Breaking Gradient Alignment for Privacy Protection in Federated Learning

Anonymous submission

Abstract

Federated Learning (FL) has become a popular approach for privacy aware Machine Learning (ML) because data stays with each client while only gradients are shared with a central server. Despite this design, recent studies show that gradients can still reveal sensitive information, which puts user privacy at risk. In this work, we take a closer look at how gradient leakage actually happens and build a new defense method that directly targets this weakness. We introduce a key lock module that can be plugged into different model architectures. With this module, clients send only locked gradients to the server, while the key and the lock parameters stay private on each device. Without access to this private information, an attacker cannot reconstruct the original training data from the shared gradients, and any attempt to do so results in models that fail to make meaningful predictions. We explain the theory behind this protection and show why the key lock module breaks the link between gradients and raw data. We also validate our method through extensive experiments on widely used datasets and neural network architectures. The results show that our approach maintains strong model performance while providing reliable defense against gradient leakage attacks.

1 Introduction and Related Work

Deep Neural Network (DNN)s rely on large datasets, which raises privacy concerns in collaborative training. Federated Learning (FL) addresses this by keeping raw data local and sharing only gradients for aggregation (McMahan et al. 2016). However, gradients can still leak sensitive patterns, and recent works show that private samples can be reconstructed even in early training (Zhu, Liu, and Han 2019).

Defense strategies such as gradient perturbation (Zhu, Liu, and Han 2019; Sun et al. 2021; Fan et al. 2024), data obfuscation (Chamikara et al. 2021; Scheliga, Mäder, and Seeland 2022; Wu et al. 2024), and hybrid designs (Ren et al. 2024; Yadav et al. 2020; Wei et al. 2021) attempt to reduce leakage but often harm accuracy or add heavy computation. Prior studies highlight that gradient alignment, model architecture, and training setup strongly affect leakage success (Wei et al. 2020). DeepIP (Fan et al. 2021) further showed that altering normalization parameters can break information flow, suggesting a promising direction for defense.

Attack methods continue to evolve. Deep Leakage from Gradients (DLG) (Zhu, Liu, and Han 2019) and Inverting Gradient (IG) (Geiping et al. 2020) reconstruct samples by matching gradients. GIAS (Jeon et al. 2021), Generative Regression Neural Network (GRNN) (Ren, Deng, and Xie 2021), and Generative Gradient Leakage (GGL) (Li et al. 2022) enhance reconstruction using generative priors, making defense more challenging. Generative Generative Adversarial Network (GAN)-based attacks (Hitaj, Ateniese, and Perez-Cruz 2017; Wang et al. 2019) also demonstrate that gradient space alignment enables recovery of private data distribution.

Classical defenses include Differential Privacy (DP), Homomorphic Encryption (HE), and Secure Multi-Party Computation (MPC) (Li et al. 2020), but they often require high communication or reduce model stability. Noise above a threshold severely affects convergence (Zhu, Liu, and Han 2019), and per-sample privacy scheduling like FedCDP (Wei et al. 2021) remains resource intensive.

We argue that leakage becomes successful when gradient representations across clients remain aligned. Based on this insight, we propose FedKL, a key lock module that introduces controlled gradient misalignment through a private key at each client without altering the FL protocol. The module replaces trainable normalization parameters with values generated by a lightweight key driven network, breaking the link between gradients and private data during backpropagation.

2 Proposed Method

2.1 Theoretical Analysis on Proposed Leakage Defense

Proposition 1. *For a Convolutional Neural Network (CNN), embedding a key-lock module into the model can prevent the inheritance of the private input information through the gradient, where this is achieved by dropping the information of the input data in the gradient.*

Proof. Theoretical analysis is conducted to investigate how the key-lock module prevents the inheritance of private information throughout the gradient in a CNN. Unlike in the traditional Batch Normalization (BN) layer where λ and β are trainable parameters, the key-lock module generates them as the output of two fully-connected layers called lock,

which take a private input sequence called key as input. The proposed approach introduces new trainable parameters, namely ω, ϕ, and two biases b_2 and b_3. The network is defined as follows:

$$
\begin{aligned}
x &= input \\
k &= key \\
u &= \theta \otimes x + b_1 \\
\gamma &= \omega \cdot k + b_2 \\
\beta &= \phi \cdot k + b_3 \\
s &= BN_{\gamma,\beta}(u) \\
a &= ReLU(s) \\
a' &= flatten(a) \\
z &= \lambda \cdot a' + b_4 \\
\hat{y} &= softmax(z) \\
\mathcal{L} &= CE(y, \hat{y})
\end{aligned}
$$

