PRIVACY AND INTEGRITY PRESERVING TRAINING USING TRUSTED HARDWARE

Hanieh Hashemi
ECE Department
University of Southern California
Los Angeles, CA 9007
hashemis@usc.edu

Yongqin Wang
ECE Department
University of Southern California
Los Angeles, CA 9007
yongqin@usc.edu

Murali Annavaram
ECE Department
University of Southern California
Los Angeles, CA 9007
annavara@usc.edu

ABSTRACT

Privacy and security-related concerns are growing as machine learning reaches diverse application domains. The data holders want to train with private data while exploiting accelerators, such as GPUs, that are hosted in the cloud. However, Cloud systems are vulnerable to the attackers that compromise privacy of data and integrity of computations. This work presents DarKnight, a framework for large DNN training while protecting input privacy and computation integrity. DarKnight relies on cooperative execution between trusted execution environments (*TEE*) and accelerators, where the TEE provides privacy and integrity verification, while accelerators perform the computation heavy linear algebraic operations.

1 INTRODUCTION

The need for protecting input privacy in Deep learning is growing rapidly in many areas. Many of the data holders are, however, not machine learning experts. Hence, data holders are relying on machine learning as a service (MLaaS) platforms (Microsoft, 2020; Google, 2020; Amazon, 2020). These services incorporate ML accelerators such as GPUs for high performance and provide easy to use ML runtimes to enable data holders to quickly set up their models and train. While these platforms lower the steep learning curve, they exacerbate the users' concern regarding data privacy.

This work proposes DarKnight, a framework for accelerating privacy and integrity preserving deep learning using untrusted accelerators. DarKnight is built on top of an MLaaS platform that uses unique collaborative computing between the *TEE* and GPU accelerators to tackle both privacy and security challenges. The data holder places their data, whether for training or inference, within the TEE of a cloud server. TEE provides hardware-assisted security for any data and computing performed within the trusted code base. DarKnight uses TEE to encode input data using a customized matrix masking technique and then uses GPUs to accelerate DNN's linear computations on the encoded data. Linear operations (convolution, matrix multiplication, etc) are significantly faster on a GPU compared to a TEE-enabled CPU. Therefore, DarKnight distributes these compute-intensive linear operations to GPUs. DarKnight's usage of TEEs is limited to protecting the privacy of data through a customized matrix masking and performing non-linear operations (ReLU, Maxpool).

TEE-GPU collaboration is first used in (Tramer & Boneh, 2018) for inference. However, the method cannot be used for training as elaborated in their paper. Several prior works on protecting privacy use cryptography techniques on Finite Fields to provide data privacy. Such approaches limit their usage to arithmetic on quantized models (Mohassel & Zhang, 2017; Gascón et al., 2017; So et al., 2019; Wagh et al., 2019; Juvekar et al., 2018). Quantization for deep learning is a challenging task. DarKnight supports *floating point* model training and control the information leakage by encoding parameters. DarKnight can also detect any malicious activities of untrusted GPUs by its computation

integrity feature. Furthermore, DarKnight can protect privacy and integrity even in the presence of a subset of colluding GPUs that try to extract information or sabotage the computation.

2 Related work and Background

Table 1: Comparison of applications and security guarantees of various prior techniques on neural networks' security

Method	Training	Inference	DP	MPC	HE	TEE	Data Privacy	Model Privacy(Client)	Model Privacy(Server)	Integrity	GPU Acceleration	Large DNNs
SecureNN (Wagh et al., 2019)	•	•	○	•	○	○	•	•	•	○	•	○
Chiron (Hunt et al., 2018)	•	•	○	○	○	•	•	•	•	•	○	○
MSP (Hynes et al., 2018)	•	•	○	○	○	•	•	•	•	•	○	○
Gazelle (Juvekar et al., 2018)	○	•	○	○	•	○	•	○	○	○	•	•
MiniONN (Liu et al., 2017)	○	•	○	•	•	○	•	•	○	○	•	•
CryptoNets (Gilad-Bachrach et al., 2016)	○	•	○	•	•	○	•	•	○	○	•	•
Slalom (Tramer & Boneh, 2018)	○	•	○	○	○	•	•	•	○	•	•	•
Origami (Narra et al., 2019)	○	•	○	○	○	○	•	○	○	○	•	•
Shredder (Mireshghallah et al., 2020)	○	○	○	○	○	•	•	○	○	○	•	•
Delphi (Mishra et al., 2020)	○	•	○	•	•	•	•	•	○	○	•	•
DarKnight	•	•	○	○	○	•	•	•	○	•	•	•

There are a variety of approaches for protecting input and model privacy and computation integrity during DNN training and inference. These methods provide different privacy guarantees Mirshghallah et al. (2020). *Homomorphic encryption (HE)* techniques encrypt input data and then perform inference directly on encrypted data. They usually provide a high theoretical privacy guarantee on data leakage, albeit with a significant performance penalty, and hence are rarely used in training DNNs. *Secure multi-party computing (MPC)* is another approach, where multiple servers may use custom data exchange protocols to protect input data. They mostly use secret sharing schemes and have super-linear overhead as the number of sharers and colluding entities grow. An entirely orthogonal approach is to use *differential privacy (DP)*, which protects individual users' information through probabilistic guarantees by inserting noise signals to some parts of the computation. The tradeoff between utility and privacy is a challenge in this line of work. TEEs attracted attention recently for their privacy and integrity properties Asvadishirehjini et al. (2020); Mo et al. (2020); Ng et al. (2019). Among TEE-based approaches, Tramer & Boneh (2018) introduced Slalom an *inference* framework that uses TEE-GPU collaboration to protect data privacy and integrity. However, as stated in their work their model was not designed for training DNNs. *Instance Hiding* is a recently introduced method Huang et al. (2020). In this work authors combined multiple images from a private dataset, merge them with a public image set, and using a sign flip function on pixels as random noise parameters. This method processes the encoded data without any decoding. However, privacy guarantees are not theoretically guaranteed, and in Carlini et al. (2020) authors designed an attack to break the system. In Table 1, we compare some of these approaches based on their privacy and integrity guarantees, and their applications.

