, as mentioned below). We found that early stopping the model works better for the non-DP implementations, where we used a patience of 20 epochs. We did not use early stopping for the DP configuration, which shows better results without it. For all SGD runs we used a maximum of 2000 epochs, while for Adam we used 500.

Importantly, for Adam-DP we noticed that more moderate learning rate values such as 0.01 were insufficient and led to far lower performance. We therefore optimized this at several values in the interval from 0.1 to 100, with some datasets and graph split values requiring learning rates as low as 0.1 (e.g. most datasets with 100 graph splits), while in other cases requiring 50 or 100 (e.g. Reddit for most graph split values).

Due to the smaller amount of epochs for Adam, it is possible to add less noise to achieve a lower ε value. Table 3 shows the mapping from noise values used for each optimizer to the corresponding ε in the full graph setting. The runtimes for our experiments reach up to 1 hour for the larger configurations on an NVIDIA A100 GPU.

Table 3: ε values from experiment B, with the corresponding noise values added to the gradient for each optimizer.

ε	**Noise-SGD**	**Noise-Adam**
136.51	4	2
9.75	26	13
4.91	48	24
2.00	112	56

Finally, regarding hyperparameter optimization on the validation set in the DP setting, Abadi et al. (2016) mention that, when optimizing on a very high number of parameter settings (e.g. in the thousands), this would additionally take up a moderate privacy budget (e.g. $\varepsilon = 4$ if they had used 6,700 hyperparameters). For our experiments, this number of hyperparameters is comparatively minimal and would be well within our privacy bounds.

B. Are 'hard' examples consistent between private and non-private models?

To look further into the nature of errors for experiments A and B, we evaluate the 'hard cases'. These are cases that the model has an incorrect prediction for with the maximum data size and non-private implementation (the first set of results of experiment A). For the experiment A learning curves, we take the errors for every setting of the experiment (10% training data, 20%, and so forth) and calculate the intersection of those errors with that of the 'hard cases' from the baseline implementation. This intersection is then normalized by the original number of hard cases to obtain a percentage value. The results for experiment A can be seen in Figure 5. We perform the same procedure for experiment B with different noise values for the SGD-DP setting, as seen in Figure 6. This provides a look into how the nature of errors differs among these different settings, whether they stay constant or become more random as we decrease the training size or increase DP noise.

Regarding the errors for experiment B, we can see a strong contrast between datasets such as Reddit and PubMed. For the latter, the more noise we add as ε decreases, the more random the errors become. In the case of Reddit, however, we see that even if we add more noise, it still fails on the same hard cases. This means that there are hard aspects of the data that remain constant throughout. For instance, out of all the different classes, some may be particularly difficult for the model.

Although the raw data for Reddit does not have references to the original class names and input texts, we can still take a look into these classes numerically and see which ones are the most difficult in the confusion matrix. In the baseline non-DP model, we notice that many classes are consistently predicted incorrectly. For example, class 10 is predicted 93% of the time to be class 39. Class 18 is never predicted to be correct, but 95% of the time predicted to be class 9. Class 21 is predicted as class 16 83% of the time, and so forth. This model therefore mixes up many of these classes with considerable confidence.

Comparing this with the confusion matrix for the differentially private implementation at an ε value of 2, we can see that the results incorrectly predict these same classes as well, but the predictions are more spread out. Whereas the non-private model seems to be very certain in its incorrect prediction, mistaking one class for another, the private model is less certain and predicts a variety of incorrect classes for the target class.

For the analysis of the hard cases of experiment A in Figure 5, we can see some of the same patterns as above, for instance between PubMed and Reddit. Even if the training size is decreased, the model trained on Reddit still makes the same types of errors throughout. In contrast, as training size is decreased for PubMed, the model makes more and more random errors. The main difference between the hard cases of the two experiments is that, apart from Reddit, here we can see that for all other datasets the errors become more random as we decrease training size. For example, Cora goes down from 85% of hard cases at 90% training data to 74% at 10% training data. In the case of experiment B, they stay about the same, for instance Cora retains just over 70% of the hard cases for all noise values.

