No Peek: A Survey of private distributed deep learning

Praneeth Vepakomma*, Tristan Swedish, Ramesh Raskar, Otkrist Gupta, and Abhimanyu Dubey

Massachusetts Institute of Technology
Cambridge, MA 02139, U.S.A

Abstract. We survey distributed deep learning models for training or inference without accessing raw data from clients. These methods aim to protect confidential patterns in data while still allowing servers to train models. The distributed deep learning methods of federated learning, split learning and large batch stochastic gradient descent are compared in addition to private and secure approaches of differential privacy, homomorphic encryption, oblivious transfer and garbled circuits in the context of neural networks. We study their benefits, limitations and trade-offs with regards to computational resources, data leakage and communication efficiency and also share our anticipated future trends.

1 Introduction

Emerging technologies in domains such as biomedicine, health and finance benefit from distributed deep learning methods which can allow multiple entities to train a deep neural network without requiring data sharing or resource aggregation at one single place. In particular, we are interested in distributed deep learning approaches that bridge the gap between distributed data sources (clients) and a powerful centralized computing resource (server) under the constraint that local data sources of clients are not allowed to be shared with the server or amongst other clients.

We survey and compare such distributed deep learning techniques and classify them across various dimensions of level and type of protection offered, model performance and resources required such as memory, time, communications bandwidth and synchronization requirements. We introduce the terminology of 'no peek' to refer to distributed deep learning techniques that do not share their data in raw form. We note that such no peek techniques allow the server to train models without 'peeking at', or directly observing, raw data belonging to clients.

Additionally, we survey some generic approaches to protecting data and models. Some of these approaches have already been used in combination with distributed deep learning methods that possess varying levels of the no peek property. These generic approaches include de-identification methods like anonymization [52], obfuscation methods like differential privacy [100,101,102] and cryptographic

---

* Corresponding author e-mail: vepakom@mit.edu

techniques like homomorphic encryption [19,93,96] and secure multi-party computation (MPC) protocols like oblivious transfer [84,47] and garbled circuits [41].

In the rest of the paper, we focus on distributed deep learning techniques such as splitNN [31,2], large batch synchronous stochastic gradient descent (SGD) [20,9], federated learning[3] and other variants [107,108,109,110] in the context of protecting data and models.

1.1   No peek rule

We refer to techniques of distributed deep learning that do not look at raw data once it leaves the client as satisfying the property of 'no peek'. No peek is necessitated by trust and regulatory issues. For example, hospitals are typically not allowed to share data with for-profit entities due to trust issues. They also are unable to share it with external entities (data cannot physically leave the premises) due to limited consent of the patients, and regulations such as HIPAA [5,1,6,7,8] that prevent sharing many aspects of the data to external entities. Some techniques go a step ahead by also not revealing details of the model architecture as well. In these techniques, neither the server nor client can access the details of the other's architecture or weights.

1.2   What needs to be protected

Protection mechanisms in the context of distributed deep learning should protect various aspects of datasets such as

1. Input features
2. Output labels or responses
3. Model details including the architecture, parameters and loss function
4. Identifiable information such as which party contributed to a specific record

1.3   Computational Goals

It is also quite important that any mechanism that aims to protect these details also preserves utility of the model above an acceptable level. These goals are to be ideally achieved at a low cost with regards to

1. Memory
2. Computational time
3. Communications bandwidth
4. Synchronization

As shown in Fig 1. below smaller hospitals or tele-healthcare screening centers do not acquire an enormous number of diagnostic images and they could also be limited by diagnostic manpower. A distributed machine learning method for diagnosis in this setting should ideally not share any raw data (no peek) and at same time achieve high accuracy while using significantly lower resources. This helps smaller hospitals to effectively serve those in need while benefiting from decentralized entities.

Distributed Method	Partial/Full Leakage	Differential Privacy	Homomorphic Encryption	Oblivious Transfer, Garbled Circuits
Distributed NN	[Dean2012, Wen2017, Das2016, Ooi2015], Ben2018]	[Hynes2018, Abadi2016, Shokri2015, Papernot2016]	[Juvekar2018, Gilad2016]	[Rouhani2017, Mohassel2017, Riazi2018, Orlandi2007]
Large Batch Synchronous SGD	[Konečný2015, Chen2016]			
Federated Learning	[McMahan2017, Nock2018]	[Geyer2017]	[Aono2018, Hardy2017]	[Bonawitz2016]
SplitNN	[Gupta2018, Vepakomma2018]			

Table 1: This is a survey of distributed deep learning methods with decreasing levels of leakage from distributed NN to splitNN. Hybrid approaches of these techniques and differential privacy, homomorphic encryption and MPC are also included. The citations for these 9 groups have been grouped separately with subtitles in the references section for convenience.
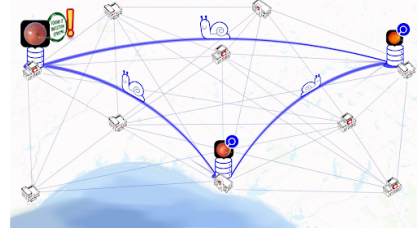

Fig. 1: Non-cooperating health units

Fig. 2: Distributed learning without raw data sharing

2 No peek approaches for distributed deep learning

In table 1 we provide corresponding references to various combinations of distributed deep learning techniques along with generic approaches for protection that are not specific to deep learning such as differential privacy homomorphic encryption and secure multi-party computation. The distributed deep learning techniques such as splitNN, federated learning and large batch synchronous SGD are 'no peek'. In addition splitNN also protects model details of the architecture and weights, unlike the other techniques. We detail this below in table 2 in terms of the levels of protection offered on data, intermediate representations and hyperparameters that include the deep learning architecture and learnt weights.

