Secure Collaborative Environment for Seamless Sharing of Scientific Knowledge

Srikanth Yoginath, Mathieu Doucet, Debsindhu Bhowmik, David Heise, Folami Alamudun, Hong-Jun Yoon, and Christopher Stanley

Oak Ridge National Laboratory, Oak Ridge, TN, USA
yoginathsb@ornl.gov, doucetm@ornl.gov, bhowmikd@ornl.gov, heiseda@ornl.gov, alamudunft@ornl.gov, yoonh@ornl.gov, stanleycb@ornl.gov

Abstract. In a secure collaborative environment, tera-bytes of data generated from powerful scientific instruments are used to train secure machine learning (ML) models on exascale computing systems, which are then securely shared with internal or external collaborators as cloud-based services. Devising such a secure platform is necessary for seamless scientific knowledge sharing without compromising individual, or institute-level, intellectual property and privacy details. By enabling new computing opportunities with sensitive data, we envision a secure collaborative environment that will play a significant role in accelerating scientific discovery. Several recent technological advancements have made it possible to realize these capabilities. In this paper, we present our efforts at ORNL toward developing a secure computation platform. We present a use case where scientific data generated from complex instruments, like those at the Spallation Neutron Source (SNS), are used to train a differential privacy enabled deep learning (DL) network on Summit, which is then hosted as a secure multi-party computation (MPC) service on ORNL's Compute and Data Environment for Science (CADES) cloud computing platform for third-party inference. In this feasibility study, we discuss the challenges involved, elaborate on leveraged technologies, analyze relevant performance results and present the future vision of our work to establish secure collaboration capabilities within and outside of ORNL.

Keywords: differential privacy, secure multi-party computation, secure sharing of scientific knowledge

1 Introduction

Scientific innovations and advancements are the result of sustained collaborative efforts among scientists spread across multiple disciplines and institutions around the world. However, an environment of stiff competition exists within a scientific community to establish a leading edge in their respective domains. While the former demands an open exchange of ideas, the latter persistently promotes guarding of ideas as intellectual properties. Without a significant mechanism to exchange shareable ideas while guarding non-shareable ones, the scientific community would either result in a slow innovation rate or suffer from losing hard-earned intellectual property.

In the era of artificial intelligence and big-data machine learning, institutions hosting large scientific instruments have an opportunity to play a significant role in hastening the innovation cycle. Dataset generation, either through sophisticated scientific instruments or using high-fidelity simulations and their subsequent analyses, are the core functionalities that are carried out in the process of scientific exploration, which over time leads to scientific innovations. A realization of long-term scientific goals involves several experiments that might be independent or follow-on, and which generally result in a reduction of exploration space. In this regard, insights from these complex, intermediary and, by themselves, insufficient experimental steps performed during the course of scientific exploration could be extremely helpful for the progress of domain science innovations. Sharing such insights could significantly reduce the scientific exploration space and experimental time of fellow researchers. Similarly, protecting certain technical details and capabilities, while being able to share the experimental insights with a peer researcher, is essential. With this approach, we not only participate actively in the progress of science but also protect the intellectual property of the enabling technology that helps to maintain a technical edge. Cryptographic techniques, like secure multi-party computation and differential privacy play a significant role in realizing such a secure sharing platform. In this paper, we bring together existing cryptographic methods to demonstrate the feasibility of a secure environment for unhindered exchange of information across research facilities to accelerate innovations. We highlight, discuss, and demonstrate secure information sharing use cases that utilize instruments from the Spallation Neutron Source (SNS), high performance computing facility Summit and ORNL's private cloud, CADES.

1.1 Background and Motivation

Apart from the need to identify the significant scientific insights in the generated data, a method is required to communicate such insights given relevant inputs. To achieve this, the data needs to be learned to generate a machine learning (ML) model. The expectation is that the ML model would impart additional insights on the input data through prediction or classification. Such insights would help the domain scientists to streamline their subsequent process of experimental exploration. Several technologies need to come together to realize secure sharing

of scientific knowledge. At the hardware level, this task touches upon experimental facilities, high-performance computing (HPC) systems and commodity clouds. At the software level, we need software implementations of cryptographic techniques, ML models that support these techniques, scalable HPC implementations for training ML models and service-oriented architectures that support and securely expose the ML models to the outside world for secure inferences. Further, all of these wide-ranging technologies should interoperate to realize a secure collaborative environment. We designed our platform around two privacy computing capabilities: differential privacy and secure multi-party computation, which are described in further detail below.

Differential privacy (DP) employs a statistical strategy, where tunable noise is introduced either to the inputs, outputs, or at certain steps within an algorithm [1]. Using the TF_Privacy library [18, 2], we utilize (ε, δ)-DP: $Pr[M(D) \in S] \leq e^{\varepsilon} Pr[M(D^{'}) \in S] + \delta$, where an algorithm ($M$) is run on adjacent inputs/databases (D and $D^{'}$) to give any subset of outputs (S) and with ε and δ as privacy parameters (ε = upper bound on privacy loss, δ = probability for privacy guarantee to not hold). The addition of δ makes the method more generally applicable by relaxing the original ε-DP definition that only considers very strong, worst case conditions. Distinct advantages of (ε, δ)-DP are sequential composability and privacy guarantees through the calculated privacy loss. Algorithmic modules can be individually composed with (ε, δ)-DP and then sequentially constructed while maintaining privacy. With TF_Privacy, this allows us to introduce (ε, δ)-DP through the layers within the DL training models [2, 3], which we also have demonstrated for DL training on medical data scenarios [5]. The quantified privacy loss of DP also sets the privacy guarantees, up to the privacy budget threshold. Overall, DP for ML model training protects both the model and training data from model inversion and membership inference attacks, respectively [4].

