ACCELERATING RELU FOR MPC-BASED PRIVATE INFERENCE WITH A COMMUNICATION-EFFICIENT SIGN ESTIMATION

Kiwan Maeng [1] **G. Edward Suh** [2 3]

ABSTRACT

Secure multi-party computation (MPC) allows users to offload machine learning inference on untrusted servers without having to share their privacy-sensitive data. Despite their strong security properties, MPC-based private inference has not been widely adopted due to their high communication overhead, mostly incurred when evaluating non-linear layers. This paper presents HummingBird, an MPC framework that reduces the ReLU communication overhead significantly. HummingBird leverages an insight that determining whether a value is positive or negative mostly does not need full-bit communication. With its theoretical analyses and an efficient search engine, HummingBird discards 66–72% of the bits during ReLU without altering the outcome, and discards 87–91% when some accuracy can be degraded. On a realistic MPC setup, HummingBird achieves on average 2.03–2.67× end-to-end speedup without introducing any errors, and up to 8.42× when some accuracy degradation is tolerated.

1 INTRODUCTION

Machine learning (ML) inference often uses privacy-sensitive user data. A model that predicts disease by looking at an X-ray image (Ho et al., 2022) uses the patients' private X-ray data. Smart home devices that take in the user's verbal command (Amazon, 2023; Home, 2023; Meta, 2023a) collect raw microphone inputs that can contain sensitive information. As models powering these services become larger and are often proprietary, an increasing trend is to host these models on a remote server owned by the service provider, to which the users send their input data. This emerging trend creates a dilemma for the users — to use these services, the users have to send their privacy-sensitive input data to a third party, risking potential privacy leakage.

Secure *multi-party computation* (MPC; Goldreich (1998)) is gaining wide interest as a potential solution to this dilemma. MPC allows users to offload ML inference to untrusted servers without having to reveal their private data to the servers (Liu et al., 2017; Mishra et al., 2020; Juvekar et al., 2018; Demmler et al., 2015; Mohassel & Rindal, 2018; Knott et al., 2021; Kumar et al., 2020; Rathee et al., 2020; Huang et al., 2022c; Jha et al., 2021; Cho et al., 2022b). In MPC, instead of sending their raw data, users send *secret shares* of their data, from which the servers cannot infer the users' raw data. The servers run inference using the secret shares and send the result back to the user, who can retrieve the output of the inference from the results. Figure 2 summarizes MPC-based private inference.

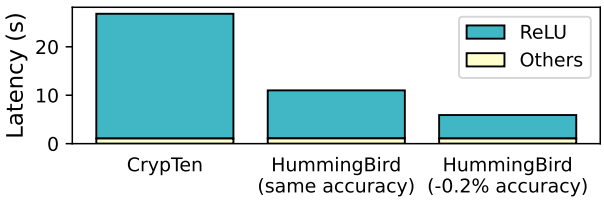

Figure 1. Compared to CrypTen (Knott et al., 2021), HummingBird achieves 2.24× speedup without *any error* and 4.41× when some accuracy can be degraded. Details can be found in Section 5.

Despite their strong security guarantees, MPC-based private inference has not been widely adopted in the real world yet due to their high runtime overheads. Even the most efficient schemes (Knott et al., 2021; Huang et al., 2022c) experience multiple orders of magnitude slowdown over a non-private baseline. Unlike non-private inference that are computation- or memory-bound, the majority of the overhead in MPC comes from communications between parties during non-linear operations, *e.g.,* ReLU. In a particular setup we studied, ReLU was accountable for over 93% of the total overhead (Figure 1, leftmost bar), which is in line with observations from prior works (Jha et al., 2021). Recent works have designed a faster algorithm for ReLU (Demmler et al., 2015; Mohassel & Rindal, 2018; Rathee et al., 2020; Wagh et al., 2019; 2021) or model architectures that use

*Equal contribution [1]Pennsylvania State University [2]FAIR, Meta [3]Cornell University. Correspondence to: Kiwan Maeng <kvm6242@psu.edu>.