In federated learning and large batch synchronous SGD, the architecture and parameters are shared between the client and server along with intermediate representations of the model that include the gradients, activations and weight updates which are shared during the learning process. Although the data is

not explicitly shared in raw form in these two techniques, works like [106] have shown that raw data can be reconstructed up to an approximation by adversaries, especially given the fact that the architecture and parameters are not completely protected. SplitNN [31] has an added advantage in this context in that it does not

share the architecture and weights of the model. The protection offered by splitNN lies in the compact representations found in deeper layers of neural networks and the difficulty of recovering the underlying data from these representations without knowing the model weights used to produce them. Such representations form after passing the data through numerous activations whose inverse in the case of ReLU are nonlinear and ill-defined (the inverse of a zero-valued ReLU can map to any negative number). Such representations have been shown to preserve information important for certain tasks (path following [4]), without revealing information about the underlying data (such as image features in a 3D coordinate system). The intermediate representation shared by splitNN also requires minimal bandwidth in comparison to federated learning and large batch synchronous SGD, as only the activations from one layer of the client called the cut layer are shared with the server without any associated functions required to invert them back to raw data. In table 3, we compare these techniques based on

No Peek Deep Learning	Data revealed	Hyperparameters revealed	Intermediate representation revealed
Large Batch Synchronous SGD	No	Yes	Yes
Federated Learning	No	Yes	Yes
SplitNN	No	No	Yes

Table 2: In this table, we compare the level of privacy offered over data, model architecture, model parameters and intermediate representations by techniques like federated learning, large batch synchronous SGD and splitNN. On all these aspects, splitNN out performs federated learning or large batch synchronous SGD.

resources required such as computations, communication bandwidth, memory and synchronization. We categorize the techniques across these dimensions as having low, medium and high requirements. As shown, splitNN requires the lowest resources on the client side. This is because the architecture is cut (arbitrary shape and not necessarily vertical) at a layer where the computations are only performed up to that cut on the client side. The rest of the computations happen on the server side. The experimental results in [31], quantify these comparisons.

3   Federated Learning

Key idea: In this approach the clients download the current model from central server and improve it by updating their model weights based on their local data. The client model parameter updates are aggregated to generate server model. This model is again downloaded by the clients and the process continues. There is no explicit sharing of raw data in this setup.

Algorithm 1 `Naive Federated Learning`. Goal: To learn $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ from data stored across a large number of clients. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

EnsureServer executes at round $t \geq 0$:
  Distribute $\mathbf{W}_t$ to a subset S_t of n_t clients
  for each client $k \in S_t$ **in parallel do**
    $\mathbf{H}_t^k \leftarrow$ ClientUpdate(k, w_t)
  Set $\mathbf{H}_t := \frac{1}{n_t} \sum_{i \in S_t} \mathbf{H}_t^i$
  Set $\mathbf{W}_{t+1} = \mathbf{W}_t + \eta_t \mathbf{H}_t$

EnsureClientUpdate$(k, \mathbf{W}_t)$:   // *Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  Set $\mathbf{W}_t^k = \mathbf{W}_t$
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $\mathbf{W}_t^k \leftarrow \mathbf{W}_t^k - \eta \nabla \ell(\mathbf{W}_t^k; b)$
  return $\mathbf{H}_t^k = \mathbf{W}_t^k - \mathbf{W}_t$ to server

---

Algorithm 2 (Communication-Efficient Learning of Deep Networks from Decentralized Data):`Federated Averaging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

EnsureServer executes:
  initialize w_0
  for each round $t = 1, 2, \ldots$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow$ ClientUpdate(k, w_t)
    $w_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} w_{t+1}^k$

EnsureClientUpdate(k, w):   // *Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return w to server

3.1 Benefits

There is no explicit sharing of raw data. It has been shown in the convex case with IID data that in the worst-case, the global model produced is no better than training a model on a single client [9,3].

3.2 Limitations

Performance drops sharply when local data with clients is non i.i.d. That said, recent work in [112] on federated learning in this setting shows positive results. It

also requires large network bandwidth, memory and computation requirements on the client side depending on size of model, computation needs of complete forward and backward propagation. Advanced compression methods can be used instead to reduce this overload. There has been active recent work for neural network compression such as [35,36,111]. These works can thereby reduce the costs for communication bandwidth when used in distributed learning. Federated learning has no theoretical guarantees or trade-offs of privacy or security to date.

Client Resources Required	Compute	Bandwidth	Memory	Synchronization
Large Batch Synchronous SGD	High	High	High	Synchronous updates with backup workers to compensate slow machines.
Federated Learning	Medium	Medium	High	Synchronous client-server updates.
SplitNN	Low	Low	Low	Synchronous client-server updates.

Table 3: In this table, we compare the resources required for computation, bandwidth, memory and synchronization by techniques like federated learning, large batch synchronous SGD and SplitNN. SplitNN consumes fewer resources than federated learning and large batch synchronous SGD in these aspects except for synchronization requirements that are similar across all three techniques.

3.3   Future Trends

Data poisoning attacks on federated learning [37] where malicious users can inject false training data to negatively effect the classification performance of the model have been proposed. Adversarial robustness to such attacks need to be improved. Using neural-network compression schemes in conjunction with federated learning to reduce the communication overload is an avenue for future work. Looking at combinations of federated learning and differential privacy, secure multi-party computation is an interesting direction for future work given that there has been active research in the recent time in all these areas.

4   Large Batch Synchronous SGD

4.1   Key Idea

The technique introduces additional backup workers to work on updating the weights, and chooses to synchronously update the aggregated model, as soon as any of the fastest N workers finish their updates. This is an improvement in accuracy over asynchronous SGD where some of the local workers might be updating the weights of a more stale model as the client-server updates are asynchronous. It also is relatively faster than synchronous SGD with no back-up workers where the servers wait for all the clients to finish their updates before aggregating the model parameters to update the model.

4.2   Benefits

It allows for faster synchronous SGD, and is more accurate than asynchronous SGD approaches where some clients end up updating the weights based on a more stale model.

4.3   Limitations

The computational requirements and communication bandwidth required is much higher than other distributed deep learning methods.

4.4   Future Trends

The future trends are similar to that of federated learning as this method is very similar in essence to federated learning although it instead runs on a single batch of data. This method has high computational overload and network footprint. To make this method more sustainable in data center or decentralized settings, future work in improving its efficiency is important.