The derivative of the loss function with respect to ω and ϕ are:

$$
\frac{\partial \mathcal{L}}{\partial \omega} = \frac{\partial \mathcal{L}}{\partial s}\frac{\partial s}{\partial \omega} = (\hat{y} - y) \cdot \lambda \circ sgn(a) \cdot k \cdot \hat{u} \tag{1}
$$

$$
\frac{\partial \mathcal{L}}{\partial \phi} = \frac{\partial \mathcal{L}}{\partial s}\frac{\partial s}{\partial \phi} = (\hat{y} - y) \cdot \lambda \circ sgn(a) \cdot k \tag{2}
$$

The gradient in the lock layer is found to be closely related to the key sequence and feature map, which makes it possible for an attacker to infer the gradient of the lock layer given the key and leak the input data. To address this issue, we propose to keep both the key sequence and the weight of the lock layer private in our proposed method, FedKL. Once embedding the key-lock module, then we have:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \hat{u}} &= \frac{\partial \mathcal{L}}{\partial s}\frac{\partial s}{\partial \hat{u}} = \frac{\partial \mathcal{L}}{\partial s} \cdot (\omega \cdot k + b_2) \\
&= (\hat{y} - y) \cdot \lambda \circ sgn(a) \cdot (\omega \cdot k + b_2)
\end{aligned} \tag{3}
$$

The key sequence k, lock layer's weight ω, and bias b_2 are considered confidential in our proposed FedKL. ϕ and b_3 are omitted in back-propagation. The derivative of s with respect to $\hat{u}$ equals γ. Therefore, in Eqn. 3, ϕ and b_3 are omitted due to the omission of β. The unknown γ and β make it impossible to infer the input information from the feature map s. The proposed key-lock module effectively prevents the feature map from inheriting the input information and carrying it throughout the gradient when the private key and parameter of the lock module are unknown to the attacker. □

2.2 Key-Lock Module

A BN layer typically succeeds a convolutional layer and encompasses two processing stages: normalization followed by scaling and shifting. The coefficients for scale and shift comprise two trainable parameters that are updated via back-propagation. Stemming from the theoretical examination earlier, we propose a key-lock module to defend against gradient leakage attacks in FL, referred to as FedKL. This module can be incorporated subsequent to the convolution-normalization block, wherein the scale factor ($\gamma \in \mathbb{R}^O$) and the shift bias ($\beta \in \mathbb{R}^O$) cease to be trainable parameters and instead become outputs of the key-lock module. Here, O denotes the number of output channels of the convolutional layer. The input of the convolutional layer is represented as $X_{conv} \in \mathbb{R}^{B \times C \times W \times H}$, with B denoting the batch size, C representing the number of channels, and W and H signifying the width and height of the input image, respectively. The weight in the convolutional layer is denoted as $W_{conv} \in \mathbb{R}^{C \times O \times K \times K}$, with K indicating the kernel size. We define the input key, $X_{key} \in \mathbb{R}^S$, as having a length of S, and the weight of the key-lock module as $W_{lock} \in \mathbb{R}^{S \times O}$. The bias is represented as b. Therefore, the embedding transformation $\mathcal{F}()$ can be formulated as:

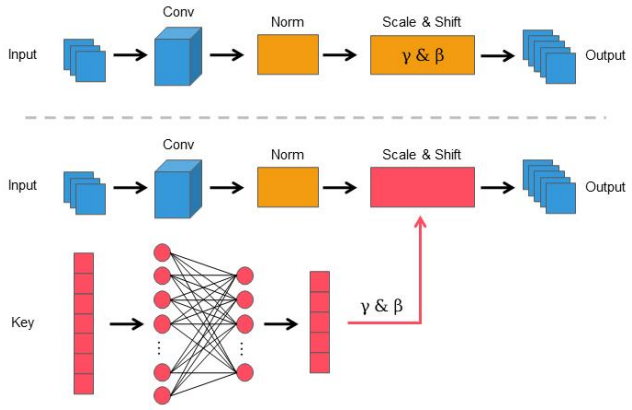


Figure 1: The top is the general convolution-normalization block, while the bottom indicates a key-lock module embedded in the block for generating the parameters, γ and β.