Overall, while we see some parallels between the hard cases for experiments A and B with respect to patterns of individual datasets such as Reddit and PubMed, the general trend of more and more distinct errors that is

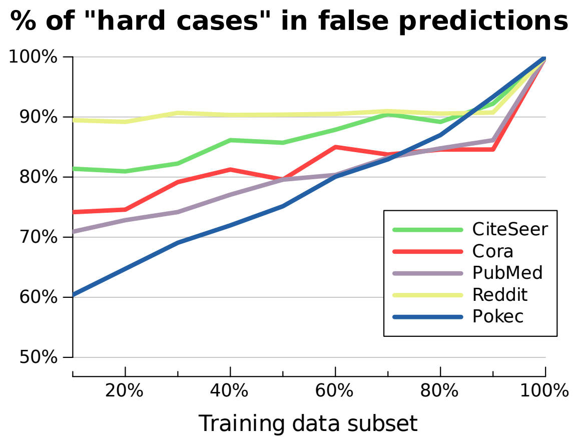

Figure 5: Hard cases in non-DP.

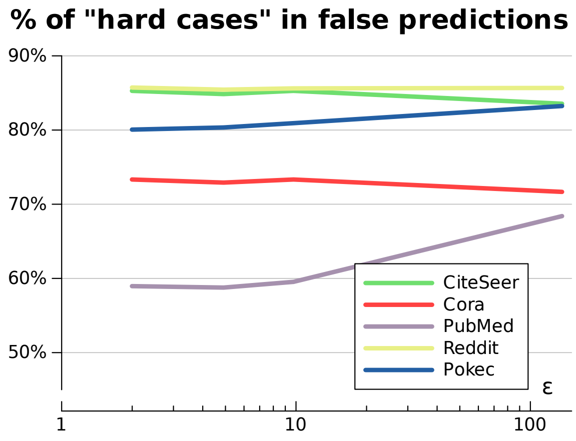

Figure 6: Hard cases analysis DP.

seen for the majority of datasets with less training size in experiment A is not the same in experiment B, staying mostly constant across different noise values for the latter. The idea that the nature of errors for DP noise and less training data being the same is thus not always the case, meaning that simply increasing training size may not necessarily mitigate the effects of DP noise.

C. MNIST Baselines

Table 4 shows results on the MNIST dataset with different lot sizes and noise values, keeping lot and batch sizes the same. We use a simple feed-forward neural network with a hidden size of 512, dropout of 50%, SGD

Table 4: Results on the MNIST dataset with varying lot sizes and noise values.

Lot Size	Noise	ε	F_1	Std.
600	4	1.26	0.90	0.02
6,000	4	4.24	0.84	0.01
60,000	4	15.13	0.45	0.04
60,000	50	0.98	0.39	0.15
60,000	100	0.50	0.10	0.01

optimizer, and a maximum of 2000 epochs with early stopping of patience 20, with other hyperparameters such as learning rate being the same as above. We note that the configuration in the first row with lot size of 600 and noise 4 is the same as described by Abadi et al. (2016) in their application of the moments accountant, reaching the same ε value of 1.2586.

We can see some important patterns in these results that relate to our main results from the GCN experiments. Maintaining a constant noise of 4, as we increase the lot size, not only does the ε value increase, but we see a dramatic drop in F_1 score, especially for a lot size of 60,000, being the full training set. If we try to increase the noise and maintain that 60,000 lot size, while we are able to lower the ε value below 1, the F_1 score continues to drop dramatically, going down to 0.1010 with a noise value of 100.

Hence, the current MNIST results further show the benefits of applying the graph splitting methodology on large one-graph datasets. By splitting the graph, we are able to utilize batches and lots of smaller sizes, allowing to add less noise to reach a lower epsilon value, ultimately retaining a higher F_1 score in the DP setting.

The four English datasets adapted from the previous work are only available in their encoded form. For the citation networks, each document is represented by a bag-of-words encoding.

The Reddit dataset combines GloVe vectors (Pennington et al., 2014) averaged over the post and its comments. Only the Pokec dataset is available as raw texts, so we opted for multilingual BERT (Devlin et al., 2019) and averaged all contextualized word embeddings over each users' textual attributes.[6] The variety of languages, sizes, and different input encoding allows us to compare non-private and private GCNs under different conditions. Table 5 summarizes data sizes and number of classes.

Table 5: Dataset statistics; size is number of nodes.