Algorithm 3 Large-Batch SGD

Ensure: **Worker Update**(k)**, where** $k = 1, \ldots, N + b$
  Input: Dataset $\mathcal{X}$, B mini-batch size.
  for $t = 0, 1, \ldots$ **do**
    Wait to read $\theta^{(t)} = (\theta^{(t)}[0], \ldots, \theta^{(t)}[M])$ from parameter servers.
    $G_k^{(t)} := 0$
    for $i = 1, \ldots, B$ **do**
      Sample datapoint $\widetilde{x}_{k,i}$ from $\mathcal{X}$.
      $G_k^{(t)} \leftarrow G_k^{(t)} + \frac{1}{B}\nabla F(\widetilde{x}_{k,i}; \theta^{(t)})$.
    Send $(G_k^{(t)}, t)$ to parameter servers.
Ensure: **Parameter Server Update**(j)**, where** $k = 1, \ldots, N + b$
  Input $\gamma_0, \gamma_1, \ldots$ learning rates, α decay rate, N number of mini-batches to aggregate, $\theta^{(0)}$ model initialization.
  for $t = 0, 1, \ldots$ **do**
    $\mathcal{G} = \{\}$
    while $|\mathcal{G}| < N$ **do**
      Wait for (G, t') from any worker.
      if $t' == t$ **then**
        $\mathcal{G} \leftarrow \mathcal{G} \cup \{G\}$.
      else
        Drop gradient G.
    $\theta^{(t+1)}[j] \leftarrow \theta^{(t)}[j] - \frac{\gamma_t}{N}\sum_{G \in \mathcal{G}} G[j]$.
    $\bar{\theta}^{(t)}[j] = \alpha\bar{\theta}^{(t-1)}[j] + (1 - \alpha)\theta^{(t)}[j]$.

---

5   Split Learning (SplitNN)

5.1   Key Idea

In this method each client trains the network upto a certain layer known as the cut layer and sends the weights to server. The server then trains the network for rest of the layers. This completes the forward propagation. Server then generates the gradients for the final layer and back-propagates the error until the cut layer. The gradient is then passed over to the client. The rest of the back-propagation is completed by client. This is continued till the network is trained. The shape of the cut could be arbitrary and not necessarily, vertical. In this framework as well there is no explicit sharing of raw data.

5.2   Benefits

Client-side communication costs are significantly reduced as the data to be transmitted is restricted to first few layers of the splitNN prior to the split. The client-side computation costs of learning the weights of the network are also significantly reduced for the same reason. In terms of model performance, the accuracies of Split NN remained much higher than federated learning and large batch synchronous SGD with a drastically smaller client side computational burden when training on a larger number of clients.

5.3   Limitations

It requires a relatively larger overall communication bandwidth when training over a smaller number of clients although it ends up being much lower than other methods in settings with large number of clients. Advanced neural network compression methods such as [35,36,111] can be used to reduce the communication load. The communication bandwidth can also be traded for computation on client by allowing for more layers at client to represent further compressed representations.

5.4   Future Trends

Given its no peek properties, no model detail sharing and high resource efficiency of this recently proposed method, it is well placed to provide direct applications to important domains like distributed healthcare, distributed clinical trials, inter and intra organizational collaboration and finance. Using neural-network compression schemes in conjunction with splitNN to reduce communication overload is a promising avenue for future work. Looking at combinations of federated learning and differential privacy, secure multi-party computation is an interesting direction for future work as well given that there has been active research in recent time in all these areas.

Algorithm 4 `SplitNN`. The K clients are indexed by k; B is the local minibatch size, and η is the learning rate.

EnsureServer executes at round $t \geq 0$**:**
  for each client $k \in S_t$ **in parallel do**
    $\mathbf{A}_t^k \leftarrow \text{ClientUpdate}(k, t)$
    Compute $\mathbf{W}_t \leftarrow \mathbf{W}_t - \eta \nabla \ell(\mathbf{W}_t; \mathbf{A}_t)$
    Send $\nabla \ell(\mathbf{A}_t; \mathbf{W}_t)$ to client k for $\text{ClientBackprop}(k, t)$

EnsureClientUpdate(k, t)**:** // *Run on client* k
  $\mathbf{A}_t^k = \phi$
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      Concatenate $f(b, \mathbf{H}_t^k)$ to $\mathbf{A}_t^k$
  return $\mathbf{A}_t^k$ to server

EnsureClientBackprop$(k, t, \nabla \ell(\mathbf{A}_t; \mathbf{W}_t))$**:** // *Run on client* k
  for batch $b \in \mathcal{B}$ **do**
    $\mathbf{H}_t^k = \mathbf{H}_t^k - \eta \nabla \ell(\mathbf{A}_t; \mathbf{W}_t; b)$

---

6 Methods to Further Reduce Leakage and Improve Efficiency

6.1 Obfuscation with Differential Privacy for NN

Key Idea: The methods in [14] modify stochastic gradient descent (SGD) based optimization used in learning neural networks by clipping the gradient for each lot of data and adding Gaussian noise to it during the optimization as opposed to adding noise to final parameters of the model, which could be overly conservative thereby affecting the utility of the trained model. The sigma for the noise is chosen at each step so as to maintain a guaranteed epsilon-delta differential privacy for a given lot of data. The tradeoff between the conflicting objectives of accuracy and privacy is determined by the lot size.

Benefits and Limitations: The privacy is always dependent on a limited privacy budget while this also has an inversely proportional dependency with model accuracy. This is unlike in SplitNN where high accuracies are achieved without sharing raw data. The guarantees of differential privacy are currently theoretically backed unlike in SplitNN or Federated Learning. It also violates the no-peak rule when the privacy budget is over.

Future Trends: There is a lot of scope in combining differential privacy with distributed deep learning methods like splitNN, federated learning and large batch SGD as it adds to stronger theoretical guarantees on preventing data leakage.

6.2   Homomorphic Encryption for NN

Key Idea Homomorphic encryption aims to preserve the structure of ciphers such that addition and multiplicative operations can be performed after the encryption. All operations in a neural network except for activation functions are sum and product operations which can be encoded using Homomorphic encryption. Activation functions are approximated with either higher degree polynomials, Taylor series, standard or modified Chebyshev polynomials that are then implemented as part of Homomorphic encryption schemes. The works in [17,19] apply these ideas in the context of deep learning. A greatly detailed survey comparing various software libraries for homomorphic encryption is provided in [115].

Algorithm 5 Differentially private SGD

Require: Examples $\{x_1, ..., x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.
  Initialize θ_0 randomly
  for $t \in [T]$ **do**
    Take a random sample L_t with sampling probability L/N
    Compute gradient
    For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$
    Clip gradient
    $\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/\max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$
    Add noise
    $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$
    Descent
    $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$
  Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Algorithm There are a variety of schemes which have been shown to have homomorphic properties, and are provably secure. The most common use the security of the LWE (learning with errors) problem, which seeks to solve a linear system after adding noise. This problem is difficult to solve in certain conditions (when the dimension of the vector space is much larger than the computational range), and has even been shown to be secure under known quantum attacks. In short, LWE contains an algebraic structure with homomorphisms for addition and multiplication under the finite field of integers (so all multiplication/addition of finite integers can be encrypted and evaluated homomorphically as a LWE problem). In practice, implementations use R-LWE (Ring-LWE, use polynomial rings instead of vector spaces explicitly), which uses a slightly different representation, but the underlying algebraic structure remains largely the same.
Simple LWE example: The key generation, encryption/decryption and corresponding add/multiply operations for a simple LWE example are given below.