3 DarKnight

System Structure: Our system model for learning is shown in Figure 1. We show K' GPU accelerators that participate in linear computations ($\text{GPU}_1, \text{GPU}_{K'}$) on data that is encoded in the TEE. In this work we use Intel SGX as our TEE.

Threat Model: The threat model on the server-side is a dynamic malicious adversary. Whenever GPUs receive data from TEE, they may use known techniques to extract information about the original data or inject faults in the computation. Moreover, a subset of *colluding GPUs* may try to extract information by collaborating with each other or inject faults to sabotage the training. In a system with K' accelerator GPUs, DarKnight provides:

Data Privacy: DarKnight provides perfect privacy with IEEE single-precision arithmetic. In Floating-Point (FP) arithmetic, perfect privacy at a given precision is when the information leakage between encoded data and raw data is less than the round off error. Namely, $I(X : X') < FP.precision$, where I is the mutual information (Cover, 1999; Guo et al., 2020).

Integrity: DarKnight is (K'-1)-secure, namely it can detect any malicious computation even if K'-1 GPUs send erroneous results to TEE.

Collusion Tolerance: DarKnight provide perfect privacy **and** integrity when M GPUs collude, where M is a function of K' and the number of inputs that can be encoded, as described later.

3.1 DarKnight Flow

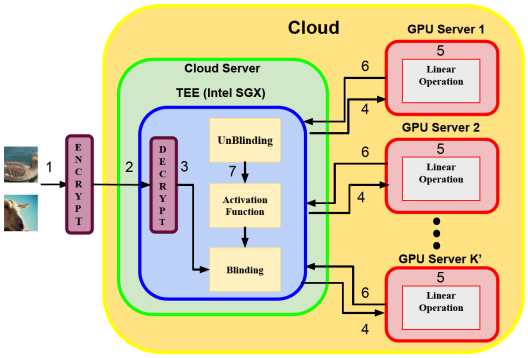

Figure 1: Forward/backward pass of DarKnight

The initial model ($\mathbf{W}$) that a user wants to train is loaded into the cloud server and is made accessible to the untrusted GPUs as well. DarKnight then uses the following steps: (1) A batch of training/inference input data set is encrypted by the client using mutually agreed keys with TEE and sent to the server. (2) TEE decrypts the images and starts the encoding process. (3) During the forward/backward pass of training, each layer requires linear and nonlinear operations. The linear operations are compute-intensive and will be offloaded to GPUs. DarKnight's encoding mechanism is used to seal the data before sending the data to GPU accelerators. To seal the data, DarKnight uses the notion of a *virtual batch*, where K inputs and a random noise are linearly combined to form $K+1$ coded inputs. The size of the virtual batch is limited by the size of the TEE memory that is necessary to encode K images, typically 4-8 images at a time. (4) The encoded data is offloaded to GPUs for linear operation. Each GPU receives at most one encoded data (5) GPUs perform linear operations on different encoded data sets and return the results to TEE in step (6). The TEE decodes the received computational outputs using DarKnight's decoding strategy and then performs any non-linear operations within the TEE in step (7). This process is repeated both for forward pass and backward propagation of each layer. *In a system with K' GPUs and virtual batch size K, DarKnight can provide data privacy and computational integrity by tolerating up to M colluding GPUs, where $K+M+1 \leq K'$.*

4 Privacy in Training

for simplicity, we first show how this mechanism works for a system in which GPUs are not colluding and next we expand the encoding to support a system with M colluding GPUs in Appendix B. For a model with L layers which is being trained with a batch of K inputs, the model parameters $\mathbf{W}_l$ at layer l are updated using the well known SGD process as:

$$\mathbf{W}_l^{\text{new}} = \mathbf{W}_l^{\text{old}} - \eta \times \nabla \mathbf{W}_l, \quad \nabla \mathbf{W}_l = \frac{1}{K} \sum_{i=1}^{K} \langle \delta_l^{(i)}, \mathbf{x}_l^{(i)} \rangle \tag{1}$$

Here $x_l^{(i)}$ is the i^{th} input of layer l. η is the learning rate, and $\delta_l^{(i)}$ is the gradient of the loss for the i^{th} point in the training batch, with respect to the output of layer l.

4.1 Forward Pass

At a layer l the forward pass, we need to compute $\mathbf{y}_l = \langle \mathbf{W}_l \ , \ \mathbf{x}_l \rangle$, where $\langle \cdot, \cdot \rangle$ corresponds to the bilinear operation at that layer (e.g. matrix product, convolution, etc.). After the linear operation finishes, an activation function ($g(\cdot)$) creates the next layer input $\mathbf{x}_{l+1} = \mathrm{g}(\mathbf{y}_l)$. Within this context, DarKnight first receives a set of K inputs $\mathbf{x}_0^{(1)}, \ldots, \mathbf{x}_0^{(K)}$ for a batch training from a client. Our goal is to perform linear calculations of $\mathbf{y}_0^{(1)} = \langle \mathbf{W}_0, \mathbf{x}_0^{(1)} \rangle, \ldots, \mathbf{y}_0^{(K)} = \langle \mathbf{W}_0, \mathbf{x}_0^{(K)} \rangle$ on the GPUs without exposing the inputs to the GPU. Note that the subscript 0 in all these variables refers to the first layer. At this point, we drop the subscript for a more clear notation. Also, we apply $\mathbf{x}$ for the inputs that need to be protected and $\bar{\mathbf{x}}$ for the encoded inputs to visually distinguish different notations. DarKnight must protect $\mathbf{x}_l^{(i)}$ for each layer of the DNN when the layer's linear operations are outsourced to GPUs.

