A Vertical Federated Learning Framework for Graph Convolutional Network

Xiang Ni[1] , **Xiaolong Xu**[1] , **Linjuan Lyu**[1] , **Changhua Meng**[1] , **Weiqiang Wang**[1]
[1]Ant Group
nixiang85@gmail.com,
lingjuanlvsmile@gmail.com{yiyin.xxl,changhua.mch,weiqiang.wwq}@antgroup.com

Abstract

Recently, Graph Neural Network (GNN) has achieved remarkable success in various real-world problems on graph data. However in most industries, data exists in the form of isolated islands and the data privacy and security is also an important issue. In this paper, we propose FedVGCN, a federated GCN learning paradigm for privacy-preserving node classification task under data vertically partitioned setting, which can be generalized to existing GCN models. Specifically, we split the computation graph data into two parts. For each iteration of the training process, the two parties transfer intermediate results to each other under homomorphic encryption. We conduct experiments on benchmark data and the results demonstrate the effectiveness of FedVGCN in the case of GraphSage.

1 Introduction

The protection of user privacy is an important concern in machine learning, as evidenced by the rolling out of the General Data Protection Regulation (GDPR) in the European Union (EU) in May 2018 ([Chi *et al.*, 2018]). The GDPR is designed to give users to protect their personal data, which motivates us to explore machine learning frameworks with data sharing while not violating user privacy ([Balle and Wang, 2018; Bonawitz *et al.*, 2019]).

Privacy is an important challenge when data aggregation and collaborative learning happens across different entities [Lyu *et al.*, 2020b; Lyu *et al.*, 2020a]. To address this privacy issue many endeavors have been taken in different directions, among which, two important techniques are differential privacy ([Acar *et al.*, 2018; Aono *et al.*, 2016]) and fully homomorphic encryption ([Zhang *et al.*, 2015]). Recent advance in fully homomorphic encryption (FHE) ([Yuan and Yu, 2013]) allows users to encrypt data with the public key and offload computation to the cloud. The cloud computes on the encrypted data and generates encrypted results. Without the secret key, cloud simply serves as a computation platform but cannot access any user information. This powerful technique has been integrated with deep learning in the pioneering work of convolutional neural network ([Lin *et al.*, 2013; Veidinger, 1960]), known as CryptoNets.

Moreover, federated Learning [McMahan *et al.*, 2017] provides a privacy-aware solution for scenarios where data is sensitive (e.g., biomedical records, private images, personal text and speech, and personally identifiable information like location, purchase etc.). Federated learning allows multiple clients to train a shared model without collecting their data. The model training is conducted by aggregating locally-computed updates and the data in clients will not be transferred to anywhere for data privacy. In vertical setting, two data sets share the same sample ID space but differ in feature space [Hardy *et al.*, 2017; Yang *et al.*, 2019]. For example, consider two different companies in the same city, one is a bank, and the other is an e-commerce company. Their user sets are likely to contain most of the residents of the area, so the intersection of their user space is large. Vertical federated learning on logistic regression, xgboost, multi-tasking learning, neural network, transfer learning, etc, have been previously studied ([Acar *et al.*, 2018; Mohassel and Zhang, 2017; Konečnỳ *et al.*, 2016; Nock *et al.*, 2018]). However, few research has studied how to train federated GNNs in a privacy-preserving manner when data are vertically partitioned, which popularly exists in practice. To fill in this gap, in this paper, we propose a vertical federated learning framework on graph convolutional network (GCN). In particular, we test our ideas on GraphSage ([Chen *et al.*, 2020b; Krizhevsky *et al.*, 2009]).

Our main contributions are the following:

- We introduce a vertical federated learning algorithm for graph convolutional network in a privacy-preserving setting to provide solutions for federation problems beyond the scope of existing federated learning approaches;
- We provide a novel approach which adopts additively homomorphic encryption (HE) to ensure privacy, while maintaining accuracy. Experimental results on three benchmark datasets demonstrate that our algorithm significantly outperforms the GNN models trained on the isolated data and achieves comparable performance with the traditional GNN trained on the combined plaintext data.

2 Preliminaries

2.1 Orthogonal Polynomials

The set of functions $\{\Psi_0, \Psi_1, ..., \Psi_n\}$ in the interval $[a, b]$ is a set of orthogonal functions with respect to a weight function w if

$$< \Psi_i, \Psi_j >= \int_a^b \Psi_i(x)\Psi_j(x)w(x)dx = \delta_{ij} \tag{1}$$

2.2 Least-Squares Approximation

Movivation: Suppose $f \in C[a, b]$, find a orthogonal polynomial $P_n(x)$ of degree at most n to approximate f such that $\int_a^b (f(x) - P_n(x))^2 dx$ is a minimum ([Ali, 2020; Carothers, 1998]).