Proceedings of the 5th MLSys Conference, Santa Clara, CA, USA, 2024.

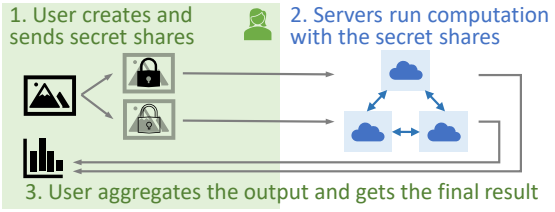

Figure 2. Overview of a multi-server MPC protocol.

fewer ReLUs (Jha et al., 2021; Cho et al., 2022a;b; Jha & Reagen, 2023; Kundu et al., 2023) to tackle the overhead.

We explore an orthogonal approach that accelerates existing MPC ReLU algorithms further. The key insight is that simply *guessing the sign*, unlike high-precision arithmetic, can be done *exactly correctly* using only a small subset of the bits. We theoretically show that discarding certain high-order bits of a secret share during ReLU can still maintain the computation *mathematically equivalent* to that of the original computation, while significantly reducing the communication. We also show that discarding low-order bits is similar to performing magnitude-based activation pruning, which is known to have little effect on accuracy (Kurtz et al., 2020; Oh et al., 2021; Haberer & Landsiedel, 2022; Li et al., 2023b; Gupta et al., 2021). We can reduce the communication even further using this second insight, when a moderate accuracy degradation is tolerated.

Based on the theoretical insights, we propose HummingBird, a framework that automatically selects proper bits to discard for each ReLU layer and uses an optimized kernel to translate the reduced bits into speedup. Compared to the popular CrypTen framework (Knott et al., 2021) on a typical LAN setup, HummingBird achieves 1.81–3.04× end-to-end speedup while *keeping the computation mathematically equivalent*, and 3.74–7.28× speedup when some accuracy degradation is tolerated. The idea of HummingBird can be applied to many other popular MPC frameworks immediately. Below summarizes our contributions:

1. We theoretically show that discarding certain high-order bits of the secret shares during a ReLU evaluation does not alter the outcome for a large family of MPC protocols. HummingBird leverages the fact to improve the performance without altering the result.
2. We show that discarding low-order bits during ReLU renders the result similar to activation pruning. HummingBird leverages the fact to further improve the performance when small accuracy degradation is allowed.
3. We propose a search algorithm to decide how many high- and low-order bits to remove for each ReLU layer given a budget, and present an efficient search engine that performs the search on a lightweight simulator. Within a reasonable amount of time (several minutes to an hour), HummingBird finds a configuration that minimally impacts the model accuracy while significantly improving the communication overhead.
4. We implemented a runtime library as an extension to CrypTen that can bring up to 8.42× average end-to-end speedup and 8.76× communication reduction with the configuration found by the search engine. We will open-source the codebase upon paper publication.

2 Background and Motivation

2.1 Private Inference with Multi-party Computation

Existing works on MPC-based inference can be broadly classified into either a *client-server* setup or a *multi-server* setup. Client-server MPC (Liu et al., 2017; Juvekar et al., 2018; Mohassel & Zhang, 2017; Mishra et al., 2020; Huang et al., 2022c; Rathee et al., 2020; Chandran et al., 2019) assumes a client holding data and a server holding a model. In these protocols, the server runs most of the linear computations using a mixture of homomorphic encryption (HE) and MPC protocols. Non-linear layers are collaboratively evaluated (Garimella et al., 2023). These protocols provide strong security as the client need not worry about collusion.