Key Insight: The main idea behind DarKnight's privacy protection scheme is the fact that the most computationally intensive operator (such as convolutions) is *bilinear*. Thus, instead of asking a GPU

to calculate $\langle \mathbf{W}, \mathbf{x}^{(i)} \rangle$, which exposes the inputs, DarKnight uses matrix masking to linearly combine the inputs and add a random noise to them. Due to the bilinear property, any linear operation on K masked inputs can be recovered if there are K different linear computations performed.

DarKnight Encoding: Using a customized version of matrix masking (Cox, 1980; 1994; Kim, 1986; Spruill, 1983; Yu et al., 2019), The SGX based enclave within the cloud server first receives a set of inputs from a data holder. Then the DarKnight scheme creates $K+1$ encoding within the SGX from K data inputs $(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)})$, as follows,

$$\bar{\mathbf{x}}^{(i)} \quad = \quad \alpha_{i,1}\mathbf{x}^{(1)} + \cdots + \alpha_{i,K}\mathbf{x}^{(K)} + \alpha_{i,(K+1)}\mathbf{r} \tag{2}$$

Where $i = 1, \ldots, (K+1)$. The scalars $\alpha_{i,j}$, and the noise vector $\mathbf{r}$ are randomly generated; and the size of $\mathbf{r}$ matches that of $\mathbf{x}$. The scalars $\alpha_{i,j}$'s are represented by matrix $\mathbf{A} \in \mathbb{R}^{(K+1),(K+1)}$, which are dynamically generated for each virtual batch and securely stored inside SGX for decoding. As we prove later, by revealing the values $\bar{\mathbf{x}}^{(i)}$'s to GPUs, we protect the privacy of inputs $\mathbf{x}^{(i)}$'s. At the next step, the encoded data $\bar{\mathbf{x}}^{(i)}$'s are sent to the GPUs which performs the following computations: $\bar{\mathbf{y}}^{(i)} = \langle \mathbf{W}, \bar{\mathbf{x}}^{(i)} \rangle, \quad i = 1, \ldots, (K+1)$. Please note that each GPU only receives one encoded data. Note-worthily matrix $\mathbf{A}$ can be chosen such that its condition number close to one, so that encoding and decoding algorithm remains numerically stable. Hence, orthogonal matrices serve us the best.

DarKnight Decoding: The $K+1$ outputs $\bar{\mathbf{y}}^{(i)}$ returned from the GPUs must be decoded within the SGX to extract the original results $\mathbf{y}^{(i)}$. These value can be extracted as follows,

$$\bar{\mathbf{Y}} = \left\langle \mathbf{W}, [\bar{\mathbf{x}}^{(1)}, \ldots, \bar{\mathbf{x}}^{(K+1)}] \right\rangle = \underbrace{\left\langle \mathbf{W}, [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)}, \mathbf{r}] \right\rangle}_{\mathbf{Y}} \cdot \mathbf{A} \;\Rightarrow\; \mathbf{Y} = \bar{\mathbf{Y}} \cdot \mathbf{A}^{-1} \tag{3}$$

4.2 Backward Propagation

The decoding process for forward pass exploited the invariant property of model parameter for any given input such that $\langle \mathbf{W}, [\bar{\mathbf{x}}^{(1)}, \ldots, \bar{\mathbf{x}}^{(k+1)}] \rangle = \langle \mathbf{W}, [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}, \mathbf{r}] \rangle \cdot \mathbf{A}$, meaning that a single $\mathbf{W}$ was shared between all the inputs of that layers. However, during the backward propagation process, we a have different $\delta_l^{(i)}$ for each input $\mathbf{x_l^{(i)}}$. Thus, decoding the $\langle \delta_l^{(i)}, \mathbf{x_l^{(i)}} \rangle$ from obfuscated inputs $\langle \delta_l^{(i)}, \bar{\mathbf{x}}_l^{(i)} \rangle$ is a more challenging approach that requires specific decoding approach.

Key Insight: While backward propagation operates on a batch of inputs, it is not necessary to compute the $\langle \delta_l^{(i)}, \mathbf{x}_l^{(i)} \rangle$ for each input $\mathbf{x}^{(i)}$. Instead, the training process only needs to compute cumulative parameter updates for the entire batch of inputs. Hence, what is necessary to compute is the entire $\nabla \mathbf{W}_l$ which is an average over all updates corresponding to inputs in the batch.

DarKnight Encoding: DarKnight exploits this insight to protect privacy without significantly increasing the encoding and decoding complexity of the blinding process. As shown in Equation equation 1, there are K inputs on which gradients are computed. DarKnight calculates the overall weight update in the backward propagation by summing up the following $K+1$ equations each of which are computed on a different GPUs,

$$\nabla \mathbf{W} = \sum_{j=1}^{K+1} \gamma_j \mathrm{Eq}_j, \qquad \mathrm{Eq}_j = \left\langle \sum_{i=1}^{K} \beta_{j,i}\, \delta^{(i)}\, , \bar{\mathbf{x}}^{(j)} \right\rangle \tag{4}$$

In the above equations,the encoded input $\bar{\mathbf{x}}^{(j)}$ to a layer is the same that was already calculated during the forward pass using Equation equation 2. Hence, the TEE can simply reuse the forward pass encoding without having to re-compute. The gradients are multiplied with the $\beta_{j,i}$ in the GPUs after which the GPUs compute the bi-linear operation to compute Eq_j.