$$
\begin{aligned}
\mathcal{F}(X_{conv}, X_{key}) &= \gamma \cdot (X_{conv} \otimes W_{conv}) + \beta \\
\gamma &= X_{key} \cdot W_{lock-\gamma} + b_{lock-\gamma} \\
\beta &= X_{key} \cdot W_{lock-\beta} + b_{lock-\beta}
\end{aligned} \tag{4}
$$

where the convolution operation is denoted by $\otimes$, while the inner product is represented by $\cdot$. It is important to note that γ and β in the general normalization layer are trainable parameters; however, in our proposed methodology, they are the outputs of the lock layer, given the private key sequence. Figure 1 demonstrates a typical DNN building block, incorporating both a general convolution-normalization module and the proposed key-lock module.

2.3 FL with Key-Lock Module

It is crucial to emphasize that the proposed key-lock module solely modifies the DNN architecture and does not require alterations to the FL strategy. In a conventional FL system, the global model is initially instantiated on the server and subsequently distributed to each client for secure local training. Local gradients from clients are aggregated on the server, converging into a new global model for the subsequent round of local training. To implement the proposed key-lock module within the system, the server first initializes the model with the key-lock module and then distributes the model to each client. Individual clients generate their own

Table 1: Testing accuracy(%) from models trained at different settings. The red ones are the highest accuracy and the blue ones are the next highest. C-10 represents CIFAR-10 and C-100 is CIFAR-100. KL means key-lock module. The accuracy results in the Reference row are taken from the relevant papers.

Model		*LeNet* (32*32)	*ResNet-20* (32*32)		*ResNet-32* (32*32)		*ResNet-18* (224*224)		*ResNet-34* (224*224)		*VGG-16* (224*224)	
Dataset		MNIST	C-10	C-100	C-10	C-100	C-10	C-100	C-10	C-100	C-10	C-100
Centralized	**w/o KL**	98.09	91.63	67.59	92.34	70.35	91.62	72.15	92.20	73.21	89.13	63.23
	w/ KL	98.07	90.58	67.49	91.05	69.89	93.12	75.90	94.68	78.22	93.84	74.86
FL	**FedAvg**	98.14	91.20	58.58	91.37	61.91	89.50	68.59	89.27	68.30	88.13	61.91
	FedKL	97.45	88.45	61.97	89.29	64.17	91.66	74.19	93.27	76.61	93.54	73.06
Reference		99.05	91.25	-	92.49	-	-	-	-	-	-	-

private key sequences for local training. At each periodic aggregation stage, the gradient of the model, excluding the lock layer, is merged at the server to generate the new global model. Throughout the process, both the key sequence and lock layer of each client are maintained as strictly confidential and securely localized. The global model does not retain any information from a client's key-lock module; therefore, the global model and gradient from the local client cannot be employed for data reconstruction attacks. In the subsequent round of local training, each client first updates its own model using the distributed global model, without updating the lock layer. The client's lock layer is then integrated into the new model.

3 Experiments

3.1 Benchmarks and Metrics

Our experiments utilize four popular public benchmarks: MNIST (LeCun 1998), CIFAR-10 (Krizhevsky, Hinton et al. 2009), CIFAR-100 (Krizhevsky, Hinton et al. 2009) and ILSVRC2012 (Deng et al. 2009). Four evaluation metrics are employed to quantify the quality of the generated image, including Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al. 2018) and Structure Similarity Index Measure (SSIM) (Wang et al. 2004). To allow maximum capacity for gradient leakage attacks and hence to evaluate defense capability, the batch size is set to 1. Consequently, the reconstructed image and the true image can be directly matched. The proposed key-lock module is exclusively embedded in the first convolution-normalization block for all experiments in this study. The length of the randomly generated key sequence is 1024. The framework used to implement the neural network models is PyTorch (Paszke et al. 2019). The optimizer is Stochastic Gradient Descent (SGD) with an initial learning rate of 0.01, a decay of 5×10^{-4}, and a momentum of 0.9. The learning rate is reduced by a factor of 0.2 at epochs (rounds) 60, 120, and 160. The total number of training epochs (rounds) is 200