Let polynomial $P_n(x)$ be $P_n(x) = \sum_{k=0}^{n} a_k x^k$ which minimizes the error

$$E = E(a_0, a_1, ..., a_n) = \int_a^b (f(x) - \sum_{k=0}^{n}(a_k x^k)^2 dx \tag{2}$$

The problem is to find $a_0, ..., a_n$ that will minimize E. The necessary condition for $a_0, ..., a_n$ to minimize E is $\frac{\partial E}{\partial a_j} = 0$, which gives the normal equations:

$$\sum_{k=0}^{n} a_k \int_a^b x^{j+k} dx = \int_a^b x^j f(x) dx \quad \text{for } j = 0, 1, ..., n. \tag{3}$$

We can now find a least-square polynomial approximation of the form

$$p_n(x) = \sum_{i=0}^{n} b_i \Psi_i(x) \tag{4}$$

where $\{\Psi_i\}_{i \in \{0,1...,n\}}$ is a set of orthogonal polynomials. The Legendre polynomials $\{P_0(x) = 1, P_1(x) = x, P_2(x) = \frac{3}{2}x^2 - \frac{1}{2}, ...$ is a set of orthogonal functions with respect to the weight function $w(x) = 1$ over the interval $[-1, 1]$. In this case, the coefficients of the least-squares polynomial approximation p_n are expressed as follows

$$b_i = \frac{1}{C_i} \int f(x)P_i(x)dx, \ i \in \{0, 1, ..., n\} \tag{5}$$

Based on this approach, we can approximate the ReLU function using a polynomial of degree two as

$$p(x) = \frac{4}{3\pi a}x^2 + \frac{1}{2}x + \frac{a}{2\pi} \tag{6}$$

where a is determined by the data you are working on.

2.3 Paillier Homomorphic Encryption

Homomorphic encryption allows secure computation over encrypted data. To cope with operations in deep neural nets (mainly multiplication and addition), we adopt a well-known partially homomorphic encryption system called *Paillier* ([Chen *et al.*, 2020a; Chi *et al.*, 2018]). Paillier homomorphic encryption supports unlimited number of additions between ciphertext, and multiplication between a ciphertext and a scalar constant. Given ciphertext $\{[[M_1]], [[M_2]], \cdots, [[M_g]]\}$ and scalar constants $\{s_1, s_2, \cdots, s_g\}$, we can calculate $([[M_1]] \otimes s_1) \oplus ([[M_2]] \otimes s_2) \oplus \cdots \oplus ([[M_g]] \otimes s_g)$. without knowing the plaintext message. Here, $[[M_g]]$ represents the ciphertext. $\oplus$ is the homomorphic addition with ciphertext and $\otimes$ is the homomorphic multiplication between a ciphertext and a scalar constant ([Barni *et al.*, 2006; Balle and Wang, 2018; Chi *et al.*, 2018]).

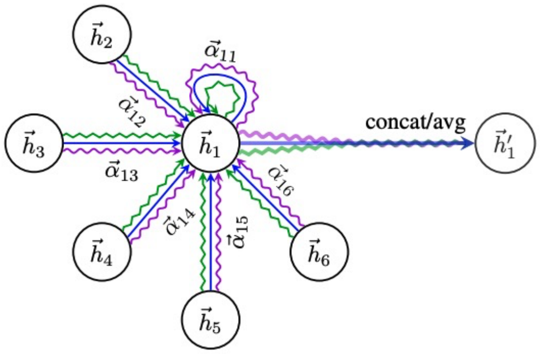

Figure 1: A Sample of GAT.

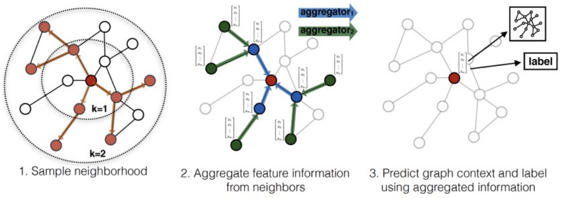
k=1
k=2
aggregator
aggregator
label
1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

Figure 2: A Sample of GraphSAGE.