Keygen:

$$A \in \mathbb{Z}_q^{m \times n}$$
$$S \sim \mathbb{Z}_q^n$$
$$e \sim \mathcal{N}^n$$
$$b = As + e \tag{1}$$

Encrypt/Encode:

$$r_1, e_1 \sim \mathcal{N}$$
$$c = (c_a, c_b) = \left(A^T r_1, b^T r_1 + m_1 + e_1 \right) \tag{2}$$

Add/Multiply:

$$c_{\text{add}} = c_1 + c_2 \tag{3}$$
$$c_{\text{mult}} = D \left(c_1 \otimes c_2 \right) \tag{4}$$

where $\otimes$ is the tensor product and D is a dimension switching matrix that simplifies the resulting ciphertext. A proof of correctness and further sophistication addressing this scheme as a practical system can be found in the LWE literature.

Benefits and Limitations These techniques need specialized hardware or extensive computational resources to scale. They are capable of providing a higher level of security that allows for perfect decryption and are not dependent on trade-offs of obfuscation vs. accuracy. The tradeoffs involved in this case are more with regards to computational efficiency. For example, some work (Microsoft's SEAL) shows that to perform logistic regression on 1MB of data, 10GB of memory are required, and massive parallelization is necessary to achieve real-time throughput on practical problems (some tasks may not be parallelizable as such). LWE hardness is believed to be valid even in a post-quantum cryptographic environment.

Future Trends This method requires very high computational resources, to make it scalable to practical deep learning architectures. Current techniques have only been benchmarked on simple networks over small datasets such as MNIST hand-written digit recognition. Development of faster methods for large scale deep learning and specialized hardware is an important avenue for future work.

6.3 Multi-Party Computation (MPC) and Garbled Circuits

Key Idea: These techniques are based on the idea of secret sharing and zero knowledge proof that we describe. The protection is achieved by sharing a

secret message with different entities and requiring that these entities cooperate together in order to gain accessibility. There are certain problems where two entities collaborate to compute a function without sharing information about the inputs to the function with each other. The classic example is the millionaire's problem, where $f(x_1, x_2)$ is computed by two parties, when one party has x_1 and the other has x_2, and it's impossible for the party to the learn the value the other party holds. $f(x_1, x_2)$ will return a positive number if $x_1 > x_2$ and a negative value if $x_2 > x_1$. In this way, two millionaire's can determine who has more money without sharing the total value at each hold. This has practical applications to untrusted "credit checks" or as an example for a particular kind of "Zero Knowledge Proof." Yao's [48] garbled circuit protocol for the millionaire's problem and 1–2 oblivious transfer [114,113] are prominent works in this direction. Computational implementations and frameworks for this work such as Obliv-C, ObliVM, SPDZ and Sharemind are prominent.

Benefits and Limitations: These techniques have been studied for problems like secure stable matching, linear system solving and parallel graph algorithms. There is not much work done at intersection of MPC with deep learning.

Future Trends: Specialized hardwares for MPC are being developed to realize practical applications of these protocols. As current day machine learning relies heavily on large scale deep-learning architectures on large datasets, bridging this gap between MPC frameworks and distributed deep learning is an important avenue for future work.

Method	100 Clients	500 Clients
Large Batch SGD	29.4 TFlops	5.89 TFlops
Federated Learning	29.4 TFlops	5.89 TFlops
SplitNN	0.1548 TFlops	0.03 TFlops

Table 4: Computation resources consumed per client when training CIFAR 10 over VGG (in teraflops) are drastically lower for SplitNN than Large Batch SGD and Federated Learning.

7 Comparison of resource efficiency across no peek distributed deep learning

We now share a comparison from [31] of validation accuracy and required client computational resources in Figure 1 for the three techniques of federated learning, large batch synchronous SGD and splitNN as they are tailored for distributed deep learning. As seen in this figure, the comparisons were done on datasets of CIFAR 10 and CIFAR 100 using VGG and Resnet-50 architectures for 100 and 500 client based setups respectively. In this distributed learning experiment we clearly see that SplitNN outperforms the techniques of federated learning and large batch synchronous SGD in terms of higher accuracies with drastically lower

Method	100 Clients	500 Clients
Large Batch SGD	13 GB	14 GB
Federated Learning	3 GB	2.4 GB
SplitNN	6 GB	1.2 GB

Table 5: Computation bandwidth required per client when training CIFAR 100 over ResNet (in gigabytes) is lower for splitNN than large batch SGD and federated learning with a large number of clients. For setups with a smaller number of clients, federated learning requires a lower bandwidth than splitNN. Large batch SGD methods popular in data centers use a heavy bandwidth in both settings.

computational requirements on the side of clients. In tables 4 and 5 we share more comparisons from [31] on computing resources in TFlops and communication bandwidth in GB required by these techniques. SplitNN again has a drastic improvement of computational resource efficiency on the client side. In the case with a relatively smaller number of clients the communication bandwidth required by federated learning is less than splitNN.
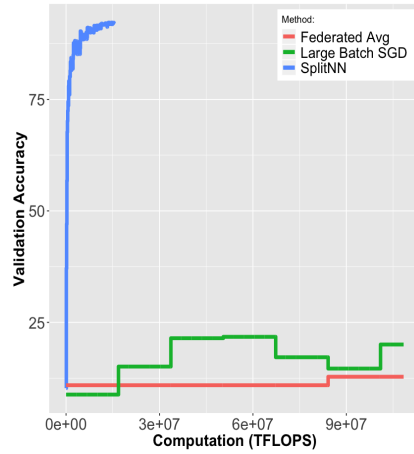
(a) Accuracy vs client-side flops on 100 clients with VGG on CIFAR 10
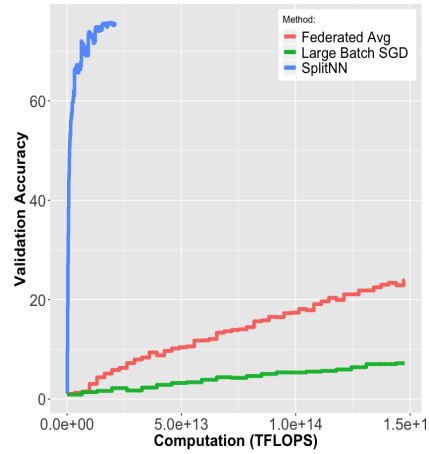
(b) Accuracy vs client-side flops on 500 clients with Resnet-50 on CIFAR 100

Fig. 3: We show dramatic reduction in computational burden (in tflops) while maintaining higher accuracies when training over large number of clients with splitNN. Blue line denotes distributed deep learning using splitNN, red line indicate federated averaging and green line indicates large batch SGD.