Secure multi-party computation (MPC) is an encryption scheme that allows two or more parties to jointly compute a function over their private inputs without revealing these inputs [8, 9]. Here, we utilize the TF_Encrypted library [19, 10], which employs the SPDZ protocol [11] with additive secret sharing. Overall, MPC offers robust security to compute on an open resource, like Cloud, but can incur substantial communication overhead. We therefore leverage its capabilities for secure ML model inferencing, where practical performance is achieved.

The secure MPC computation involves two entities A and B. A has a trained model on certain data, called M. B holds another data D and wants to use model M to obtain some helpful insights on their data. However, both A and B consider their holdings M and D, respectively, as their intellectual property that provide them a leading edge in their respective domains. Secure MPC allows such interactions to happen using the encrypted model and data on an independent set of servers. TF_Encrypted [19] uses three servers S_0, S_1 and S_2 for this purpose and establishes secure communication MPC protocols over gRPC [21].

1.2 Organization

In Section 2, we discuss the possible use case scenarios for secure information sharing, with a detailed description of one particular case. In Section 3, we discuss the ML model development and training process with and without DP. We demonstrate how a trained ML model is securely shared among users, along with the relevant performance results in Section 4. We summarize, discuss our future work, and conclude the paper in Section 5.

2 Use Case Scenarios

Insights from physical experiments and computational simulations are known drivers of scientific innovations. Regardless of whether the data from the experiments and simulations are complementary or similar, they both contribute informative results. The advancements in ML-based technologies have provided a means by which significant insights from such datasets can be exposed through classification or prediction models. The advancements in web technologies have provided a means to share such models for inferencing across a wide community, and audiences, around the world.

Such exchange scenarios include (a) Experimental or computational scientists exchanging insights with their peers (b) Experimental scientists exchanging insights with computational scientists for validation purposes (c) Computational scientists exchanging insights with experimental scientists that could be helpful for the design of experiments (d) Real-time steering of simulations or experiments using insights from previous experiments or simulations, respectively (e) Multiple collaborating groups can come together to train ML models that can be shared without divulging any details of their datasets to each other.

As a more in-depth example, we consider a practical use case scenario for a scientific user facility that involves secure collaboration between instrument operators and facility users. A fundamental collaboration is required for data quality assessment of neutron scattering instruments. We will also see that the needs of this scenario brings together the experimental facilities, HPC and the Cloud infrastructure resources.

2.1 Data Quality Assessment from Small-Angle Neutron Scattering (SANS) instrument

Experiments involving sophisticated instruments usually are a multi-institutional operation. Also, several types of instruments typically are hosted by an institutional facility, like SNS at ORNL, and such instruments are made available to experimental scientists either from academia or industry (outside users). These experimental processes are a collaborative effort between the outside users and instrument scientists at the facility. During such procedures, special scenarios can exist where a user would like to guard all data generated. Examples are industry partners that will generate proprietary data without intent to publish

in the open literature, and scientific researchers working in a highly competitive area where the new, timely results are sensitive. However, to ensure the efficient use of the user's beamtime, including collection of the highest quality results, the instrument scientists need access to certain details on the collected data. Hence, a security concern arises, as the data generators would not want to freely share their datasets. Such a practical scenario can arise to create barriers and ultimately prohibit the full utilization of a scientific instrument.

To frame the challenge within a specific example, consider a company or industry partner is using the extended Q-range small-angle neutron scattering (EQ-SANS) instrument at SNS [12, 13] to measure a series of proprietary pharmaceutical formulations, and the data needs to be protected during collection. However, the instrument scientists would like to assist, with their expertise, in evaluating how the experiment is progressing and inform the decision making (e.g. samples to prepare and run) during the data collection. If some samples are showing very low signal-to-noise, they may need to be measured for longer exposure times and/or prepared at higher concentration. If a sample appears to show unanticipated results, like sample aggregation, early detection and rectifying measures would be important. While these activities normally occur during a conventional data collection, the challenge here with a sensitive-data user is to maintain privacy of all data except the necessary information needed to properly assist in still performing an optimal experiment. Current practices with industry users that plan to generate sensitive data without publishing involve lengthy measures: the participating instrument scientists sign a non-disclosure agreement (NDA), and isolation approaches are employed at the instrument and with the data. A more efficient and versatile solution, which we propose, is to establish seamless data privacy computing methods within the experimental data collection and initial analysis phases, such that only pertinent, pre-defined and agreed upon information is shared. By incorporating privacy methods, we also envision the capability to train ML models using multitudes of instrument data that protects the training data privacy and can then inform on these sensitive data experiments in real-time without disclosing the raw data information. Next, we delineate these steps, and associated details, by training a privacy-enabled ML model on SANS data, utilizing DP, to then perform secure inferencing on the model using secure MPC. Of note, the dataset generated and used below is meant to serve as a starting point for the given scenario. The chosen classes were based on current data availability. For a real use case, we would include more classes, along with more training examples per class. Stemming from this particular scenario, the privacy tools can be extended to afford general data sharing and analytics among scientific user facility users with other collaborators and researchers where concerns over data sensitivity are encountered.