2.4 Graph Convolutional Neural Networks

Graph convolutional network (GCN) [Kipf and Welling, 2017] generalizes the operation of convolution from grid data to graph data. The main idea is to generate a node v's representation by aggregating its own features x_v and neighbors' features x_u, where $u \in N(v)$. Different from Recurrent Graph Neural Network, GCN stacks multiple graph convolutional layers to extract high-level node representations. GCN plays a central role in building up many other complex GNN models ([Chen *et al.*, 2020b]). Moreover, many other popular algorithms, such as Graph attention networks (GAT, shown in Figure 1) [Veličković *et al.*, 2018], and Graph-SAGE (shown in Figure 2) [Will *et al.*, 2017]), typically perform well in graph tasks. However, GCN is the transductive method which is not suitable for large graphs.

The main difference between different GNN methods is the aggregation process. The general aggregation function can be written as follows:

$$\hat{h}_i^l = w_{i,i}^l \hat{h}_i^{l-1} + w_{i,j}^l \hat{h}_j^{l-1} \ (i \neq j) \tag{7}$$

Here, $\hat{h}_i^l$ is the hidden status in l-th layer for the i-th node. Different aggregation functions define different $w_{i,i}$ and $w_{i,j}$. In GraphSAGE method, the aggregation function is mean. And $w_{i,j}$ is calculated by adopting the attention mechanism.

3 Methodology

Additively homomorphic encryption and polynomial approximations have been widely used for privacy-preserving machine learning, and the trade-offs between efficiency and privacy by adopting such approximations have been discussed intensively [Ali, 2020]. Here we use a second order Taylor approximation for loss and gradients computations:

Define

$$L(w_1, w_2) = p(w_1 h_1 + w_2 h_2) \tag{8}$$

Then

$$\frac{\partial L}{\partial w_1} = \frac{8h}{3\pi a}(w_1 h_1 + w_2 h_2) * h_1 + \frac{1}{2} h_1 \tag{9}$$

$$\frac{\partial L}{\partial w_2} = \frac{8h}{3\pi a}(w_1 h_1 + w_2 h_2) * h_2 + \frac{1}{2} h_2 \tag{10}$$

Applying the Homomorphic encryption to L and $\frac{\partial L}{\partial w_i}$ gives

$$[[L]] = \frac{4}{3\pi a}[[(w_1 h_1 + w_2 h_2)^2]] + \frac{1}{2}[[(w_1 h_1 + w_2 h_2)]] + \frac{a}{2\pi} \tag{11}$$

$$[[L_A]] = \frac{4}{3\pi a}[[(w_1 h_1)^2]] + \frac{1}{2}[[(w_1 h_1]] + \frac{a}{4\pi} \tag{12}$$

$$[[L_B]] = \frac{4}{3\pi a}[[(w_2 h_2)^2]] + \frac{1}{2}[[(w_2 h_2]] + \frac{a}{4\pi} \tag{13}$$

$$[[L_{AB}]] = \frac{4}{3\pi a}[[2 w_1 h_1 w_2 h_2]] \tag{14}$$

$$[[L]] = [[L_A]] + [[L_B]] + [[L_{AB}]] \tag{15}$$

and

$$[[\frac{\partial L}{\partial w_1}]] = \frac{8h}{3\pi a}[[(w_1 h_1 + w_2 h_2) * h_1]] + \frac{1}{2}[[h_1]] \tag{16}$$

$$[[\frac{\partial L}{\partial w_2}]] = \frac{8h}{3\pi a}[[(w_1 h_1 + w_2 h_2) * h_2]] + \frac{1}{2}[[h_2]] \tag{17}$$

Here [[x]] is the ciphertext of x. The reason to use quadratic orthogonal polynomials to approximate/replace the relu activation is to preserve the multiplication/addition under homomorphic encryption.

Suppose that companies A and B would like to jointly train a machine learning model, and their business systems each have their own data. In addition, Company B also has label data that the model needs to predict. We call company A the passive party and the company B the active party. For data privacy and security reasons, the passive party and the active party cannot directly exchange data. In order to ensure the confidentiality of the data during the training process, a third-party collaborator C is involved, which is called the server party. Here we assume the server party is honest-but-curious and does not collude with the passive or the active party, but the passive and active party are honest-but-curious to each other. The passive party trusted the server party is a reasonable assumption since the server party can be played by authorities such as governments or replaced by secure computing node such as Intel Software Guard Extensions (SGX) ([Costan and Devadas, 2016]).