8   Conclusion and Future Work

No peek deep neural networks require new thinking when compared to existing data protection methods that attempt to aggregate siloed data for the benefit of server models. We describe the emergence of three methods in this setting: splitNN, federated learning and large batch SGD. Novel combinations of these methods with differential privacy, homomorphic encryption and secure MPC could further exploit theoretical guarantees. We show that in settings with large number of clients, splitNN needs the least communications bandwidth while federated learning does better with relatively smaller number of clients. In this direction, improving resource and communication efficiencies of no peek methods would be another avenue for impactful future work. Using advanced neural network compression methods [35,111,36] will help further reduce the required network footprint. It is also important to study adversarial robustness to data poisoning attacks [37] where malicious users can inject false training data to negatively effect the classification performance of the model. Adversarial attack schemes from this parallel research area need to be taken into consideration while developing no peek mechanisms. Efficient no peek methods have direct applications to important domains like distributed healthcare, distributed clinical trials, inter and intra organizational collaboration and finance. We therefore contemplate novel no peek distributed deep learning applications in the future.

9   References

References

1. Centers for Disease Control and Prevention,HIPAA privacy rule and public health. Guidance from CDC and the US Department of Health and Human Services, MMWR: Morbidity and mortality weekly report, US Centers for Disease Control and Prevention, 2003
2. keywords = block10, Vepakomma, Praneeth and Gupta, Otkrist and Swedish, Tristan and Raskar, Ramesh, Split learning for health: Distributed deep learning without sharing raw patient data, arXiv1812.00564,2018
3. H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson and Blaise Aguera y Arcas, Communication-efficient learning of deep networks from decentralized data, 20'th International Conference on Artificial Intelligence and Statistics (AISTATS), 2017.
4. Swedish, Tristan and Raskar, Ramesh, Deep Visual Teach and Repeat on Path Networks, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2018.
5. Annas, George J. , HIPAA regulations-a new era of medical-record privacy?, New England Journal of Medicine, Vol.348 (15), pp.1486–1490, 2003.
6. Mercuri, Rebecca T. , The HIPAA-potamus in health care data security, Communications of the ACM, Vol.47 (7), pp.25–28, 2004.
7. Gostin, Lawrence O., Levit, Laura A. and Nass, Sharyl J. , Beyond the HIPAA privacy rule: enhancing privacy, improving health through research, National Academies Press, 2009.

8. Luxton, David D and Kayl, Robert A and Mishkind, Matthew C. , mHealth data security: The need for HIPAA-compliant standardization, Telemedicine and e-Health, Vol.18(4), pp. 284–288, 2012.
9. Konečny, Jakub and McMahan, H Brendan and Yu, Felix X and Richtárik, Peter and Suresh, Ananda Theertha and Bacon, D. , Federated learning: Strategies for improving communication efficiency, arXiv preprint arXiv:1610.05492, 2016.
10. Hynes, Nick and Cheng, Raymond and Song, Dawn , Efficient Deep Learning on Multi-Source Private Data, arXiv preprint arXiv:1807.06689, 2018.
11. Abadi, Martin and Chu, Andy and Goodfellow, Ian and McMahan, H Brendan and Mironov, Ilya and Talwar, Kunal and Zhang, Li, Deep learning with differential privacy, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp.308–318, 2016.
12. Shokri, Reza and Shmatikov, Vitaly, Privacy-preserving deep learning, Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp.1310–1321 2015.
13. Papernot, Nicolas and Abadi, Martín and Erlingsson, Ulfar and Goodfellow, Ian and Talwar, Kunal, Semi-supervised knowledge transfer for deep learning from private training data, arXiv preprint arXiv:1610.05755, 2016.
14. Geyer, Robin C and Klein, Tassilo and Nabi, Moin, Differentially private federated learning: A client level perspective, arXiv preprint arXiv:1712.07557, 2017.
15. Rouhani, Bita Darvish and Riazi, M Sadegh and Koushanfar, Farinaz, Deepsecure: Scalable provably-secure deep learning, arXiv preprint arXiv:1705.08963, 2017.
16. Rouhani, Bita Darvish and Riazi, M Sadegh and Koushanfar, Farinaz, SecureML: A system for scalable privacy-preserving machine learning, 38th IEEE Symposium on Security and Privacy (SP), 2017.
17. Hesamifard, Ehsan and Takabi, Hassan and Ghasemi, Mehdi, CryptoDL: Deep Neural Networks over Encrypted Data, arXiv preprint arXiv:1711.05189, 2017.
18. Hardy, Stephen and Henecka, Wilko and Ivey-Law, Hamish and Nock, Richard and Patrini, Giorgio and Smith, Guillaume and Thorne, Brian, Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption, arXiv preprint arXiv:1711.10677, 2017.
19. Aono, Yoshinori and Hayashi, Takuya and Wang, Lihua and Moriai, Shiho, Privacy-preserving deep learning via additively homomorphic encryptionn, IEEE Transactions on Information Forensics and Security, Vol.13(5), pp.1333–1345arXiv preprint arXiv:1711.10677, 2018.
20. Chen, Jianmin and Pan, Xinghao and Monga, Rajat and Bengio, Samy and Jozefowicz, Rafal, Revisiting distributed synchronous SGD, IEEE Transactions on Information Forensics and Security, Vol.13(5), arXiv preprint arXiv:1604.00981, 2016.
21. Bonawitz, Keith and Ivanov, Vladimir and Kreuter, Ben and Marcedone, Antonio and McMahan, H Brendan and Patel, Sarvar and Ramage, Daniel and Segal, Aaron and Seth, Karn, Practical secure aggregation for privacy-preserving machine learning, Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp.1175–1191, 2017.
22. Ben-Nun, Tal and Hoefler, Torsten, Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis, arXiv preprint arXiv:1802.09941, 2018.
23. Shickel, Benjamin and Tighe, Patrick James and Bihorac, Azra and Rashidi, Parisa, Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis, IEEE journal of biomedical and health informatics, Vol.22(5) pp.1589–1604, 2018.