3.2 Accuracy Impact

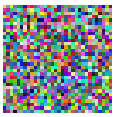
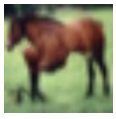
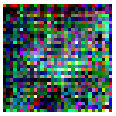
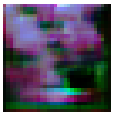
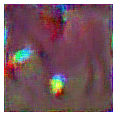
In order to assess the impact on performance, we devised two approaches: 1) centralized training with or without the key-lock module, and 2) collaborative training within the Federated Averaging (FedAvg) framework, also with or without the key-lock module. We employed *LeNet* (LeCun et al. 1998), *VGGNet* (Simonyan and Zisserman 2014), and *ResNet* (He et al. 2016) as the fundamental networks for training image classification models. A total of forty-four models were trained, utilizing eleven experimental configurations and two distinct training methodologies (centralized and federated), to compare standard networks and those incorporating key-lock modules. The findings are presented in Table 1.

Overall, the most favorable results are observed when centralized training is implemented. For instance, *LeNet* on MNIST yields the highest performance of 99.05% accuracy according to paper (LeCun et al. 1998), while *ResNet-20* achieves the maximum accuracy of 91.63% on CIFAR-10 and 67.59% on CIFAR-100. The addition of the key-lock module does not lead to a significant decline in model inference performance. In certain instances, models incorporating the key-lock module even outperform those without it. For example, *ResNet-20* on CIFAR-100 attains a testing accuracy of 61.97% within the FedKL framework, which is 5.79% higher than the conventional FedAvg model that achieves 58.58%. The most substantial performance enhancement is observed with *VGG-16* on CIFAR-100, where the FedKL model attains an accuracy of 73.06%, an improvement of 18.43% over the 61.69% achieved using FedAvg. Conversely, for experiments utilizing images with a resolution of $32 * 32$, optimal performance is obtained with centralized configurations without the key-lock module. Nevertheless, we note that the centralized setup incorporating the proposed key-lock module achieves the highest prediction accuracy when the resolution is increased to $224 * 224$. Moreover, the outcomes from *ResNet-18*, *ResNet-34*, and *VGG-16* indicate that models with the key-lock module significantly outperform those without it in both centralized and FL systems.

3.3 Defense Performance

To assess the defensive efficacy against gradient leakage attacks, three state-of-the-art attack methods were employed: DLG and GRNN. A batch size of 1 and resolution of $32 * 32$ were used for MNIST, CIFAR-10, and CIFAR-100. ILSVRC2012 utilized a resolution of $256 * 256$ instead of $224 * 224$, as GRNN is only capable of generating images with resolutions that are exponential multiples of 2. Images

Table 2: Comparison of image reconstruction using DLG and GRNN with the key-lock module. In this case, the malicious server has no knowledge of the private key sequence and gradient of the lock layer. "×" refers to a failure image reconstruction.

Model		*LeNet* (32*32)			*ResNet-20* (32*32)			*ResNet-18* (256*256)		
Dataset		MNIST	C-10	C-100	MNIST	C-10	C-100	C-10	C-100	ILSVRC
DLG	True				×	×	×	×	×	×
	w/o KL				×	×	×	×	×	×
	w/ KL				×	×	×	×	×	×
GRNN	True									
	w/o KL									
	w/ KL									

with lower resolution were upsampled using linear interpolation.