2.2 Data Collection

The SANS data were obtained from the EQ-SANS instrument at SNS. A series of standard calibration sample measurements, along with data from two unpublished scientific experiments (data generated by C.S.), were used to construct

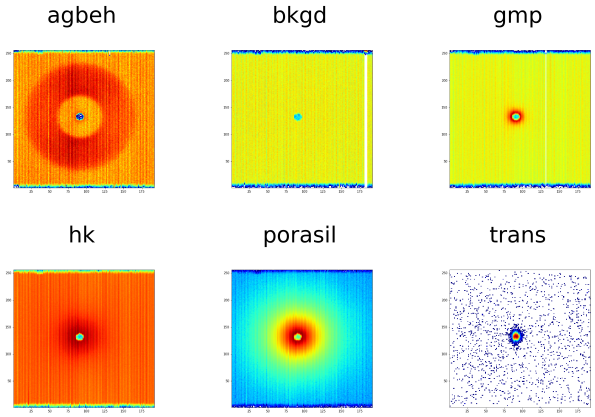


Fig. 1. Representative SANS detector images for the 6 classes. These specific sample classes generalize to some of the common scattering patterns obtained in experiments: peaks (agbeh), background/low intensity samples (bkgd), rod-like particles (gmp), globular particles (hk), strong intensity samples (porasil), and transmission measurements (trans).

the raw SANS detector image dataset. The standard samples included silver behenate (AgBeh), porous silica (Porasil) [14], hydrogenated water (H_2O), and deuterated water (D_2O). Scientific SANS data on a protein, hexokinase (HK), and self-assembled structures of the guanosine derivative, 2'-deoxyguanosine-5'-monophophate (dGMP), measured over multiple experiments also were included into the dataset. The HK and dGMP experiments also had background measurements, which were buffer and salt in H_2O/D_2O. The images for these background runs were combined with the H_2O and D_2O standard run images to form a background category (bkgd) in the dataset. In addition, a transmission mode measurement is made on every sample, and those detector images were included as a transmission category (trans) in the dataset.

The physical detector on the EQ-SANS instrument is 1 x 1 m in size and provides 192 x 256 (nearly square) pixel resolution. All measurements comprised a range of instrument settings, using 1.3 to 8 m sample-to-detector distance (SDD) and 1 to 20 Å wavelength neutrons, with a nominal 3.5 Å bandwidth at each setting. The detector images used to make the dataset were raw neutron counts, covering the entire time-of-flight (ToF) and exposure time measured for each sample, and summed for each detector pixel. Overall, the images were grouped into 6 classes, with each image assigned one of the following single labels: agbeh, bkgd, gmp, hk, porasil, trans (Figure 1). These classes can be generally considered as scattering patterns of: peaks, background (or low intensity samples), rod-like particles, globular particles, strong intensity samples, and transmission (direct beam) mode measurements, respectively.

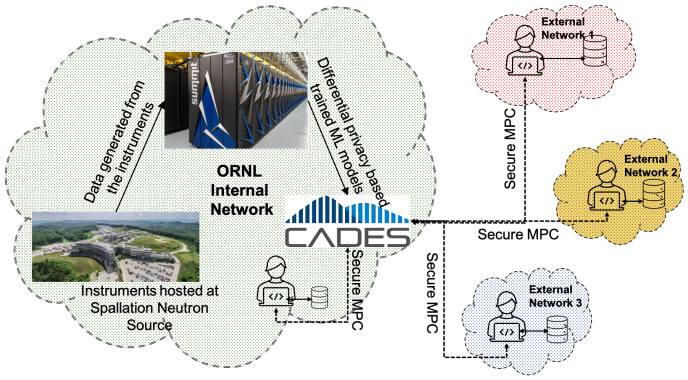


Fig. 2. Secure information sharing use case schematic involves data generated from the EQ-SANS instrument used to train ML models on Summit, an HPC platform, and finally securely shared among internal and/or external scientists through CADES, a Cloud platform.

2.3 ML Models and Privacy

We used a convolutional neural network (CNN) to train on collected data, which can be used to classify datasets from future experiments. The ML models that are built by the operators classify the new set of incoming data from the experiments to a certain category, which provides the operators necessary information for data quality assessment. By using secure sharing methods, the experimental scientists can ensure their data is protected from leakage while still gaining assistance and insights from the instrument operators.

Using DP, the ML models can be trained to ensure the information privacy of the ML model. However, training of the privacy models are more compute intensive than their non-privacy models. This, along with the resolution and volume of the incoming data, creates a demand for HPC resources. Once trained, these models can be shared with the external world, as shown in Figure 2, through application services hosted in a community Cloud platform. DP protects the model and training data from model inversion and membership inference attacks. However, since DP does not inherently provide collaborative setups or secure a client's test data, we use secure MPC protocols. The secure MPC protocols that can be realized over public Cloud infrastructure ensures both data and model privacy, addressing the privacy concerns of both data and model owners.

3 Machine Learning Model Training

3.1 Model Setup

Data Pre-processing Images were first center-cropped to remove white-space, resized to $n \times n$ ($n = 64, 128, 224$) and rescaled by the factor $1/255$. Horizontal

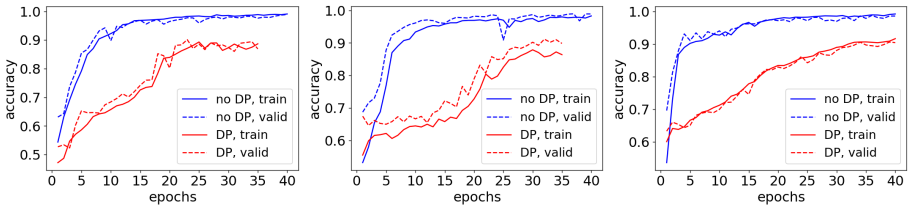
and vertical flip image data augmentation was applied to avoid the training algorithm from learning on any characteristic detector patterns, like tubes turned off or top/bottom edges, for any subset of the data. The data was split into training/validation/testing (64%/16%/20%), where the test data paths were saved for each run to use the exact same samples for MPC inferencing.