Algorithm 1 FedVGCN: Forward Propagation

```
Active Party
compute $w_1 * h_1$ and $N_1$, and send to the server party
Passive Party
compute $w_2 * h_2$ and $N_2$, and send to the server party
Server Party
compute $[[w_1 * h_1 + w_2 * h_2]]$ and decrypt them to the active party and the passive party
```

3.1 Unsupervised loss function

Existing FL algorithms are mainly for supervised learning. In order to learn useful, predictive representations in a fully unsupervised setting, we adopt the loss function in [Hamiltonm, 2017]. That is, the graph-based loss function encourages nearby nodes to have similar representations, while enforcing that the representations of disparate nodes are highly distinct:

$$J_{\mathcal{G}}(z_u) = -\log(\sigma(z_u^T z_v)) - Q \cdot \mathbb{E}_{V_N \ P_n(v)} \log(\sigma(-z_u^T z_{v_n})) \tag{18}$$

where v is a node that co-occurs near u on fixed-length random walk, σ is the sigmoid function, P_n is a negative sampling distribution, and Q defines the number of negative samples.

3.2 Algorithms

In isolated GNNs, both node features and edges are hold by different parties. The first step under vertically data split setting is secure ID alignment, also known as Private Set Intersection (PSI) [Pinkas *et al.*, 2014]. That is, data holders align their nodes without exposing those that do not overlap with each other. In this work, we assume data holders have aligned their nodes beforehand and are ready for performing privacy preserving GNN training.

In the case of GCN, the forward propagation and backward propagation of our proposed vertical federated algorithm are presented in Algorithm 1 and Algorithm 2 respectively. Here N_i is the number of edges at party i. The backward propagation algorithm can be illustrated by Figure 3.

4 Security Analysis

Theorem 4.1. *If the number of samples is much greater than the number of features, then the algorithm FedVGCN is secure.*

Proof. In the above protocol, the passive party learns its gradient at each step, but this is not enough for the passive party to learn any information from the active party, because the security of scalar product protocol is well-established based on the inability of solving n equations in more than n unknowns. Here we assume the number of samples N_A is much greater than n_A, where n_A is the number of features. Similarly, the active party can not learn any information from the passive party. Therefore the security of the protocol is proved. □

5 Complexity Analysis

5.1 Communication Cost

The communication cost is analyzed as the total number of messages transmitted between the client and server. We assume a unit message size for encrypted data. For an n-layer

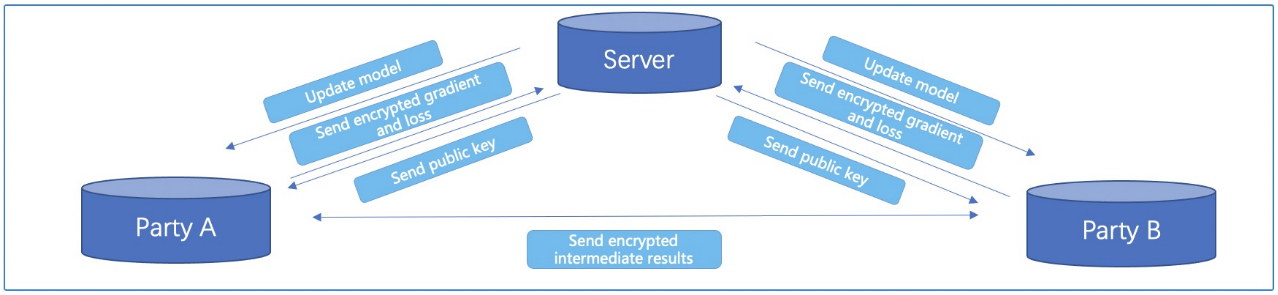
Server
Update model
Send encrypted gradient and loss
Send public key
Update model
Send encrypted gradient and loss
Send public key
Party A
Party B
Send encrypted intermediate results

Figure 3: FedVGCN: Backward Propagation

Algorithm 2 FedVGCN: Backward Propagation

```
Initialization: $w_1, w_2$
Input: learning rate $\eta$, data sample $x$
Server Party
create an encryption key pair, send public key to the active party and the passive party.
Passive Party
compute $[[w_1 h_1]], [[L_A]], [[N_1]]$, and send to Active Party, where $N_1$ is the number of neighborhood nodes of the passive party.
Active Party
compute $[[w_2 h_2]], [[L]]$, send $[[w_2 h_2]]$ to Passive Party and send $[[L]]$ to the Server Party.
Passive Party
Initialize random noise $\sigma_A$, compute $[[\frac{\partial L}{\partial w_1}]] + [[\sigma_A]]$ and send to the Server Party.
Active Party
Initialize random noise $\sigma_B$, compute $[[\frac{\partial L}{\partial w_2}]] + [[\sigma_B]]$ and send to the Server Party.
Server Party
The Server party decrypts $L$, send $\frac{\partial L}{\partial w_1} + \sigma_A$ to the passive party and $\frac{\partial L}{\partial w_2} + \sigma_B$ to the active party.
return
$w_1, w_2$
```