In Table 2, qualitative results are presented to demonstrate the comparative performance of defense efficacy against targeted leakage attacks, including DLG and GRNN, with various backbone networks and benchmark datasets. DLG consistently fails when employing *ResNet-20* and *ResNet-18*, and as a result, no data is presented. In the case of utilizing *LeNet*, the reconstructed image is entirely perturbed once the key-lock module is integrated into the network. When a BN layer or key-lock module was incorporated into *LeNet*, DLG failed to reconstruct accurate images in all experiments, as DLG directly regresses image pixels by approximating the gradient. In other words, DLG can only reconstruct the true image from a gradient with explicit input data information. However, both the BN layer and key-lock module re-normalize feature maps, leading to latent space misalignment and input data information ambiguity within the gradient. Nevertheless, the proposed key-lock module can still effectively prevent true image leakage from gradient-based reconstruction. When employing GRNN on *ResNet-18* with an image resolution of $256 * 256$, the reconstructed image is not recognizable for private data identification, even though the reconstructed image contains minimal, albeit visible, information from the true image.

4 Conclusion

In this paper, we provided comprehensive theoretical analysis on gradient leakage, based on which we proposed a gradient leakage defense method, called FedKL, for the FL system. The proposed key-lock module prevents the input data information from leaking through the gradient during the training stage. Importantly, we theoretically demonstrated the efficacy of the key-lock module in defending against gradient leakage attacks. Extensive empirical studies were conducted with three state-of-the-art attack methods. All the qualitative comparison results indicate that by incorporating the proposed key-lock module into the model, reconstructing input images from the publicly shared gradient is no longer feasible. Additionally, we discussed the influence of the key-lock module on the model in terms of classification accuracy. The implementation of our proposed FedKL is made publicly available to ensure consistent replication and facilitate further comparison for researchers in related fields.

References

Chamikara, M. A. P.; Bertok, P.; Khalil, I.; Liu, D.; and Camtepe, S. 2021. Privacy preserving distributed machine learning with federated learning. *Computer Communications*, 171: 112–125.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Fan, L.; Ng, K. W.; Chan, C. S.; and Yang, Q. 2021. DeepIP: Deep Neural Network Intellectual Property Protection with Passports. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Fan, M.; Liu, Y.; Chen, C.; Wang, C.; Qiu, M.; and Zhou, W. 2024. Guardian: Guarding against Gradient Leakage with Provable Defense for Federated Learning. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 190–198.

Geiping, J.; Bauermeister, H.; Dröge, H.; and Moeller, M. 2020. Inverting Gradients–How easy is it to break privacy in federated learning? *arXiv:2003.14053*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *IEEE CVPR*, 770–778.

Hitaj, B.; Ateniese, G.; and Perez-Cruz, F. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In *ACM CCS*, 603–618.

Jeon, J.; Lee, K.; Oh, S.; Ok, J.; et al. 2021. Gradient inversion with generative image prior. *Advances in Neural Information Processing Systems*, 34.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *Citeseer*.

LeCun, Y. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86: 2278–2324.

Li, Y.; Zhou, Y.; Jolfaei, A.; Yu, D.; Xu, G.; and Zheng, X. 2020. Privacy-preserving federated learning framework based on chained secure multiparty computing. *IEEE Internet of Things Journal*, 8: 6178–6186.

Li, Z.; Zhang, J.; Liu, L.; and Liu, J. 2022. Auditing Privacy Defenses in Federated Learning via Generative Gradient Leakage. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10132–10142.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; et al. 2016. Communication-efficient learning of deep networks from decentralized data. *arXiv:1602.05629*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Neural Information Processing Systems*, 32: 8026–8037.

Ren, H.; Deng, J.; and Xie, X. 2021. GRNN: Generative Regression Neural Network–A Data Leakage Attack for Federated Learning. *ACM Transactions on Intelligent Systems and Technology*.

Ren, H.; Deng, J.; Xie, X.; Ma, X.; and Wang, Y. 2024. FedBoosting: Federated learning with gradient protected boosting for text recognition. *Neurocomputing*, 569: 127126.

Scheliga, D.; Mäder, P.; and Seeland, M. 2022. PRECODE-A Generic Model Extension to Prevent Deep Gradient Leakage. In *Winter Conference on Applications of Computer Vision*, 1849–1858.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*.

Sun, J.; Li, A.; Wang, B.; Yang, H.; Li, H.; and Chen, Y. 2021. Soteria: Provable Defense Against Privacy Leakage in Federated Learning From Representation Perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9311–9319.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13: 600–612.