24. Ching, Travers and Himmelstein, Daniel S and Beaulieu-Jones, Brett K and Kalinin, Alexandr A and Do, Brian T and Way, Gregory P and Ferrero, Enrico and Agapow, Paul-Michael and Zietz, Michael and Hoffman, Michael M , Opportunities and obstacles for deep learning in biology and medicine, Journal of The Royal Society Interface, Vol.15(141), 2018.
25. Miotto, Riccardo and Wang, Fei and Wang, Shuang and Jiang, Xiaoqian and Dudley, Joel T., Deep learning for healthcare: review, opportunities and challenges, Briefings in bioinformatics, 2017.
26. Smith, Virginia and Chiang, Chao-Kai and Sanjabi, Maziar and Talwalkar, Ameet S., Advances in Neural Information Processing Systems, pp.4424–4434, 2017.
27. Syverson, Paul and Dingledine, R and Mathewson, N, Tor: The second generation onion router,Usenix Security, 2004.
28. Ravı, Daniele and Wong, Charence and Deligianni, Fani and Berthelot, Melissa and Andreu-Perez, Javier and Lo, Benny and Yang, Guang-Zhong, Deep learning for health informatics, IEEE journal of biomedical and health informatics, Vol.21(1), pp.4–21, 2017.
29. Alipanahi, Babak and Delong, Andrew and Weirauch, Matthew T and Frey, Brendan J, Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning, Nature biotechnology, Vol.33(8), 2015.
30. Litjens, Geert and Kooi, Thijs and Bejnordi, Babak Ehteshami and Setio, Arnaud Arindra Adiyoso and Ciompi, Francesco and Ghafoorian, Mohsen and van der Laak, Jeroen AWM and Van Ginneken, Bram and Sánchez, Clara I, A survey on deep learning in medical image analysis, Medical image analysis, Vol.42, pp.60–88, 2017.
31. Gupta, Otkrist and Raskar, Ramesh, Distributed learning of deep neural network over multiple agents, Journal of Network and Computer Applications, Vol.116, pp.1–8, 2018.
32. Navathe, Shamkant and Ceri, Stefano and Wiederhold, Gio and Dou, Jinglie, Vertical partitioning algorithms for database design, ACM Transactions on Database Systems (TODS), Vol.9(4), pp.680–710, 1984.
33. Agrawal, Sanjay and Narasayya, Vivek and Yang, Beverly, Integrating vertical and horizontal partitioning into automated physical database design, Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp.359–370, 2004.
34. Abadi, Daniel J and Marcus, Adam and Madden, Samuel R and Hollenbach, Kate, Scalable semantic web data management using vertical partitioning, Proceedings of the 33rd international conference on Very large data bases, pp.411–422, 2007.
35. Lin, Yujun and Han, Song and Mao, Huizi and Wang, Yu and Dally, William J, Deep gradient compression: Reducing the communication bandwidth for distributed training, arXiv preprint arXiv:1712.01887, 2017.
36. Han, Song and Mao, Huizi and Dally, William J, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, arXiv preprint arXiv:1510.00149, 2015.
37. Fung, Clement and Yoon, Chris JM and Beschastnikh, Ivan, Mitigating Sybils in Federated Learning Poisoning, arXiv preprint arXiv:1808.04866, 2018.
38. Xie, Liyang and Lin, Kaixiang and Wang, Shu and Wang, Fei and Zhou, Jiayu, Differentially Private Generative Adversarial Network, arXiv preprint arXiv:1802.06739, 2018.
39. Crawford, Jack LH and Gentry, Craig and Halevi, Shai and Platt, Daniel and Shoup, Victor, Doing Real Work with FHE: The Case of Logistic Regression, 2018

40. Sans, Edouard Dufour and Gay, Romain and Pointcheval, David, Reading in the Dark: Classifying Encrypted Digits with Functional Encryption, 2018
41. Rosulek, Mike, Improvements for Gate-Hiding Garbled Circuits, International Conference in Cryptology in India, 2017
42. Choudhury, Ashish and Loftus, Jake and Orsini, Emmanuela and Patra, Arpita and Smart, Nigel P, Between a Rock and a Hard Place: Interpolating between MPC and FHE, International Conference on the Theory and Application of Cryptology and Information Security, 2013
43. Keller, Marcel and Pastro, Valerio and Rotaru, Dragos, Overdrive: making SPDZ great again, Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2018
44. Juvekar, Chiraag and Vaikuntanathan, Vinod and Chandrakasan, Anantha, Gazelle: A low latency framework for secure neural network inference, arXiv preprint arXiv:1801.05507, 2018
45. Gilad-Bachrach, Ran and Dowlin, Nathan and Laine, Kim and Lauter, Kristin and Naehrig, Michael and Wernsing, John, Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy, International Conference on Machine Learning, 2016
46. Riazi, M Sadegh and Weinert, Christian and Tkachenko, Oleksandr and Songhori, Ebrahim M and Schneider, Thomas and Koushanfar, Farinaz, Chameleon: A hybrid secure computation framework for machine learning applications, Proceedings of the 2018 on Asia Conference on Computer and Communications Security, 2018
47. Liu, Jian and Juuti, Mika and Lu, Yao and Asokan, N, Oblivious neural network predictions via minionn transformations, Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017
48. , Andrew C. Yao, Protocols for Secure Computations, University of California Berkeley, California, IEEE Foundations of Computer Science, 23rd Annual Symposium on, https://research.cs.wisc.edu/areas/sec/yao1982-ocr.pdf, 1982
49. Ziad, M Tarek Ibn and Alanwar, Amr and Alzantot, Moustafa and Srivastava, Mani, Cryptoimg: Privacy preserving processing over encrypted images, IEEE Conference on Communications and Network Security, 2016
50. Mironov, Ilya, Renyi differential privacy, IEEE 30th Computer Security Foundations Symposium, 2017
51. Kuo, Yu-Hsuan and Chiu, Cho-Chun and Kifer, Daniel and Hay, Michael and Machanavajjhala, Ashwin, Differentially private hierarchical count-of-counts histograms, Proceedings of the VLDB Endowment, 2018
52. Li, Ninghui and Li, Tiancheng and Venkatasubramanian, Suresh, t-closeness: Privacy beyond k-anonymity and l-diversity, IEEE 23rd International Conference on Data Engineering (ICDE), 2007
53. He, Xi and Machanavajjhala, Ashwin and Ding, Bolin, Blowfish privacy: Tuning privacy-utility trade-offs using policies, Proceedings of the 2014 ACM SIGMOD international conference on Management of data, 2014
54. He, Xi and Cormode, Graham and Machanavajjhala, Ashwin and Procopiuc, Cecilia M and Srivastava, Divesh, DPT: differentially private trajectory synthesis using hierarchical reference systems, Proceedings of the VLDB Endowment, 2015
55. Carlini, Nicholas and Liu, Chang and Kos, Jernej and Erlingsson, Úlfar and Song, Dawn, The Secret Sharer: Measuring Unintended Neural Network Memorization & Extracting Secrets,arXiv preprint arXiv:1802.08232, 2018
56. Hisham Husain, Zac Cranko, Richard Nock, Integral Privacy for Sampling from Mollifier Densities with Approximation Guarantees, arXiv:1806.04819, 2018