Multi-server MPC (Tan et al., 2021; Mohassel & Rindal, 2018; Kumar et al., 2020; Wagh et al., 2021; 2019; Riazi et al., 2018; Chaudhari et al., 2019; Patra & Suresh, 2020; Byali et al., 2020; Chaudhari et al., 2020; Knott et al., 2021) assumes multiple non-colluding servers collaboratively running an MPC-based inference. While users can also act as one of the parties, it is more common to assume they simply offload the inference to multiple non-colluding servers (Mohassel & Zhang, 2017) by generating and sending secret shares of their inputs (Figure 2). The servers cannot learn about the users' input from the received secret shares unless they collude. The model can be shared between the parties or be private to one of them. If the model is private, other parties use an encrypted model, and the execution is slower compared to when the model is shared.

Multi-server MPC is usually faster than the client-server MPC because it does not involve expensive HE operations (Huang et al. (2022c) observed a 15× difference between the two due to the HE operations). The major downside of a multi-server MPC is that the user data are safe only when the involved parties do not collude (Knott et al., 2021). This non-colluding assumption can be realized with policies and contracts between the parties. Many companies are forming an alliance (Alliance, 2023) to explore and adopt MPC technologies, and some simple form of MPC is already being adopted in the industry (Meta, 2023b).

Evaluation of ReLU This paper focuses on ReLU-heavy networks (*e.g.*, CNN) following recent works (Jha et al., 2021; Cho et al., 2022a;b; Jha & Reagen, 2023; Kundu et al., 2023). ReLU is evaluated in several different ways: some of the popular approaches include the Goldreich-Micali-Wigderson (GMW) protocol (Goldreich et al., 1987), garbled circuit (Yao, 1982), or a variant of SecureNN (Wagh et al., 2019)'s protocol. Among these, the GMW protocol is GPU-friendly and is often used in GPU-based high-throughput systems (Knott et al., 2021; Tan et al., 2021).

Many of the aforementioned protocols evaluate ReLU by first evaluating whether the secret is positive, *i.e.*, $x \geq 0$?, and multiplying the boolean result by the original secret (Goldreich et al., 1987; Wagh et al., 2019). Following prior works (Kumar et al., 2020), we call this sign estimation operator *DReLU* [1]: $\mathrm{DReLU}(x) = 1$ iff $x \geq 0$ and 0 otherwise. With DReLU, ReLU is trivially:

$$\mathrm{ReLU}(x) = x \times \mathrm{DReLU}(x). \tag{1}$$

As we show in Section 2.3, evaluating DReLU consists of most of the overhead of ReLU in these protocols.

Scope of HummingBird We describe and evaluate the idea of HummingBird on top of CrypTen (Knott et al., 2021), a GMW-based multi-server MPC framework developed and maintained by Meta. CrypTen is popular due to its high-speed GPU support and has served as a foundation of several recent works (Tan et al., 2021; Li et al., 2023a; Wang et al., 2022; Zhang et al., 2023). However, our idea is relevant to a wider range of works.

First, the idea is directly applicable to any other protocol that uses Equation 1 for ReLU and experiences a DReLU overhead that increases with the number of bits in the secret share. All the other GMW-based systems (Mohassel & Rindal, 2018; Tan et al., 2021; Patra & Suresh, 2020) and other popular frameworks (Wagh et al., 2019; 2021; Kumar et al., 2020; Rathee et al., 2020) fall into this category.

Second, the same idea can be applied to comparison in general ($x \geq a$? is equivalent to $\mathrm{DReLU}(x - a)$), and can accelerate other comparison-based operators (*e.g.*, max pooling) or non-linear operators that internally uses comparison in their approximation. For example, Pang et al. (2023) uses comparison to approximate Tanh and GELU.

We only consider a setup where two parties participate in the computation ($p = 2$ in the notation from Section 2.2), which is the most efficient (Knott et al., 2021) and common setup (Mohassel & Rindal, 2018; Kumar et al., 2020; Wagh et al., 2021; 2019). However, Theorem 1 is applicable to setups with more parties ($p > 2$). As in the original CrypTen paper, we assume an honest-but-curious adversary.