network, in the forward propagation, the communication cost is $2m(n-1)$, where m is the number of activations in a layer. This is because total m ciphertexts need to be transmitted by the client and the server, for the sum of inputs z_i and activations a_i, respectively, at each layer. In the backpropagation, the client needs at most $7 * (m^2 + m + 1)$ messages for model updates between two consecutive layers. Except the final layer, the client interacts with the server to calculate the gradients, which requires transmitting encrypted error $[[\delta_i + 1]]c$ in m messages between client and server for each layer. Summing up the costs from the forward and backpropagation, the entire network requires $O(nm^2)$ communication messages for an iteration.

5.2 Computation Cost

Arithmetic multiplications and additions are mapped to modular exponentiations and modular multiplications over ciphertext, respectively. Here, we denote such cost of conducting homomorphic arithemtics in Paillier by p. For n layers, both forward and backpropagations take $O(nm^2p)$ so the total computation cost is $O(nm^2p)$.

6 Experiments

For experiments, we mainly focus on GraphSage algorithm for illustration purpose (in this case, we use FedVGraphSage to denote FedVGCN). This part aims to answer the following questions:

- Q1: whether FedVGraphSage outperforms the GraphSage models that are trained on the isolated data.
- Q2: how does FedVGraphSage behave comparing with the centralized but insecure model that is trained on the plaintext mixed data.

6.1 Experimental Setup

We first describe the datasets, comparison methods, and parameter settings used in our experiments.

Datasets. We use three benchmark datasets which are popularly used to evaluate the performance of GNN, i.e., Cora, Pubmed, and Citeseer ([Balle and Wang, 2018]), as shown in Table 1. We split node features and edges randomly and assume they are hold by two parties.

Comparison methods. We compare FedVGraphSage with GraphSAGE models ([Chen *et al.*, 2020b]) that are trained using isolated data and mixed plaintext data respectively to answer **Q1** and **Q2**. For all the experiments, we split the datasets randomly, use five-fold cross validation and adopt accuracy as the evaluation metric.

Parameter settings. For the models which are trained on isolated data, we use relu as the activation function. For the deep neural network on server, we set the dropout rate to 0.5 and network structure as $(64, 64, |C|)$, where $|C|$ is the number of classes. We set the learning rate as 10^{-5}.

Dataset	#Node	#Edge	#Feature	#Classes
Cora	2708	5409	1433	7
Pubmed	19717	44338	500	3
Citeseer	3327	4732	3703	6

Table 1: Dataset statistics

Dataset	Cora	Pubmed	Citeseer
$GraphSage_A$	0.5222	0.6936	0.4630
$GraphSage_B$	0.4867	0.6801	0.5510
FedVGraphSage	0.6770	0.7830	0.6820
$GraphSage_{A+B}$	0.7080	0.7890	0.6983

Table 2: Accuracy comparison on three datasets

6.2 Comparison Results and Analysis

Answer to Q1: We compare FedVGraphSage with the GraphSages that are trained on the isolated feature and edge data, i.e., $GraphSage_A$ and $GraphSage_B$. From Table 2, we can find that, FedVGraphSage significantly outperforms $GraphSage_A$ and $GraphSage_B$ on all the three datasets. Take Cora for example, our FedVGraphSage improves $GraphSage_A$ and $GraphSage_B$ by as high as 29.6% and 39.1%, in terms of accuracy.

Analysis: $GraphSage_A$ and $GraphSage_B$ can only use partial feature and edge information hold by A and B, respectively. In contrast, FedVGraphSage provides a solution for A and B to train GraghSages collaboratively without compromising their own data. By doing this, FedVGraphSage can use the information from the data of both A and B simultaneously, and therefore achieve better performance.

Answer to Q2: We then compare FedVGraphSage with $GraphSage_{A+B}$ that is trained on the mixed plaintext data. It can be seen from Table 2 that FedVGraphSage has comparable performance with $GraphSage_{A+B}$, e.g., 0.6770 vs. 0.7080 on Cora dataset, 0.7830 vs. 0.7890 on Pubmed dataset, 0.6820 vs. 0.6983 on Citeseer dataset.