Model To demonstrate the use case with scientific user facilities data, we performed deep learning (DL) training, both without and with DP, on our generated dataset of SANS detector images comprised of 6 classes (see Figure 3). For the CNN model training on the SANS detector images, VGG9 (for $n = 64$) and VGG16 (for $n = 128$, 224) models [22] were implemented with Keras in Tensorflow 2.4.1 [23]. For the standard, non-DP training, the runs used a learning rate = 0.008, batch size = 16, epochs = 40, with a stochastic gradient decent (SGD) optimizer. For DP training, the TF_Privacy library (v0.5.1) [18] was used. Similar settings to the standard training were used for comparison, along with the additional privacy hyperparameters. The only adjustments were to use a learning rate = 0.004, epochs = 35, and the TF_Privacy DPKerasSGDOptimizer. Privacy hyperparameters used were noise multiplier = 0.76, L2 norm clip = 1.0, and microbatches = 4. These settings were reached based on initial runs using typical values and ranges described in TF_Privacy, along with previous reported settings [2].

To consider (ε, δ)-DP for machine learning, as we do here with TF_Privacy, ε specifies an upper bound on the probability for a model output to vary if any single training sample is included or excluded. For many practical uses, ε between 0.1-10 is typically used [6, 7]. The δ is set in the training and reflects the leak rate probability of the sample training inputs, where $\delta \leq 1/$(training size) is a typical range to target. Overall, we could continue to optimize across the privacy parameters for the DL training, but these results fully satisfy the (ε, δ)-DP conditions while also yielding reasonable accuracy. They also remain as similar to the non-DP training for best baseline comparison.

3.2 Model Training

With a VGG16 model architecture and images resized to 224×224 input size, the standard (no DP) training achieved ∼99% accuracy after 40 epochs (Figure 3). In comparison, the DP settings (see Methods for details) reached a plateau of ∼88% accuracy by 35 epochs (see Figure 3) and additionally provided a (ε, δ)-DP privacy guarantee of (9.35, 1.79e^{-3})-DP. The reduction in training accuracy is expected when adding DP, as this is part of the privacy-utility tradeoffs of the method. We also performed training runs using VGG16 and 128×128 input size, and a smaller network and input size (VGG9 and 64×64, respectively), for comparison and for MPC inferencing, described below. The corresponding accuracy curves are shown in Figure 3b and Figure 3c. The VGG16 model training with 128×128 input size and DP was trained for 35 epochs and also achieved (9.35, 1.79e^{-3})-DP. The VGG9 model training with DP was trained for 40 epochs and achieved (10.11, 1.79e^{-3})-DP.



(a) VGG16, 224×224 data (b) VGG16, 128×128 data (c) VGG9, 64×64 data

Fig. 3. VGG training accuracy curves that compare without and with DP, and for different image input sizes. The privacy-utility balance is noticeable in all cases, where each DP training plateaus at a lower accuracy compared to no DP (specific DP settings used are listed in the main text).

We make the following observations on the model training with DP, compared to the non-DP case:

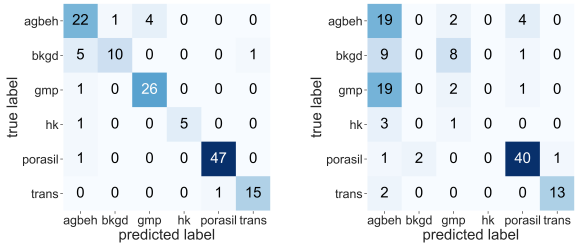
1. A lower learning rate was required to maintain a stable and monotonic accuracy growth (and commensurate loss function decay) due to the added noise and gradient clipping.
2. The ε increases with the number of training epochs, so we performed a series of runs with varying epochs to reach 35 for DP training of the VGG16 network, which yielded a reasonable privacy-utility balance. Fewer epochs did not reach sufficient accuracy and more epochs resulted in $\varepsilon > 10$.
3. We performed training runs with larger noise multiplier values (> 0.67), but they often resulted in sub-optimal accuracy and, therefore, utility. This is reasonable, given the relatively small training size we are using for our example here. DP is best for generalizing over large data, so it is expected that utilizing larger, aggregated datasets would provide more utility along with capacity for higher privacy settings in real use cases.

Further, when performing inferencing on the test data, we observe a loss in accurately predicting certain classes (bkgd, gmp, hk). This can be seen by comparing the confusion matrix for non-DP and DP training (Figure 4a, 4b, respectively).

3.3 Differential Privacy Cost

To assess training performance, we also varied image input size (from 64×64 up to 320×320) for non-DP and DP training with the VGG network (Figure 5). Again, the 64×64 input used a smaller, VGG9, model while the other input sizes used the VGG16 model. We found a strong divergence in the DP training time, relative to the non-DP, for increasing input size. Also, the DP training failed at the highest input size (320×320) due to GPU device memory limitations.

At the input size used for the results shown above (with 224×224), DP training is ∼2× slower compared to non-DP training. We observed that adjusting DP



(a) non-DP (b) DP

Fig. 4. Confusion matrix plots for VGG16 testing (224×224 data) that show accuracy is maintained in DP models for some classes (agbeh, porasil, trans) but with a discrepancy in accurately predicting certain other classes (bkgd, gmp, hk).

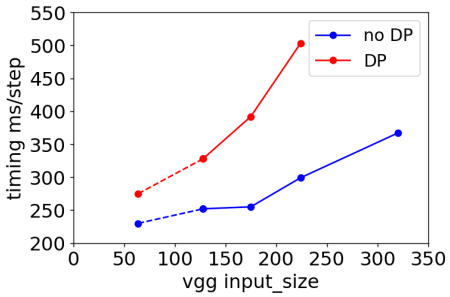


Fig. 5. Cost of training VGG network with DP. The DP training was observed to be almost 2× slower compared to non-DP training and could get much slower for increased privacy levels.