In contrast to inference where $\mathbf{W}$'s are fixed for all the inputs, during training the parameter updates are with respect to a specific input. Hence, each $\delta_l^{(i)}$'s corresponds to different $\mathbf{x}_l^{(i)}$ during training. As such, DarKnight uses a different encoding strategy where the overall parameter updates $\nabla \mathbf{W}$ can

be decoded very efficiently. In particular, DarKnight selects $\alpha_{j,i}$'s, $\beta_{j,i}$'s and γ_i's such that

$$\mathbf{B}^{\intercal} \cdot \mathbf{\Gamma} \cdot \mathbf{A} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix}_{K \times (K+1)} \tag{5}$$

Assuming batch size is equal to K, the $\beta_{i,j}$ parameters used for scaling δ values is gathered in the $K+1$ by K matrix, $\mathbf{B}$. $\alpha_{i,j}$'s are gathered in the $K+1$ by $K+1$ matrix $\mathbf{A}$, the scalar matrix with the same size for intermediate features and γ_i's form the diagonal of a $K+1$ by $K+1$ matrix Γ, that gives us the proper parameters for efficient decoding. Note that the SGX keeps matrix Γ and $\mathbf{A}$ as secret. We provide the details of privacy guarantee in Appendix A.

DarKnight Decoding: Given the constraint imposed on $\alpha_{j,i}$'s, $\beta_{j,i}$'s and γ_i's the decoding process is trivially simple to extract $\nabla \mathbf{W}$. It is easy to see that if the scalars $\alpha_{i,j}$'s, $\beta_{i,j}$'s and γ_i's satisfy the relation equation 5, the decoding process only involves calculating a linear combination of the values in Equation equation 4.

$$\frac{1}{K} \sum_{j=1}^{K+1} \gamma_j \, \mathrm{Eq}_j = \frac{1}{K} \sum_{i=1}^{K} \langle \delta_l^{(i)}, \mathbf{x}_l^{(i)} \rangle = \nabla \mathbf{W}_l \tag{6}$$

Computational Integrity: DarKnight's encoding scheme can be extended to detect computational integrity violations by untrusted GPUs. To provide integrity, DarKnight creates one additional linear combination of inputs (say $\bar{\mathbf{x}}^{(K+2)}$), using the same approach as in Equation equation 2. This additional equation allows us to verify the accuracy of each result $\mathbf{y}^{(i)}$ by computing it redundantly.

5 Experiments

DarKnight's training scheme and the related unique coding requirements are implemented as an SGX enclave thread on an Intel Coffee Lake server. We used three different DNN models: VGG16 (Simonyan & Zisserman, 2014), ResNet152 (He et al., 2016) and, MobileNetV2 (Sandler et al., 2018) and ImageNet (Russakovsky et al., 2015) as our dataset.

Training Execution Time: Figure 2 demonstrates the speedup of training using DarKnight relative to the baseline fully implemented on SGX with $K = 2$ images encoded and offloaded to 3 GPUs. The results break down the execution time spent into linear (GPU operations and communication time with GPU) and non-linear (all other operations) categories. The results show that DarKnight speeds up the total linear operation time of VGG16 by 23x by using the vast GPUs parallelism. The baseline has to encryption/decrypt data that do not fit within the SGX memory, such as some of the large intermediate feature maps in training. Hence non-linear operations observe $1.89X$ speedup in DarKnight. Overall the execution time is improved by more than $8X$ with DarKnight. Both ResNet and MobileNet models have batch normalization layers that are computation-intensive and cannot be offload to GPU accelerators. Even in this worst-case scenario, performance gains of $4.2X$ and $2.2X$ are achieved. More results are provided in Appendix C.

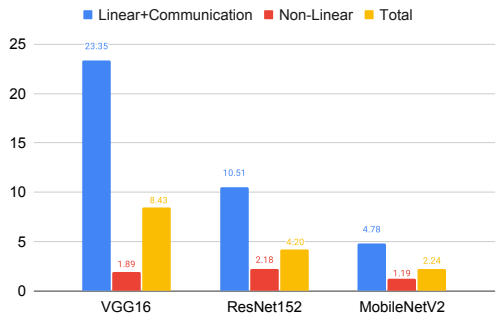

Figure 2: Training Speedup over Baseline

REFERENCES

Amazon. *Machine Learning on AWS*, 2020. URL https://aws.amazon.com/machine-learning.

Aref Asvadishirehjini, Murat Kantarcioglu, and Bradley Malin. Goat: Gpu outsourcing of deep learning training with asynchronous probabilistic integrity verification inside trusted execution environment. *arXiv preprint arXiv:2010.08855*, 2020.

Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Shuang Song, Abhradeep Thakurta, and Florian Tramer. An attack on instahide: Is private learning possible with instance encoding? *arXiv preprint arXiv:2011.05315*, 2020.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Lawrence H Cox. Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, 75(370):377–385, 1980.

LH Cox. Matrix masking methods for disclosure limitation in microdata. *Surv. Methodol.*, 20: 165–169, 1994.

Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *Proceedings on Privacy Enhancing Technologies*, 2017(4):345–364, 2017.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pp. 201–210, 2016.

Google. *Google AI platform*, 2020. URL https://cloud.google.com/products/ai.

Chuan Guo, Awni Hannun, Brian Knott, Laurens van der Maaten, Mark Tygert, and Ruiyu Zhu. Secure multiparty computations in floating-point arithmetic. *arXiv preprint arXiv:2001.03192*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In *International Conference on Machine Learning*, pp. 4507–4518. PMLR, 2020.

Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961*, 2018.

Nick Hynes, Raymond Cheng, and Dawn Song. Efficient deep learning on multi-source private data. *arXiv preprint arXiv:1807.06689*, 2018.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 1651–1669, 2018.

Jay J Kim. A method for limiting disclosure in microdata based on random noise and transformation. In *Proceedings of the section on survey research methods*, pp. 303–308. American Statistical Association Alexandria, VA, 1986.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2009.

Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 619–631, 2017.