Dataset	Classes	Test size	Training size
CiteSeer	6	1,000	1,827
Cora	7	1,000	1,208
PubMed	3	1,000	18,217
Pokec	2	2,000	16,000
Reddit	41	5,643	15,252

D. Further details on Pokec dataset pre-processing

In order to prepare the binary classification task for the Pokec dataset, the original graph consisting of 1,632,803 nodes and 30,622,564 edges is sub-sampled to only include users that filled out the 'pets' column and had either cats or dogs as their preference, discarding entries with multiple preferences. For each pet type, users

[6]Sentence-BERT (Reimers and Gurevych, 2019) resulted in lower performance. Users fill in the attributes such that the text resembles a list of keywords rather than actual discourse.

were reordered based on percent completion of their profiles, such that users with most of the information were retained.

For each of the two classes, the top 10,000 users are taken, with the final graph consisting of 20,000 nodes and 32,782 edges. The data was split into 80% training, 10% validation and 10% test partitions.

The textual representations themselves were prepared with 'bert-multilingual-cased' from Huggingface transformers,[7] converting each attribute of user input in Slovak to BERT embeddings with the provided tokenizer for the same model. Embeddings are taken from the last hidden layer of the model, with dimension size 768. The average over all tokens is taken for a given column of user information, with 49 out of the 59 original columns retained. The remaining 10 are left out due to containing less relevant information for textual analysis, such as a user's last login time. To further simplify input representations for the model, the average is taken over all columns for a user, resulting in a final vector representation of dimension 768 for each node in the graph.

E. Moments Accountant in Detail

SGD-DP introduces two features, namely (1) a reverse computation of the privacy budget, and (2) tighter bounds on the composition of multiple queries. First, a common DP methodology is to pre-determine the privacy budget (ε, δ) and add random noise according to these parameters. In contrast, SGD-DP does the opposite: Given a pre-defined amount of noise (hyperparameter of the algorithm), the privacy budget (ε, δ) is computed retrospectively. Second, generally in DP, with multiple executions of a 'query' (i.e. a single gradient computation in SGD), we can simply sum up the ε, δ values associated with each query, such that for k queries with privacy budget (ε, δ), the overall algorithm is $(k\varepsilon, k\delta)$-DP. However, this naive composition leads to a very large privacy budget as it assumes that each query used up the maximum given privacy budget.

The simplest bound on a continuous random variable Z, the Markov inequality, takes into account the expectation $\mathbb{E}[Z]$, such that for $\varepsilon \in \mathbb{R}^+$:

$$\Pr[Z \geq \varepsilon] \leq \frac{\mathbb{E}[Z]}{\varepsilon} \tag{6}$$

Using the Chernoff bound, a variant of the Markov inequality, on the privacy loss Z treated as a random variable (Eq. 2), we obtain the following formulation by multiplying Eq. 6 by $\lambda \in \mathbb{R}$ and exponentiating:

$$\Pr[\exp(\lambda Z) \geq \exp(\lambda \varepsilon)] \leq \frac{\mathbb{E}[\exp(\lambda Z)]}{\exp(\lambda \varepsilon)} \tag{7}$$

where $\mathbb{E}[\exp(\lambda Z)]$ is also known as the moment-generating function.

The overall privacy loss Z is composed of a sequence of consecutive randomized algorithms $X_1, \ldots, X_k$. Since all X_i are independent, the numerator in Eq. 7 becomes a product of all $\mathbb{E}[\exp(\lambda X_i)]$. Converting to log form and simplifying, we obtain

$$\Pr[Z \geq \varepsilon] \leq \exp\left(\sum_i \ln \mathbb{E}[\exp(\lambda X_i)] - \lambda \varepsilon\right) \tag{8}$$

Note the moment generating function inside the logarithmic expression. Since the above bound is valid for any moment of the privacy loss random variable, we can go through several moments and find the one that gives us the lowest bound.

Since the left-hand side of Eq. 8 is by definition the δ value, the overall mechanism is (ε, δ)-DP for $\delta = \exp(\sum_i \ln \mathbb{E}[\exp(\lambda X_i)] - \lambda \varepsilon)$. The corresponding ε value can be found by modifying 8:

$$\varepsilon = \frac{\sum_i \ln \mathbb{E}[\exp(\lambda X_i)] - \ln \delta}{\lambda} \tag{9}$$

The overall SGD-DP algorithm, given the right noise scale σ and a clipping threshold C, is thus shown to be $(O(q\varepsilon\sqrt{T}), \delta)$-differentially private using this accounting method, with q representing the ratio $\frac{L}{N}$ between the lot size L and dataset size N, and T being the total number of training steps. See (Abadi et al., 2016) for further details.

[7]https://github.com/huggingface/transformers