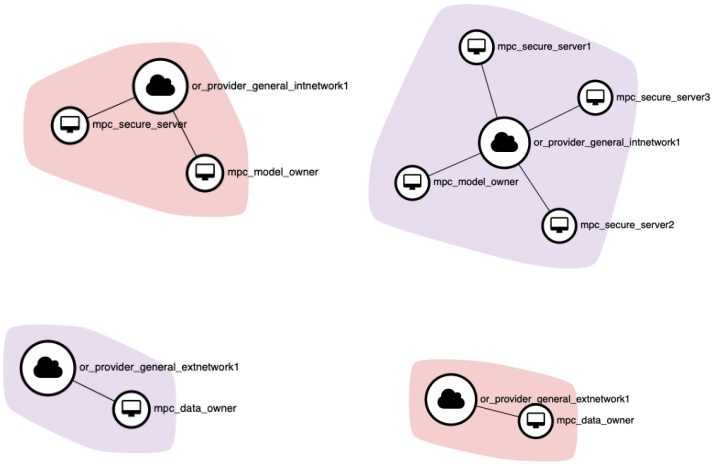
parameters for increased privacy levels (e.g. increasing noise multiplier) can impact training speed even further, with ∼5× slower training times compared to non-DP. With the incremental, step training time for the 224×224 input size and DP settings used here (503 ms for batch size = 16), 100K images with 40 epochs would require ∼35 hr of training time on a single V100 GPU.

To frame these data volumes and compute timings within a possible SANS data collection, there are three high-throughput SANS instruments at ORNL: EQ-SANS at SNS, Bio-SANS and GP-SANS at the High Flux Isotope Reactor (HFIR) and each can operate at a rate of ∼10 min/sample and these facilities operate at up to ∼70% of the year. Given these trends, a large volume of over 100K images per year can be obtained by SANS experiments. Training CNN models with and without privacy over such a large volume of data would demand a significant amount of computational resources and such needs can be met using an HPC system like Summit.

4 Secure Inference

4.1 Secure MPC Setup

A secure MPC test setup was developed over CADES Cloud platform. To enable secure inference by an external entity on an internally hosted ML model, we used TF_encrypted, a python module that implements secure MPC protocols. The setup prescribed by TF_encrypted involves a ML model hosting server (M_s) and a client with test datasets (C_d) that do not see each other but they interact through three intermediate secure servers (S_1, S_2 and S_3). We developed containers to host model, intermediate secure servers and the data owner who performs inferences. Each of these containers were spawned on the virtual machine (VM) instances of CADES cloud. Three different setups were used in our experiments.



(a) Setup 1 (b) Setup 2

Fig. 6. Setup 1: single host for secure servers and Setup2: separate hosts for secure servers on CADES for secure MPC evaluation.

- **Setup 1:** Each VM instance involved in the experiments were of type *m1.2xlarge* supporting 16 virtual CPUs, 32 GB of memory and 40GB of disk space. VM instances *mpc_model_owner* host container with M_s, *mpc_data_owner* hosts a container with C_d and *mpc_secure_server* hosts a container with secure servers S_1, S_2 and S_3 listening on different ports. Figure 6a represents this setup.
- **Setup 2:** Three VM instances (*mpc_secure_server1*, *mpc_secure_server2* and *mpc_secure_server3*) of type *m1.2xlarge* were used to host secure servers S_1,

S_2 and S_3. Two VM instances (*mpc_model_owner* and *mpc_model_owner*) of type *m1.large* supporting 4 virtual CPUs, 8 GB of memory and 80 GB of disk space, were used to host (M_s) and (C_d). Figure 6b represents this setup.
– **Setup 3:** Two VM instances (*mpc_model_owner* and *mpc_model_owner*) of type *m1.large* supporting 4 virtual CPUs, 8 GB of memory and 80 GB of disk space, were used to host (M_s) and (C_d). This setup was used to estimate baseline performance without secure MPC.

4.2 Secure MPC Performance

We performed 100 random inferences over Cloud-based secure MPC setup on CADES for both VGG models with and without differential privacy. At each inference cycle, the client sends an encrypted single image over the network to the intermediate secure servers. A model server executing as a service on a different VM instance interacts with the intermediate secure servers to fulfill the inference request of the client. To measure the performance, we polled the container M_s, container with intermediate servers (S_1, S_2 and S_3) and a container hosting C_d using *docker stats*. The VGG9 network comprises only the first six convolutional layers and 3 dense layers as opposed to 13 convolution layers and 3 dense layers in the usual VGG16 network.

Table 1. Accuracy and Runtime performance

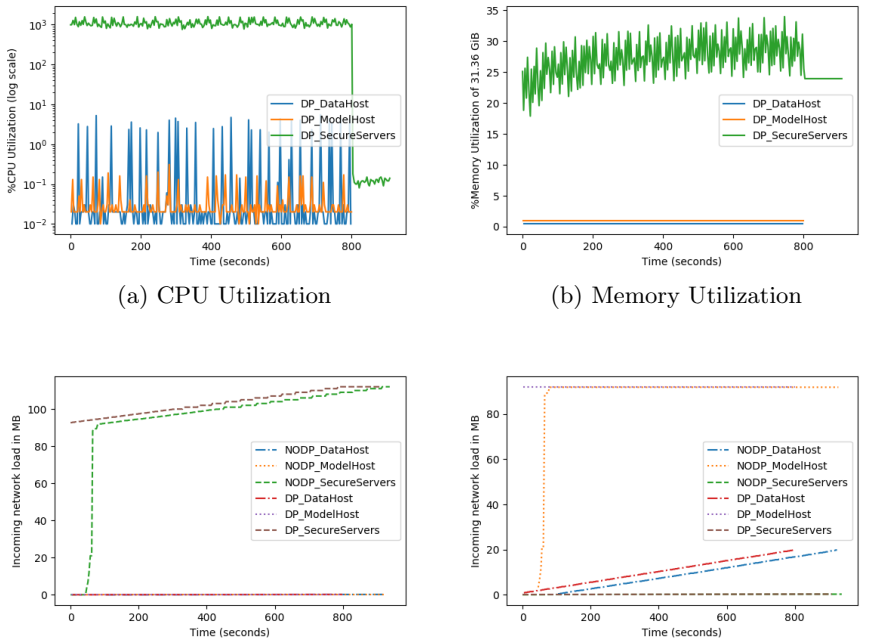
Diff-Privacy	Accuracy	Inference time
No	0.96	8.82 ± 0.43
Yes	0.66	8.3 ± 0.31