Gregory J Matthews, Ofer Harel, et al. Data confidentiality: A review of methods for statistical disclosure limitation and methods for assessing privacy. *Statistics Surveys*, 5:1–29, 2011.

Microsoft. *Azure Machine Learning*, 2020. URL `https://azure.microsoft.com/en-us/services/machine-learning`.

Fatemehsadat Mireshghallah, Mohammadkazem Taram, Prakash Ramrakhyani, Ali Jalali, Dean Tullsen, and Hadi Esmaeilzadeh. Shredder: Learning noise distributions to protect inference privacy. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 3–18, 2020.

Fatemehsadat Mirshghallah, Mohammadkazem Taram, Praneeth Vepakomma, Abhishek Singh, Ramesh Raskar, and Hadi Esmaeilzadeh. Privacy in deep learning: A survey. *arXiv preprint arXiv:2004.12254*, 2020.

Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. Darknetz: towards model privacy at the edge using trusted execution environments. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pp. 161–174, 2020.

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

Krishna Giri Narra, Zhifeng Lin, Yongqin Wang, Keshav Balasubramaniam, and Murali Annavaram. Privacy-preserving inference in machine learning services using trusted execution environments. *arXiv preprint arXiv:1912.03485*, 2019.

Lucien KL Ng, Sherman SM Chow, Anna PY Woo, Donald PH Wong, and Yongjun Zhao. Goten: Gpu-outsourcing trusted execution of neural network training and prediction. 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Jinhyun So, Basak Guler, A Salman Avestimehr, and Payman Mohassel. Codedprivateml: A fast and privacy-preserving framework for distributed machine learning. *arXiv preprint arXiv:1902.00641*, 2019.

Nancy Spruill. The confidentiality and analytic usefulness of masked business microdata. *Proceedings of the Section on Survey Research Methods, 1983*, pp. 602–607, 1983.

Florian Tramer and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *arXiv preprint arXiv:1806.03287*, 2018.

Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019(3):26–49, 2019.

Qian Yu, Songze Li, Netanel Raviv, Seyed Mohammadreza Mousavi Kalan, Mahdi Soltanolkotabi, and Salman A Avestimehr. Lagrange coded computing: Optimal design for resiliency, security, and privacy. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1215–1225. PMLR, 2019.

A PRIVACY GUARANTEE

Darknight provides privacy by matrix masking. Masking keeps all the variables in their floating-point representations while adding Gaussian noise (or uniform noise) to the vector we would like to protect.

The information leaked with masking indicates how much information the masked vector possesses about the raw data (Guo et al., 2020; Matthews et al., 2011). In the other words, it represents the amount of information the adversary can potentially gain from the raw data, *without* any assumption or limitation on adversaries power.

We will first explain a general matrix masking introduced by (Cox, 1980; 1994). Next, we will explain Darknight privacy, through the notation used in matrix masking. Finally, we will calculate the information leakage in our masked matrix, as a measure of privacy.

Matrix Masking:
Introduced by (Cox, 1980; 1994), matrix masking scheme can be used for a variety of reasons such as noise addition, sampling, etc. The general form of $BXA + C$ is used for protecting Matrix X. In the above formula B, A, and C are called record transformation masks, attribute transformation masks, and displacing masks, respectively. Any of these matrices can be used for encoding data based on the data privacy goal. For instance, (Kim, 1986) first added random noise to data and then transformed it to form a distribution with the desired expected value and variance, by carefully tuning A and B. (Spruill, 1983) empirically compared different masking schemes including additive and multiplicative noise. Darknight encoding is a form of matrix masking, with the right choice of the matrices A, B, and C. A combination of Matrix Masking and coded computing first introduced in Yu et al. (2019), for secure and robust computation.

DarKnight Encoding:
Following our notation in equation 2, our goal is to protect the vectors $\mathbf{x}_i$, by adding a random noise to each as follows

$$\begin{aligned} \bar{\mathbf{x}}^{(i)} \quad &= \quad \alpha_{i,1}\mathbf{x}^{(1)} + \cdots + \alpha_{i,K}\mathbf{x}^{(K)} + \alpha_{i,(K+1)}\mathbf{r}\ , \\ & i = 1, \ldots, (K+1)\ , \end{aligned} \tag{7}$$

where $\mathbf{r}$ is a random noise vector, and $\alpha_{i,j}$'s are also chosen randomly. Now first, we denote $\mathbf{X} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)}]$ to be the matrix that we would like to protect, and $\bar{\mathbf{X}} = [\bar{\mathbf{x}}^{(1)}, \ldots, \bar{\mathbf{x}}^{(K)}]$ to be the masked matrix that we send to unsecured GPU. In this case, the equation equation 7 can be rewritten as follows.

$$\bar{\mathbf{X}} = \mathbf{X} \cdot A_1 + \mathbf{r} \cdot \mathbf{a}_2^T \tag{8}$$

where the matrix $A = [\alpha_{i,j}]_{i,j} \in \mathbb{R}^{(K+1)\times(K+1)}$ contains some values of $\alpha_{i,j}$'s, and $\mathbf{a}_2^T = [\alpha_{1,(K+1)}, \ldots, \alpha_{(K+1),(K+1)}]$.

We also prefer to choose a matrix $\mathbf{A}_1$, with a condition number close to one, so that our encoding and decoding algorithm remains numerically stable. For this purpose, orthogonal matrices serve us the best. In addition to that, the transformation of the matrix whose entities are independent and identically distributed standard normal variants is invariant under orthogonal transformations. Therefore, if an orthogonal matrix is used for encoding, the distribution of the raw data and encoded data remains the same (Kim, 1986), which is preferable in data privacy.