57. Zhang, Zuhe and Rubinstein, Benjamin IP and Dimitrakakis, Christos, On the Differential Privacy of Bayesian Inference, AAAI conference on artificial intelligence, AAAI, 2016
58. Jain, Prateek and Kothari, Pravesh and Thakurta, Abhradeep, Differentially private online learningConference on Learning Theory,2012
59. Xi Wu and Fengan Li and Arun Kumar and Kamalika Chaudhuri and Somesh Jha and Jeffrey F. Naughton, Bolt-on Differential Privacy for Scalable Stochastic Gradient Descent-based Analytics, SIGMOD Conference, 2017
60. Bagdasaryan, Eugene and Veit, Andreas and Hua, Yiqing and Estrin, Deborah and Shmatikov, Vitaly, How To Backdoor Federated Learning, arXiv preprint arXiv:1807.00459, 2018
61. Jalaj Upadhyay, Differentially Private Linear Algebra in the Streaming Model, IACR Cryptology ePrint Archive, 2014
62. Nikita Mishra, Private Stochastic Multi-arm Bandits: From Theory to Practice, 31 st International Conference on Machine Learning (ICML), 2014
63. Ruochi Zhang and Parv Venkitasubramaniam, Mutual-Information-Private Online Gradient Descent Algorithm, 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018
64. Seth Gilbert and Xiao Liu and Haifeng Yu, On Differentially Private Online Collaborative Recommendation Systems, International Conference on Information Security and Cryptology, 2015
65. John N. Tsitsiklis and Kuang Xu and Zhi Xu, Private Sequential Learning, COLT, 2018
66. Depeng Xu and Shuhan Yuan and Xintao Wu, Differential Privacy Preserving Causal Graph Discovery, IEEE Symposium on Privacy-Aware Computing (PAC), 2017
67. Jacob D. Abernethy and Chansoo Lee and Audra McMillan and Ambuj Tewari, Online Learning via Differential Privacy, CoRR, abs/1711.10019, 2017
68. Ngoc-Son Phan and Xintao Wu and Dejing Dou, Preserving differential privacy in convolutional deep belief networks, Machine Learning Journal, Springer, 2017
69. Giulia C. Fanti and Vasyl Pihur and Úlfar Erlingsson, Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries, 2016
70. Ziteng Wang and Chi Jin and Kai Fan and Jiaqi Zhang and Junliang Huang and Yiqiao Zhong and Liwei Wang, Differentially Private Data Releasing for Smooth Queries, Journal of Machine Learning Research, 2016
71. Raghavendran Balu and Teddy Furon, Differentially Private Matrix Factorization using Sketching Techniques, Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security, 2016
72. Mohsen Ghassemi and Anand D. Sarwate and Rebecca N. Wright, Differentially Private Online Active Learning with Applications to Anomaly Detection, AISec-CCS, 2016
73. Yao, Zhewei and Gholami, Amir and Lei, Qi and Keutzer, Kurt and Mahoney, Michael W, Hessian-based Analysis of Large Batch Training and Robustness to Adversaries, arXiv preprint arXiv:1802.08241, 2018
74. Yuncheng Wu and Yao Wu and Hui Peng and Juru Zeng and Hong Chen and Cuiping Li, Differentially private density estimation via Gaussian mixtures model, IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), 2016
75. Shiva Prasad Kasiviswanathan and Hongxia Jin, Efficient Private Empirical Risk Minimization for High-dimensional Learning, Interncational Conference on Machine Learning, ICML, 2016

76. Lu Tian and Bargav Jayaraman and Q. Gu and David Evans, Aggregating Private Sparse Learning Models Using Multi-Party Computation, 2016
77. John C. Duchi and Michael I. Jordan and Martin J. Wainwright, Privacy Aware Learning, Neural Information Processing Systems (NIPS), 2012
78. Marco Gaboardi and Emilio Jesús Gallego Arias and Justin Hsu and Aaron Roth and Zhiwei Steven Wu, Dual Query: Practical Private Query Release for High Dimensional Data, Interncational Conference on Machine Learning, ICML, 2014
79. Nikolaenko, Valeria and Weinsberg, Udi and Ioannidis, Stratis and Joye, Marc and Boneh, Dan and Taft, Nina, Privacy-preserving ridge regression on hundreds of millions of records, IEEE Symposium on Security and Privacy (SP), 2013
80. Miran Kim and Kristin E. Lauter, Private Genome Analysis through Homomorphic Encryption, BMC medical informatics and decision making, 2015
81. Thore Graepel and Kristin E. Lauter and Michael Naehrig, ML Confidential: Machine Learning on Encrypted Data, IACR Cryptology ePrint Archive, 2012
82. Raphael Bost and Raluca A. Popa and Stephen Tu and Shafi Goldwasser, Machine Learning Classification over Encrypted Data, IACR Cryptology ePrint Archive, 2014
83. Oded Goldreich and Shafi Goldwasser and Dana Ron, Property Testing and its connection to Learning and Approximation, Journal of the ACM, 1996
84. Orlandi, Claudio and Piva, Alessandro and Barni, Mauro, Oblivious neural network computing via homomorphic encryption, EURASIP Journal on Information Security, Springer, 2007
85. Shengshan Hu and Qian Wang and Jingjun Wang and Sherman S. M. Chow and Qin Zou, Securing Fast Learning! Ridge Regression over Encrypted Big Data, IEEE Trustcom/BigDataSE/ISPA, 2016
86. Mauro Barni and Pierluigi Failla and Riccardo Lazzeretti and Ahmad-Reza Sadeghi and Thomas Schneider, Privacy-Preserving ECG Classification With Branching Programs and Neural Networks, IEEE Transactions on Information Forensics and Security, 2011
87. Zhan Qin and Jingbo Yan and Kui Ren and Chang Wen Chen and Xinyu Wang, SecSIFT: Secure Image SIFT Feature Extraction in Cloud Computing, TOMCCAP, 2016
88. Yifeng Zheng and Helei Cui and Xinyu Wang and Jiantao Zhou, Privacy-Preserving Image Denoising From External Cloud Databases, IEEE Transactions on Information Forensics and Security, 2017
89. Reda Bellafqira and Gouenou Coatrieux and Dalel Bouslimi and Gwénolé Quellec and Michel Cozic, Secured Outsourced Content Based Image Retrieval Based on Encrypted Signatures Extracted From Homomorphically Encrypted Images, CoRR abs/1704.00457,2017
90. Reda Bellafqira and Gouenou Coatrieux and Emmanuelle Génin and Michel Cozic Secure Multilayer Perceptron Based On Homomorphic Encryption, CoRR abs/1806.02709, 2018
91. Ryo Yonetani and Vishnu Naresh Boddeti and Kris M. Kitani and Yoichi Sato, Privacy-Preserving Visual Learning Using Doubly Permuted Homomorphic Encryption, 2017 IEEE International Conference on Computer Vision (ICCV), 2017
92. Tribhuvanesh Orekondy and Seong Joon Oh and Bernt Schiele and Mario Fritz, Understanding and Controlling User Linkability in Decentralized Learning, CoRR abs/1805.05838, 2018
93. David Wu, Using Homomorphic Encryption for Large Scale Statistical Analysis, Stanford report, 2012