Secure Inference of VGG9 on 64×64 Data over CADES Setup 1 The accuracy and inference time readings are tabulated in Table 1. A high accuracy is observed for the model without DP. However, as anticipated, the accuracy of the DP model is lower.

In Figure 7a and Figure 7b we show the instantaneous CPU and memory utilization percentages in the *model_host*, *data_host* and the *secure_servers*. In Figure 7c and Figure 7d we show the cumulative incoming and outgoing network loads during inferencing ML models with DP and without DP.

From Figure 7c and Figure 7d, we see that the cumulative network loads remain similar for inferencing on ML models with and without DP. This is expected and we also observed similar trend in the CPU and memory utilization. Hence, we show CPU and memory utilization plots for inferencing models using DP, which is almost the same as the one without DP.

Instantaneous CPU utilization curves in Figure 7a show regular spikes corresponding to consecutive inference activities. A very low (<10%) virtual CPU utilization and almost negligible amount of memory utilization can be observed



(a) CPU Utilization (b) Memory Utilization

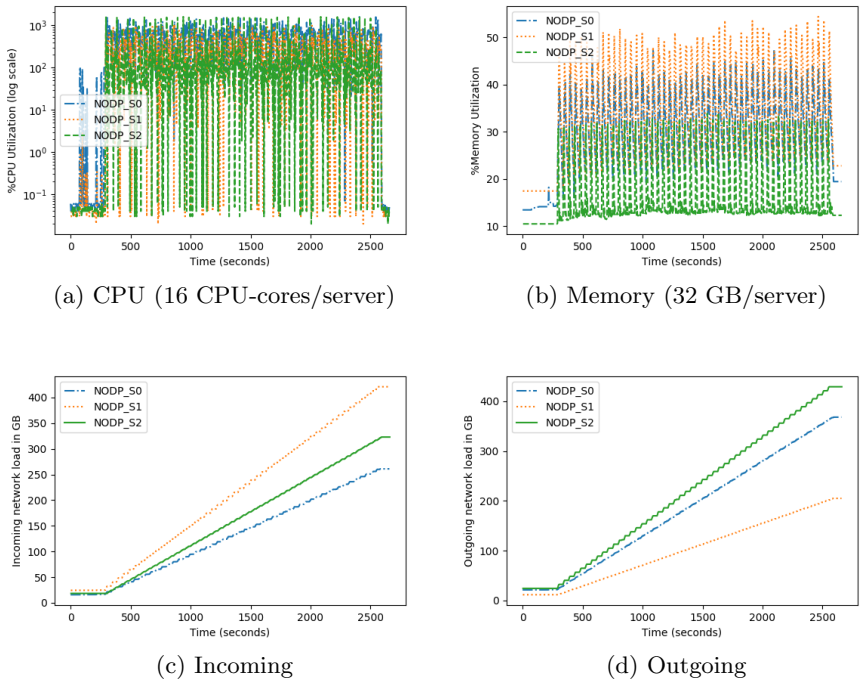
(c) Cumulative incoming network load (d) Cumulative Outgoing network load

Fig. 7. Percent CPU, memory and network utilization in the data host, model host and the secure servers during secure MPC inference. The plots suggest that the secure servers are computationally intensive, while the model owners and data owners computationally lightweight.

in the *data_owner* and *model_owner*. However, the VM hosting the secure servers (S_1, S_2 and S_3) appears to utilize almost all of 16 virtual CPU cores and around 30% (∼11GB) of memory.

The cumulative outgoing network load in Figure 7d suggest that the model is communicated once to the *secure_servers* by the *model_owner* and the *data_owner* consecutively sends data at regular intervals for inferencing. Further, the outgoing network load from the *secure_servers*, which is an inference label, is almost negligible. The cumulative incoming network load in Figure 7c suggest the *secure_servers* receives the network traffic from both *data_owner* and *model_owner*. Hence, from the plots in Figure 7, we can conclude that much of the computational tasks are carried out on secure servers. A single CPU VM instance with minimal memory 5% of their 32GB of memory (<3GB) should suffice the computational needs of the model host and the data host for VGG9 network inferencing data of dimension 64×64.

Observations (a) Similar runtime performance and resource utilization is observed during secure inference of models with and without DP. (b) The secure



(a) CPU (16 CPU-cores/server) (b) Memory (32 GB/server)

(c) Incoming (d) Outgoing

Fig. 8. Percent CPU, memory and network utilization across secure servers S0, S1 and S2 during secure inference in a multi-node setup show that secure server computations are compute-,memory- and network-wise expensive.

servers are computationally intensive, while the model owners and data owners just communicate their encrypted model and data to the secure servers.