Privacy Guarantee:
In this section, we bound the information that leaks, when using Darknight's masking approach. The amount of information leaked by $\bar{\mathbf{x}}^{(i)}$'s about $\mathbf{x}^{(j)}$ is the **mutual information** between these two variables (Cover, 1999). In this setting, each GPU can observe *at most one* encoded data, hence the mutual information is defined by

$$I(\mathbf{x}^{(j)}; \bar{\mathbf{x}}^{(i)}) = h(\mathbf{x}^{(j)}) - h(\mathbf{x}^{(j)}|\bar{\mathbf{x}}^{(i)}) \qquad j = 1, \ldots K\ . \tag{9}$$

Here, $h(\cdot)$ denotes the Shannon entropy function. Note that the information that adversary can potentially learn about $\mathbf{x}^j$ by having $\bar{\mathbf{x}}^i$ is fundamentally bounded by $I(\mathbf{x}^{(j)}; \bar{\mathbf{x}}^{(i)})$. Next, we will rigorously bound this information leakage and show how it can be bounded by properties of the noise.

Theorem 1. *Assume that $X^1, \ldots, X^K$ are scalars such that $|X^i| \leq C_1$ for all i. Suppose $\alpha_{i,j}$'s are real non-zero scalars and R denotes a Gaussian random variable with variance σ^2. Also $\bar{X}$ is defined as*

$$\bar{X} = \sum_{j=1}^{K} \alpha_j X^j + \alpha_{(K+1)} R \,. \tag{10}$$

Then the information leaked from $\bar{X}$ about X^j is bounded by

$$I\left(X^j; \bar{X}\right) \leq \frac{K C_1^2 \bar{\alpha}^2}{2\underline{\alpha}^2 \sigma^2} \,, \quad j = 1, \ldots, K \,. \tag{11}$$

Here $\bar{\alpha} = \max_{i,j} |\alpha_{i,j}|$ and $\underline{\alpha} = \min_{i,j} |\alpha_{i,j}|$.

Proof. Since $\alpha_{i,j}$'s are non-zero, we have

$$\begin{aligned}
I\left(X^j; \bar{X}\right) &= I\left(\alpha_j X^j; \bar{X}\right) \\
&\overset{(1)}{=} I\left(\alpha_j X^j; \sum_{l=1}^{K} \alpha_l X^l + \alpha_{(K+1)} R\right) \\
&\overset{(2)}{=} H\left(\sum_{l=1}^{K} \alpha_l X^l + \alpha_{(K+1)} R\right) \\
&\quad - H\left(\sum_{\substack{l=1 \\ l \neq j}}^{K} \alpha_l X^l + \alpha_{(K+1)} R\right) \\
&\overset{(3)}{\leq} H\left(\sum_{l=1}^{K} \alpha_l X^l + \alpha_{(K+1)} R\right) - H\left(\alpha_{(K+1)} R\right) \\
&= I\left(\sum_{l=1}^{K} \alpha_l X^l; \sum_{l=1}^{K} \alpha_l X^l + \alpha_{(K+1)} R\right) \,.
\end{aligned} \tag{12}$$

Here, for equality (1), we simply replace $\bar{X}^i$ with its definition. (2) is due to the definition of the mutual information ($I(X; X+Y) = H(X+Y) - H(Y)$). Finally, inequality (3) holds due to Lemma 1.
Now, note that since $|X^l| \leq C_1$, we have

$$\text{Var}\left(\sum_{l=1}^{K} \alpha_l X^l\right) = \sum_{l=1}^{K} \text{Var}\left(\alpha_l X^l\right) \leq K \bar{\alpha}^2 C_1^2 \tag{13}$$

Also $\alpha_{(K+1)} R$ is a zero-mean Gaussian random variable with variance $\alpha_{(K+1)}^2 \sigma^2$. Therefore, using Lemma 2, we have

$$\begin{aligned}
&I\left(\sum_{l=1}^{K} \alpha_l X^l; \sum_{l=1}^{K} \alpha_l X^l + \alpha_{(K+1)} R\right) \qquad \leq \\
&\frac{\text{Var}\left(\sum_{l=1}^{K} \alpha_l X^l\right)}{2\alpha_{(K+1)}^2 \sigma^2} \leq \frac{K C_1^2 \bar{\alpha}^2}{2\underline{\alpha}^2 \sigma^2}
\end{aligned} \tag{14}$$

Finally, using equation 12, equation 14, we conclude that

$$I\left(X^j; \bar{X}\right) \leq \frac{K C_1^2 \bar{\alpha}^2}{2\underline{\alpha}^2 \sigma^2} \tag{15}$$

□

Lemma 1. *Suppose that X and Y are two independent random variables. Then we have,*

$$\max\{H(X), H(Y)\} \leq H(X+Y)\,. \tag{16}$$

Proof. Since X and Y are independent, we have $H(X+Y|X) = H(Y|X)$ and $H(Y|X) = H(Y)$. Therefore,

$$H(X+Y) \geq H(X+Y|X) = H(Y|X) = H(Y)\,. \tag{17}$$

The same argument shows that $H(X+Y) \geq H(X)$, which concludes the proof. □

Lemma 2. *Assume that $X_i \sim P_{X_i}$ is a random variable, and $R_i \sim \mathcal{N}(0, \sigma_i^2)$ is a Gaussian random variable with variance σ^2 and mean 0. Also, assume that X_is and R_is are independent. Then we have,*

$$\begin{aligned} &I(X^1, X^2, ..., X^n; X^1 + R^1, X^2 + R^2, ..., X^k + R^n) \\ &\leq \sum_{i=1}^{N} \frac{1}{2} \log\left(1 + \frac{Var(X^i)}{\sigma_i^2}\right) \leq \sum_{i=1}^{N} \frac{Var(X^i)}{2\sigma_i^2}, \end{aligned} \tag{18}$$

where $Var(X^i)$ is variance of the random variable X^i.

Please refer to section 9.4 of Cover (1999) for the detailed proof of Lemma equation 2.