Analysis: First, we use Paillier Homomorphic Encryption for A and B to securely communicate intermidiate results, as described in Algorithm 1 and Algorithm 2. Second, we use the quadratic orthogonal polynomials activation functions to preserve the sum and multiplication operations which acts on the encrypted results. Therefore, FedVGraphSage has comparable performance with $GraphSage_{A+B}$.

7 Conclusion

In this work, we introduce a novel vertical federated learning algorithm for graph neural network. We adopt additively homomorphic encryption to ensure privacy, while maintaining accuracy. Experimental results on three benchmark datasets demonstrate that our algorithm significantly outperforms the GNN models trained on the isolated data and has comparable performance with the traditional GNN trained on the combined plaintext data. In this paper, we mainly investigate the vertical federated GNN by adopting GraphSAGE, and in the future, we will extend our method to more complex GNN models, such as GAT, GIN, etc. We also plan to apply our method to real industrial applications.

References

[Acar *et al.*, 2018] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):1–35, 2018.

[Ali, 2020] So Jinhyun Avestimehr A. Salman Ali, Ramy E. On polynomial approximations for privacy-preserving and verifiable relu networks. *arXiv preprint https://arxiv.org/abs/2011.05530*, 2020.

[Aono *et al.*, 2016] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Scalable and secure logistic regression via homomorphic encryption. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 142–144, 2016.

[Balle and Wang, 2018] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR, 2018.

[Barni *et al.*, 2006] Mauro Barni, Claudio Orlandi, and Alessandro Piva. A privacy-preserving protocol for neural-network-based computation. In *Proceedings of the 8th workshop on Multimedia and security*, pages 146–151, 2006.

[Bonawitz *et al.*, 2019] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[Carothers, 1998] Neal L Carothers. A short course on approximation theory. *Department of Mathematics and Statistics, Bowling green State University*, 1998.

[Chen *et al.*, 2020a] Chaochao Chen, Liang Li, Bingzhe Wu, Cheng Hong, Li Wang, and Jun Zhou. Secure social recommendation based on secret sharing. *arXiv preprint arXiv:2002.02088*, 2020.

[Chen *et al.*, 2020b] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. Fede: Embedding knowledge graphs in federated setting. *arXiv preprint arXiv:2010.12882*, 2020.

[Chi *et al.*, 2018] Jianfeng Chi, Emmanuel Owusu, Xuwang Yin, Tong Yu, William Chan, Patrick Tague, and Yuan Tian. Privacy partitioning: Protecting user data during the deep learning inference phase. *arXiv preprint arXiv:1812.02863*, 2018.

[Costan and Devadas, 2016] Victor Costan and Srinivas Devadas. Intel sgx explained. *IACR Cryptol. ePrint Arch.*, 2016(86):1–118, 2016.

[Hamiltonm, 2017] Ying RexLeskovec Jure Hamiltonm, William L. Inductive representation learning on large graphs. *arXiv preprint https://arxiv.org/abs/1706.02216*, 2017.

[Hardy *et al.*, 2017] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[Konečnỳ *et al.*, 2016] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

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

[Lin *et al.*, 2013] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[Lyu *et al.*, 2020a] Lingjuan Lyu, Han Yu, Xingjun Ma, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S Yu. Privacy and robustness in federated learning: Attacks and defenses. *arXiv preprint arXiv:2012.06337*, 2020.

[Lyu *et al.*, 2020b] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[Mohassel and Zhang, 2017] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38. IEEE, 2017.

[Nock *et al.*, 2018] Richard Nock, Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Entity resolution and federated learning get a federated resolution. *arXiv preprint arXiv:1803.04035*, 2018.

[Pinkas *et al.*, 2014] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on {OT} extension. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 797–812, 2014.

[Veidinger, 1960] L Veidinger. On the numerical determination of the best approximations in the chebyshev sense. *Numerische Mathematik*, 2(1):99–105, 1960.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[Will *et al.*, 2017] Hamilton Will, Ying Zhitao, and Leskovec Jure. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1024–1034. Curran Associates and Inc., 2017.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[Yuan and Yu, 2013] Jiawei Yuan and Shucheng Yu. Privacy preserving back-propagation neural network learning made practical with cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):212–221, 2013.

[Zhang *et al.*, 2015] Qingchen Zhang, Laurence T Yang, and Zhikui Chen. Privacy preserving deep computation model on cloud for big data feature learning. *IEEE Transactions on Computers*, 65(5):1351–1362, 2015.