Secure Inference of VGG16 on 128×128 Data over CADES Setup 2 This test used the VGG16 model without DP support, since the performance results were almost similar. The same experimental procedure of random image inferences was performed over CADES Setup 1. However, after two successful secure inferences, the execution stalled or significantly slowed down. This behavior of significant slow down could be due to page swapping due to lack of memory at the secure servers. The first run took around 200 seconds to complete and the next run took almost 37 seconds for completion.

Splitting the secure servers into three different containers and having a large resource allocation for each secure server overcomes this resource relevant problem. Hence, we use CADES Setup 2 to understand the CPU, memory and network resource utilization among the *secure_servers* with the larger ML model VGG16 working on a larger dataset input size 128×128.

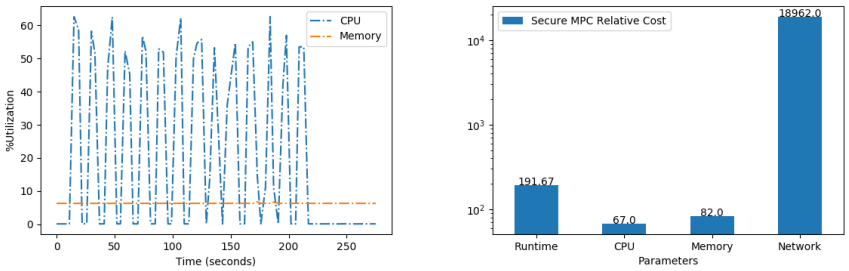
We performed 50 random single image inferences over CADES Setup 2. In Figure 8 instantaneous CPU and memory utilization percentages and cumulative network load plots among secure servers (S_0, S_1 and S_2) that are hosted as separate containers on three different VMs. The regular spikes seen in Figure 8a in all the servers correspond to the relevant computations. With a maximum instantaneous percent utilization of 1600%, 1000% and 1600%, the servers seem to significantly utilize all virtual CPU-cores. As seen in Figure 8b memory utilization of $S_1 > S_0 > S_2$, a maximum memory utilization of 47% (14.8 GiB), 54% (17 GiB) and 44% (10.6 GiB) was observed in servers S_0, S_1 and S_2, respectively. The cumulative network load measuring several hundreds of gigabytes were observed to be exchanged between the servers for just 50 inferences. These numbers indicate that the Tf_encrypted implementation of secure MPC is not only network-wise expensive but are also memory-wise and compute-wise expensive. Further the turn-around time for inferencing a single 128×128 data image over a CADES Setup 2 configuration was around 46 seconds on average. Comparing this runtime with our similar experiment on Setup 1, we can see almost a fourth of the 46 seconds corresponds to network communications between the servers hosted on different VMs.

Observations: (a) Secure MPC implementation of Tf_encrypted is compute-, memory- and network-wise expensive.

4.3 Secure MPC Cost

To estimate the cost of secure MPC, we used CADES Setup 3 to perform inference without secure MPC. We used Tensorflow Serving [20] to setup a SANS inference service at *model_owner*, which is accessed by the *data_owner* using the REST API for inference. In this setup, the *data_owner* sends the input data of dimension 128×128 for inference and the *model_owner* performs are the requisite computations and communicates the result to the *data_owner*. Hence, the network load is significantly low. We polled the container at the *model_owner* VM to obtain the runtime statistics using docker stats. We performed 100 random single image inferences over CADES Setup 3. Figure 9a shows the instantaneous CPU and memory utilization percentage. The cumulative network load mostly corresponds to the communication of input images to the *model_host*, which was around 106 MB. This is seen from plots in Figure 9a with a maximum of 63% of CPU load, maximum of 6.4% (540 MB) of memory load. From these readings we can infer that the secure MPC inference operation is compute-, memory- and network-wise extremely expensive in comparison with the regular inference. Further, a regular inference is also extremely fast with a turn-around time of 0.24 seconds.

Figure 9b shows exactly how expensive in terms of runtime, CPU load, memory load and network load. The CPU and memory loads are over an order-of-magnitude expensive, with memory load closer to two orders-of-magnitude. Runtime performance is over two orders-of-magnitude slower. The network load with over four orders-of-magnitude is the most expensive of all. These numbers



(a) CPU and Memory (b) Secure MPC Cost

Fig. 9. Privacy cost in DP training and secure MPC inference.

were obtained from the results from the experimental runs carried out for secure inference on CADES Setup 2 and regular inference on CADES Setup 3. A ratio of per inference runtimes were used to determine relative runtime cost. The maximum instantaneous CPU and memory utilization from these two experiments were used to determine relative CPU and memory cost. We aggregated the final cumulative incoming network load values of all the secure servers and divided them by the number of secure inferences performed (50) to obtain network load per secure inference. This network load for regular inference corresponds to the input data communicated by the *data_owner*, which was ∼1MB. The ratio (network load per secure inference) / (network load per regular inference) was used to obtain relative network cost for secure inference.

5 Conclusion and Future Work

We started the paper highlighting the need for secure information exchange and its feasibility using existing technologies. We showcased a practical use case scenario of secure information exchange between EQ-SANS instrument operators at SNS and facility users at academic institutions or industry. The information exchange was formulated using a ML model, such that the privacy of the model and the data were preserved. To achieve this a secure ML model using DP algorithms and secure inference using MPC protocols were used. We performed training of a deep learning VGG network with and without DP on a Summit node. With the performance readings from the model training process and the expected data resolution and volume, we emphasized the need for an HPC facility to handle this compute-intensive model training process. We developed secure inference services using containers over VM instances from the CADES Cloud platform. A performance study of MPC-based secure inference involving multiple VM instances and containers were carried out and the results were presented. With these tasks, we implemented a full life-cycle iteration of a SANS use case scenario where privacy preserving algorithms were used. We

successfully demonstrated its practical feasibility using ORNL resources namely, the SNS EQ-SANS instrument, Summit HPC and the CADES Cloud infrastructure. Performance of DP and secure MPC training and inference were recorded. The relative cost of privacy for training with DP and inference with secure MPC were presented. Despite the higher resource cost and significantly slower runtimes, the training with DP and inference with secure MPC for privacy were shown to be realistically feasible.