Theorem 1 shows that by increasing the power of the noise, one can arbitrarily reduce the leaked information. Please note that for deep learning applications normalization is common in the prepossessing phase. Furthermore, many of the networks such as MobileNet and ResNet variants take advantage of the batch normalization layers. Hence, the value of C_1 in the above theorem is bound by $N^{(\frac{-1}{2})}$ in case ℓ_2 normalization is used (which obviously implies $C_1 \leq 1$). With a batch size of K = 2, setting variance of the noise, $\mathbf{r}$, to be $\sigma^2 = 4e^8$, and limiting $\frac{\bar{\alpha}^2}{\alpha^2} < 10$, we have the upper bound of $5e^{-8}$ on the leaked information, Because our amount of leakage is less thank the precision loss(round off error) in IEEE single-precision arithmetic, we achieve perfect privacy; meaning that the amount of data leakage is less than the accuracy loss due to round off error (Guo et al., 2020).

B COLLUDING GPUS

In this section, we investigate the scenario in which multiple GPUs can collaborate to extract information from the encoded data. With K' GPUs and virtual batch size of K, we can tolerate $M < K' - K$ colluding GPUs without compromising privacy. We show how we can securely outsource calculating $\langle \mathbf{W}, \mathbf{x}^{(i)} \rangle$, $i = 1, \ldots, K$, to the GPUs. We first create $P = M + K$ encoded data vectors, $\bar{\mathbf{x}}^i$, $i = 1, \ldots, P$, using M noise vectors $\mathbf{R}^1, \ldots, \mathbf{R}^M$, as follows.

$$\begin{aligned} &\bar{\mathbf{X}} = \mathbf{X}\mathbf{A}_1 + \mathbf{R}\mathbf{A}_2\,, \quad \text{where}\,, \\ &\bar{\mathbf{X}} = \left[\bar{\mathbf{x}}^1, \ldots, \bar{\mathbf{x}}^P\right] \in \mathbb{R}^{N \times P}\,, \\ &\mathbf{X} = \left[\mathbf{x}^1, \ldots, \mathbf{x}^K\right] \in \mathbb{R}^{N \times K}\,, \\ &\mathbf{R} = \left[\mathbf{R}^1, \ldots, \mathbf{R}^M\right] \in \mathbb{R}^{N \times M}\,, \\ &\text{and}\,,\ \mathbf{A}_1 \in \mathbb{R}^{K \times P}\,,\ \mathbf{A}_2 \in \mathbb{R}^{M \times P}\,. \end{aligned} \tag{19}$$

Here, the matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are the encoding coefficient similar to the initial scheme we used for DarKnight. Theorem 2 provides privacy guarantees for this approach under very mild conditions on the matrix $\mathbf{A}_2$.

Theorem 2. *In the encoding scheme described above, assume that the encoding matrix $\mathbf{A}_2$ is a full-rank matrix, such that for every column $\mathbf{A}_2^{(i)}$ in $\mathbf{A}_2$, we have $\|\mathbf{A}_2^{(i)}\|_2 \geq C$. Also assume that the vectors $\mathbf{R}^i$ are independently drawn from $\mathcal{N}(\mathbf{0}, \sigma^2\mathbb{I})$.Then the maximum leaked information with M colluding GPUs is bounded by*

$$\sum_{i,j} \frac{Var(X^{i,j})}{C\sigma^2} \tag{20}$$

Proof. Assume that a subset $S \subseteq [1, \dots K']$ of the K' GPUs are colluding and $|S| = M$. Thus, those GPUs have are given the encoded vectors $\{\bar{\mathbf{x}}^i\}^{i\in S}$. Our goal is to bound the mutual information between $\{\bar{\mathbf{x}}^i\}^{i\in S}$ and $\mathbf{X}$.

$$I\left(\mathbf{X}\ ;\ \mathbf{X}\mathbf{A}_1(:,S) + \mathbf{R}\mathbf{A}_2(:,S)\right)\ . \tag{21}$$

Here, for a matrix M, $M(:,S)$ denotes a sub-matrix of M, whose columns are chosen from the set S. Note that the matrix $\mathbf{A}_2(:,S)$ is full-rank, whose norm of each column is lower-bounded by C. Therefore,

$$\begin{aligned} &I\left(\mathbf{X}\ ;\ \mathbf{X}\mathbf{A}_1(:,S) + \mathbf{R}\mathbf{A}_2(:,S)\right) \\ &\qquad \leq I\left(\mathbf{X}\ ;\ \mathbf{X}\mathbf{A}_1(:,S) + C\sigma^2\bar{\mathbf{R}}\right)\ , \end{aligned} \tag{22}$$

where $\bar{\mathbf{R}}$ is a matrix with iid standard Gaussian entries. This is because for a Gaussian matrix $\mathbf{M}$ and a vector $\mathbf{v}$, we have $\mathbf{M}\mathbf{v} \sim \mathbf{g}\|\mathbf{v}\|$, where $\mathbf{g}$ is a Gaussian vector. Now, simply using Lemma 2 yields

$$\begin{aligned} &I\left(\mathbf{X}\ ;\ \mathbf{X}\mathbf{A}_1(:,S) + \mathbf{R}\mathbf{A}_2(:,S)\right) \\ &\qquad \leq I\left(\mathbf{X}\ ;\ \mathbf{X}\mathbf{A}_1(:,S) + C\sigma^2\bar{\mathbf{R}}\right) \\ &\qquad \leq \sum_{i,j} \frac{\text{Var}(X^{i,j})}{C\sigma^2}\ , \end{aligned} \tag{23}$$

and this concludes the proof. □

As you saw in the proof, we needed every sub-matrix $\mathbf{A}_2(:,S) \in \mathbb{R}^{M\times|S|}$ has linearly independent columns. That is why it was necessary to have at most M colluding GPUs ($|S| \leq M$) when we use M noise vectors in our scheme. In the other words, when using M noise vectors (which required M extra equations/GPUS), we can tolerate at most M colluding GPUs.