[1] for derivative of ReLU

2.2 Operation of CrypTen and the GMW Protocol

Notations Let $x \in \mathbb{Z}/Q\mathbb{Z}$ be a secret value in an integer ring of size $Q = 2^N$. We denote p arithmetic secret shares of x as $\langle x \rangle_p^Q \in \mathbb{Z}/Q\mathbb{Z}$, where $\Sigma_{i=0}^{p-1} \langle x \rangle_i^Q \equiv x \pmod{Q}$. We simply denote the set of the shares as $\langle x \rangle^Q = \{\langle x \rangle_p^Q\}$. For x represented in an N-bit signed integer representation (two's complement), we denote p binary secret shares of x as $\langle x \rangle_p^B$, where $\oplus_{i=0}^{p-1} \langle x \rangle_i^B = x$ for a bitwise XOR operation $\oplus$. Throughout the paper, we assume an element in a ring of size 2^n is always in an n-bit signed integer representation for any n. We express bits from the m-th bit to the $k-1$-th bit in x ($m \leq k$) as $x[k:m]$. For example, if $x = 11011101_b$, $x[5:1] = 1110_b$. Note that the k-th bit is excluded. We treat the resulting $x[k:m]$ as an element on a smaller ring $\mathbb{Z}/2^{k-m}\mathbb{Z}$ unless stated otherwise. Similarly, we denote the k-th bit of x as $x[k]$.

Operation of CrypTen Here, we briefly explain how CrypTen works. An in-depth understanding of CrypTen is not necessary for the rest of the paper, and we encourage curious readers to read the original paper for more details.

In CrypTen, users split their secret input $x \in \mathbb{Z}/Q\mathbb{Z}$ into p ($p \geq 2$) arithmetic secret shares and send each share $\langle x \rangle_p^Q$ to participating servers P_p. When $p = 2$, secret shares can be easily generated by the client generating a random number r and making $\langle x \rangle_0^Q = x + r$, $\langle x \rangle_1^Q = -r$. Floating-point values x_f are converted to an integer ring element x by multiplying with a large integer D and rounding ($x = \lfloor D x_f \rceil$). Addition or multiplication by a public value can be trivially done directly on arithmetic secret shares (*e.g.*, $\Sigma_{i=0}^{p-1} a \langle x \rangle_i^Q \equiv ax \pmod{Q}$), allowing efficient linear operations by a public weight. Addition between two secret shares can also be done trivially. Multiplication between secret shares requires communications between the parties and a set of random numbers called the Beaver triplets (Beaver, 1991), which can be generated by a trusted third party (TTP) or by using oblivious transfer (Knott et al., 2021).

Non-linear operations, such as ReLU, are much more expensive. CrypTen evaluates ReLU by first separately evaluating DReLU (Equation 1). When DReLU is applied to a secret share $\langle x \rangle^Q$, the output is a secret share of one ($\langle 1 \rangle^Q$) if $x \geq 0$ and $\langle 0 \rangle^Q$ otherwise. Then, ReLU is evaluated by:

$$\mathrm{ReLU}(\langle x \rangle^Q) = \langle x \rangle^Q \times \mathrm{DReLU}(\langle x \rangle^Q), \tag{2}$$

which requires a multiplication between secret shares and uses the aforementioned Beaver triplets.

CrypTen estimates $\mathrm{DReLU}(\langle x \rangle^Q)$ using the GMW protocol. First, the arithmetic shares $\langle x \rangle^Q$ are converted into binary shares $\langle x \rangle^B$. This arithmetic-to-binary (A2B) conversion is done by each party P_p generating binary shares of their arithmetic shares, $\langle \langle x \rangle_p^Q \rangle^B$, and adding their binary shares $\langle \langle x \rangle^Q \rangle_p^B$ locally using an adder circuit. For an N-bit