Wang, Z.; Song, M.; Zhang, Z.; Song, Y.; Wang, Q.; and Qi, H. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, 2512–2520. IEEE.

Wei, W.; Liu, L.; Loper, M.; Chow, K.-H.; Gursoy, M. E.; Truex, S.; and Wu, Y. 2020. A framework for evaluating gradient leakage attacks in federated learning. *arXiv:2004.10397*.

Wei, W.; Liu, L.; Wut, Y.; Su, G.; and Iyengar, A. 2021. Gradient-leakage resilient federated learning. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, 797–807. IEEE.

Wu, J.; Hayat, M.; Zhou, M.; and Harandi, M. 2024. Concealing Sensitive Samples against Gradient Leakage in Federated Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 21717–21725.

Yadav, K.; Gupta, B. B.; Chui, K. T.; and Psannis, K. 2020. Differential privacy approach to solve gradient leakage attack in a federated machine learning environment. In *International Conference on Computational Data and Social Networks*, 378–385. Springer.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 586–595.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep Leakage from Gradients. *Advances in Neural Information Processing Systems*, 32: 14774–14784.

Reproducibility Checklist

Instructions for Authors:
This document outlines key aspects for assessing reproducibility. Please provide your input by editing this `.tex` file directly.

For each question (that applies), replace the "Type your response here" text with your answer.

Example: If a question appears as

```
\question{Proofs of all novel
claims are included}
{(yes/partial/no)}
Type your response here
```

you would change it to:

```
\question{Proofs of all novel
claims are included}
{(yes/partial/no)}
yes
```

Please make sure to:

- Replace ONLY the "Type your response here" text and nothing else.
- Use one of the options listed for that question (e.g., **yes**, **no**, **partial**, or **NA**).
- **Not** modify any other part of the `\question` command or any other lines in this document.

You can `\input` this .tex file right before `\end{document}` of your main file or compile it as a stand-alone document. Check the instructions on your conference's website to see if you will be asked to provide this checklist with your paper or separately.

1. General Paper Structure

1.1. Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA) yes

1.2. Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no) yes

1.3. Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes/no) yes

2. Theoretical Contributions

2.1. Does this paper make theoretical contributions? (yes/no) yes

If yes, please address the following points:

2.2. All assumptions and restrictions are stated clearly and formally (yes/partial/no) yes

2.3. All novel claims are stated formally (e.g., in theorem statements) (yes/partial/no) yes

2.4. Proofs of all novel claims are included (yes/partial/no) partial

2.5. Proof sketches or intuitions are given for complex and/or novel results (yes/partial/no) yes

2.6. Appropriate citations to theoretical tools used are given (yes/partial/no) yes

2.7. All theoretical claims are demonstrated empirically to hold (yes/partial/no/NA) yes

2.8. All experimental code used to eliminate or disprove claims is included (yes/no/NA) no

3. Dataset Usage

3.1. Does this paper rely on one or more datasets? (yes/no) yes

If yes, please address the following points:

3.2. A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA) NA

3.3. All novel datasets introduced in this paper are included in a data appendix (yes/partial/no/NA) NA

3.4. All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no/NA) NA

3.5. All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations (yes/no/NA) yes

3.6. All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available (yes/partial/no/NA) yes

3.7. All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing (yes/partial/no/NA) NA

4. Computational Experiments

4.1. Does this paper include computational experiments? (yes/no) yes

If yes, please address the following points:

4.2. This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting (yes/partial/no/NA) yes

4.3. Any code required for pre-processing data is included in the appendix (yes/partial/no) yes

4.4. All source code required for conducting and analyzing the experiments is included in a code appendix (yes/partial/no) yes

4.5. All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no) yes

4.6. All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no) yes

4.7. If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results (yes/partial/no/NA) yes

4.8. This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks (yes/partial/no) partial

4.9. This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics (yes/partial/no) yes

4.10. This paper states the number of algorithm runs used to compute each reported result (yes/no) yes

4.11. Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information (yes/no) yes

4.12. The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank) (yes/partial/no) yes

4.13. This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments (yes/partial/no/NA) partial