Now that we took care of inference as described above, we would like to update our training procedure for this new scenario. Same as before, we can calculate the weight updates using the following equations:

$$\nabla\mathbf{W} = \sum_{j=1}^{P} \gamma_j \text{Eq}_j, \qquad \text{Eq}_j = \left\langle \sum_{i=1}^{K} \beta_{j,i}\ \delta^{(i)}\ , \bar{\mathbf{x}}^{(j)} \right\rangle \tag{24}$$

We now define

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}\ , \mathbf{B} = [\beta_{j,i}]\ , \Gamma = \text{Diag}(\gamma_1, \dots, \gamma_K) \tag{25}$$

Now, it is easy to show that if

$$\mathbf{B}^\intercal \cdot \mathbf{\Gamma} \cdot \mathbf{A} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \ddots & \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}_{K\times K'} \tag{26}$$

C Experimental Setup and Results

DarKnight server consisted of an Intel Coffee Lake E-2174G 3.80GHz processor and Nvidia GeForce GTX 1080 Ti GPUs. The server has 64 GB RAM and supports Intel Soft Guard Extensions (SGX). DarKnight's training scheme and the related unique coding requirements are implemented as an SGX enclave thread where both the decoding and encoding are performed. For SGX implementations, we used Intel Deep Neural Network Library (DNNL) for designing the DNN layers including the Convolution layer, ReLU, MaxPooling, and Eigen library for Dense layer. We used Keras 2.1.5, Tenseflow 1.8.0, and Python 3.6.8.

We used three different DNN models: VGG16 (Simonyan & Zisserman, 2014), ResNet152 (He et al., 2016) and, MobileNetV2 (Sandler et al., 2018). We chose MobileNetV2 because it is the worst-case benchmark for our model as it reduces linear operations considerably (using depth-wise

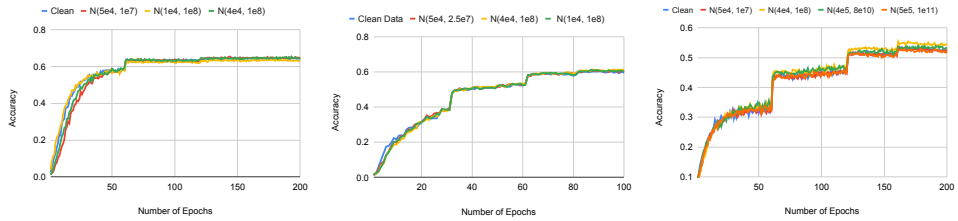

Figure 3: Training accuracy of DarKnight for CIFAR-100 with (a) VGG16 (b) ResNet152 (c) MobileNetV2

separable convolution), thereby reducing the need for GPU acceleration. We used ImageNet (Russakovsky et al., 2015), CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) as our datasets. All the parameters, models' and implementation details, and dataset descriptions are attached in the supplementary material.

C.1 Training Results

For evaluating training performance, three aspects are examined: accuracy impact, speed up of training, and maximum information leakage.

Effect of Random Noise on Accuracy: Adding large noise to inputs to encode the data may cause floating-point rounding errors on GPUs. To study the impact, Fig. 3 shows the training accuracy when using different noise strengths on VGG16, ResNet152, and MobileNetV2. We use a random Gaussian vector with iid entries, $\mathcal{N}(\mu, \sigma^2)$, as the noise vectors $\mathbf{r}_i$'s, where σ^2 is the order of magnitude strength over the typical input and feature map values seen in a model. For instance, $\mathcal{N}(5e^4, 1e^7)$ means the noise is drawn from a distribution with the mean at $5e^4$ and variance of $1e^7$. Figure 3 (a) shows the accuracy of training for VGG16 on CIFAR-100 dataset. Even with a powerful noise signal ($\sigma^2 = e^8$), the accuracy loss after epoch 50 is less than 0.001 compared to training on open data without any privacy controls. Very similar behavior is observed across a wide range of input datasets and models.

Table 2: Effect of different noise signals on the accuracy of DarKnight inference for different models on ImageNet

	VGG16		ResNet152		MobileNetV1		All Models
Noise	Top1 Accuracy	Top5 Accuracy	Top1 Accuracy	Top5 Accuracy	Top1 Accuracy	Top5 Accuracy	**MI upper bound**
No privacy	64.26	85.01	72.93	90.60	64.96	85.29	–
$\mathcal{N}(4e3, 1.6e7)$	64.23	85.01	72.46	90.47	64.99	85.26	$1.25 * 10^{-6}$
$\mathcal{N}(1e4, 2.5e7)$	64.25	85.06	72.35	90.23	64.81	85.26	$0.8 * 10^{-6}$
$\mathcal{N}(1e4, 1e8)$	64.25	85.05	71.87	89.93	64.54	85.15	$2 * 10^{-7}$
$\mathcal{N}(0, 4e8)$	64.24	85.01	72.24	90.09	64.87	85.19	$\mathbf{5 * 10^{-8}}$

Information Leakage and Mutual Information: Table 2 show accuracy impact of various noise strengths, on the inference accuracy. For noise strengths that have 7 orders of magnitude higher variance than the input signal, negligible accuracy losses were observed. When the noise strength reaches 8 orders of magnitude ResNet152 seems a worst-case Top1 accuracy drop of about 1%. The last column represents the upper bound of mutual information computed from Theorem 1. By limiting $\frac{\bar{\alpha}^2}{\alpha^2} < 10$ for $K = 2$ when using $\mathcal{N}(0, 4e8)$, we have 5×10^{-8} upper bound on the information leakage which is less than the roundoff error in IEEE single-precision arithmetic and hence, perfect privacy is achieved with this precision.