94. Pedro M. Esperança and Louis J. M. Aslett and Chris C. Holmes, Encrypted accelerated least squares regression, Artificial intelligence and statistics, AISTATS, 2017

95. Richard Nock and Stephen Hardy and Wilko Henecka and Hamish Ivey-Law and Giorgio Patrini and Guillaume Smith and Brian Thorne, Entity Resolution and Federated Learning get a Federated Resolution, CoRR, abs/1803.04035, 2018

96. Wen-Jie Lu and Jun Sakuma, Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data, 2016

97. Yuchen Zhang and Wenrui Dai and Xiaoqian Jiang and Hongkai Xiong and Shuang Wang, FORESEE: Fully Outsourced secuRe gEnome Study basEd on homomorphic Encryption, BMC medical informatics and decision making, 2015

98. Md. Nazmus Sadat and Md Momin Al Aziz and Noman Mohammed and Feng Chen and Shuang Wang and Xiaoqian Jiang, SAFETY: Secure gwAs in Federated Environment Through a hYbrid solution with Intel SGX and Homomorphic Encryption, IEEE/ACM transactions on computational biology and bioinformatics, 2018

99. Md Momin Al Aziz and Mohammad Zahidul Hasan and Noman Mohammed and Dima Alhadidi, Secure and Efficient Multiparty Computation on Genomic Data, 2016

100. Dwork, Cynthia and Roth, Aaron, The algorithmic foundations of differential privacy, Foundations and Trends® in Theoretical Computer Science, Now Publishers Inc, 2014

101. Cynthia Dwork and Frank McSherry and Kobbi Nissim and Adam D. Smith, Calibrating Noise to Sensitivity in Private Data Analysis, TCC, 2006

102. Dwork, Cynthia, A Firm Foundation for Private Data Analysis, Commun. ACM, 10.1145/1866739.1866758, ACM

103. Bourse, Florian and Minelli, Michele and Minihold, Matthias and Paillier, Pascal Fast homomorphic evaluation of deep discretized neural networks, Annual International Cryptology Conference, Springer, 2018

104. Upadhyay, Jalaj, The Price of Differential Privacy for Low-Rank Factorization, Neural Information Processing Systems, 2018

105. Awan, Jordan and Slavkovic, Aleksandra, Differentially Private Uniformly Most Powerful Tests for Binomial Data, Neural Information Processing Systems, 2018

106. Hitaj, Briland and Ateniese, Giuseppe and Perez-Cruz, Fernando, Deep models under the GAN: information leakage from collaborative deep learning, Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017

107. Dean, Jeffrey and Corrado, Greg and Monga, Rajat and Chen, Kai and Devin, Matthieu and Mao, Mark and Senior, Andrew and Tucker, Paul and Yang, Ke and Le, Quoc V, Large scale distributed deep networks, Advances in neural information processing systems, 2012

108. Wen, Wei and Xu, Cong and Yan, Feng and Wu, Chunpeng and Wang, Yandan and Chen, Yiran and Li, Hai, Terngrad: Ternary gradients to reduce communication in distributed deep learning, Advances in neural information processing systems, 2017

109. Das, Dipankar and Avancha, Sasikanth and Mudigere, Dheevatsa and Vaidynathan, Karthikeyan and Sridharan, Srinivas and Kalamkar, Dhiraj and Kaul, Bharat and Dubey, Pradeep, Distributed deep learning using synchronous stochastic gradient descent, arXiv preprint arXiv:1602.06709, 2016

110. Ooi, Beng Chin and Tan, Kian-Lee and Wang, Sheng and Wang, Wei and Cai, Qingchao and Chen, Gang and Gao, Jinyang and Luo, Zhaojing and Tung, Anthony KH and Wang, Yuan, SINGA: A distributed deep learning platform, Proceedings of the 23rd ACM international conference on Multimedia, 2015

111. Louizos, Christos and Ullrich, Karen and Welling, Max Bayesian compression for deep learning, Advances in Neural Information Processing Systems, 2017

112. Zhao, Yue and Li, Meng and Lai, Liangzhen and Suda, Naveen and Civin, Damon and Chandra, Vikas, Federated Learnin

113. Even, Shimon and Goldreich, Oded and Lempel, Abraham, A randomized protocol for signing contracts, Communications of the ACM, 1985

114. Rabin, Michael O, How To Exchange Secrets with Oblivious Transfer, IACR Cryptology ePrint Archive, 2005

115. Sathya, Sai Sri and Vepakomma, Praneeth and Raskar, Ramesh and Ramachandra, Ranjan and Bhattacharya, Santanu, A Review of Homomorphic Encryption Libraries for Secure Computation, arXiv1812.02428, 2018