The presented SANS use case can be extended to other high-throughput SANS instruments at ORNL: Bio-SANS and GP-SANS HFIR. This work can also be extended to train an agent to automate the process of experimental setup using reinforcement learning to a certain extent. Further, this method can be used as a general template to achieve secure exchange of scientific knowledge amongst local and non-local researchers. In this regard, we are working on secure model and inference for a prediction class of supervised learning problems using data from the reflectometry instruments at SNS [15, 16]. We are also working on secure inference of an unsupervised learning class of problems with a convolutional variational auto-encoder model using molecular dynamics simulation data to identify and analyze the microstates in biomolecules [17].

Acknowledgements

This work was supported by the Laboratory Directed Research and Development (LDRD) program of Oak Ridge National Laboratory, under LDRD project 9831. A portion of this research at ORNL's Spallation Neutron Source was sponsored by the Scientific User Facilities Division, Office of Basic Energy Sciences, U.S. Department of Energy. C.S. acknowledges the EQ-SANS beamline staff: Changwoo Do, Carrie Gao, and William Heller, that also assisted in the calibration samples data collection over the time period. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This research used resources of the Compute and Data Environment for Science (CADES) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. We would like to acknowledge the timely support and assistance provided by Chris Layton and Daniel Dewey. We very much appreciate their help and support.

References

1. Dwork, C., Roth, A. (2014). The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 9(3-4), 211-407.
2. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K. and Zhang, L. 2016. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery, New York, NY, USA, 308–318.
3. Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., Erlingsson, Ú. (2018). Scalable private learning with PATE. arXiv:1802.08908

4. Shokri, R., Stronati, M., Song, C., Shmatikov, V. Membership inference attacks against machine learning models. In 2017 IEEE Symposium on Security and Privacy (SP) (pp. 3-18). IEEE.
5. Yoon, H.J., Klasky, H.B., Durbin, E.B., Wu, X.C., Stroup, A., Doherty, J., Coyle, L., Penberthy, L., Stanley, C., Christian, J.B. and Tourassi, G.D., 2020. Privacy-Preserving Knowledge Transfer with Bootstrap Aggregation of Teacher Ensembles. In: Gadepally V. et al. (eds) Heterogeneous Data Management, Polystores, and Analytics for Healthcare. DMAH 2020, Poly 2020. Lecture Notes in Computer Science, vol 12633. Springer, Cham.
6. Lee, J., Clifton, C. 2011. How much is enough? choosing ε for differential privacy. In International Conference on Information Security, 325-340. Springer, Berlin, Heidelberg.
7. Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B. C., Roth, A. 2014. Differential privacy: An economic method for choosing epsilon. In 2014 IEEE 27th Computer Security Foundations Symposium, 398-410.
8. Hazay, C., Lindell, Y. (2010). Efficient secure two-party protocols. Information Security and Cryptography.
9. Evans, D., Kolesnikov, V., Rosulek, M. A pragmatic introduction to secure multi-party computation. NOW Publishers, 2018.
10. Dahl, M., Mancuso, J., Dupis, Y., Decoste, B., Giraud, M., Livingstone, I., Patriquin, J., Uhma, G., 2018. Private machine learning in tensorflow using secure computation. arXiv:1810.08130
11. Damgård I., Pastro V., Smart N., Zakarias S. (2012) Multiparty Computation from Somewhat Homomorphic Encryption. In: Safavi-Naini R., Canetti R. (eds) Advances in Cryptology – CRYPTO 2012. CRYPTO 2012. Lecture Notes in Computer Science, vol 7417. Springer, Berlin, Heidelberg.
12. Zhao, J. K., Gao, C. Y., Liu, D. The extended Q-range small-angle neutron scattering diffractometer at the SNS. J. Appl. Crystallogr. 43, 1068–1077 (2010).
13. Heller, W., Cuneo, M., Debeer-Schmitt, L., Do, C., He, L., Heroux, L., Littrell, K., Pingali, S. V., Qian, S., Stanley, C., Urban, V., Wu, B., Bras, W. The suite of small-angle neutron scattering instruments at Oak Ridge National Laboratory. J. Appl. Cryst. 2018, 51, 242–248.
14. G. D. Wignall, F. S. Bates, Absolute calibration of small-angle neutron scattering data. J. Appl. Crystallogr. 20, 28–40 (1987).
15. Doucet, M., et al, Machine learning for neutron reflectometry data analysis of two-layer thin films, Mach. Learn.: Sci. Technol. 2 035001 (2021).
16. Maranville B B, Kienzle P, Sheridan R, Doucet M, Nelson A, Hoogerheide D P, Book A and Ghose R 2020 reflectometry/refl1d: v0.8.13 https://github.com/reflectometry/refl1d
17. Bhowmik, D., Gao, S., Young, M.T. et al. Deep clustering of protein folding simulations. BMC Bioinformatics 19, 484 (2018).
18. TensorFlow Privacy. https://github.com/tensorflow/privacy
19. TF_Encrypted: Encrypted Learning in Tensorflow. https://github.com/tf-encrypted
20. TF_Serving: Serving Models. https://www.tensorflow.org/tfx/guide/serving
21. gRPC: A high performance, open source universal RPC framework. https://grpc.io/
22. Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.
23. Martín Abadi, et al., TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.