secret x, an adder circuit of depth $logN$ is required, with each depth incurring $O(N)$ communication ($\mathbf{O(NlogN)}$ **total communication overhead**). Many other popular protocols experience similar A2B overhead that grows with N (Mohassel & Rindal, 2018; Tan et al., 2021; Patra & Suresh, 2020; Wagh et al., 2019; 2021; Kumar et al., 2020; Rathee et al., 2020). After the conversion, the most significant bit (MSB; sign bit) of $\langle x \rangle^B$ ($\langle x \rangle^B[\mathrm{N}-1]$ if $Q = 2^N$) holds the binary secret share of 0 if x is positive and 1 if negative. Converting $\langle x \rangle^B[\mathrm{N}-1]$ back into arithmetic shares (binary-to-arithmetic; B2A) and subtracting it from $\langle 1 \rangle^Q$ gives us our desired $\mathrm{DReLU}(\langle x \rangle^Q)$ (Knott et al., 2021).

2.3 Detailed Overhead Characterization

How fast can we currently get? Despite recent advancements, there is widespread pessimism that MPC is inherently too slow to be deployed in the real world. To see exactly what the current state-of-the-art looks like, we evaluated a carefully designed setup that maximizes throughput, a metric that is crucial for an industry-scale deployment.

We used CrypTen on two servers ($p = 2$), each with an A100 GPU, connected with a 10 Gbps LAN. The LAN-based setup shows what can be achieved when multiple companies forming an MPC alliance (Alliance, 2023) install their own servers at the same datacenter for a high-performance MPC. We assumed that the model is unencrypted, which makes the linear layers more efficient. The unencrypted model setup represents cases when the parties are running public models, or the parties abide by some confidentiality contract (*e.g.*, NDA) and cannot steal the model. We focused on CNNs with max pooling replaced with average pooling as in recent works (Garimella et al., 2023; 2021a), as they are the closest to becoming practical. Transformers, on the other hand, report a throughput only around 0.02 samples/s (Zhang et al., 2023; Zeng et al., 2023) and require more innovations to become near-practical. Again, HummingBird is not only relevant to the particular setup we studied and can accelerate other setups as well (*e.g.*, max pooling, Transformers, *etc.*), as discussed in Section 2.1.

The leftmost bar of Figure 1 shows the measured overhead breakdown for ResNet18 (He et al., 2016) and CIFAR10 (Krizhevsky et al., 2009) with a batch size of 512. Encouragingly, the numbers are already quite efficient (19.1 samples/s), thanks to various design choices to maximize throughput. However, the overhead is still significant — you can run *millions* of non-private inference per second with the same setup — and the community has a long way to go before this becomes truly practical. See Section 5 for details on the evaluation setup.

Detailed overhead characterization To study how the throughput can be further improved, we measured the major overheads of the aforementioned setup. As the leftmost bar in Figure 1 shows, the first thing we noticed is that 93% of the overhead comes from ReLU layers. This is because we carefully eliminated other major overheads, such as max pooling or encrypted models, but the ReLU overhead could not be easily avoided.

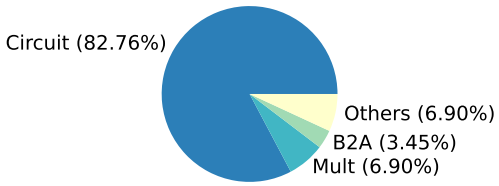

Figure 3. Communication incurred by each part of ReLU.

Figure 3 further breaks down the large overhead of ReLU into different components. **Circuit** refers to the adder circuit explained in Section 2.2 during the A2B conversion (82.76%). **Mult** refers to the multiplication shown in Equation 2 that is done between the secret share and the DReLU output (6.9%). **B2A** refers to the B2A conversion of the 1-bit DReLU output. Unlike the A2B counterpart that performs N-bit to N-bit conversion, B2A converts only one bit (sign bit) and is much cheaper (3.45%). **Others** are AND operations happening inside A2B other than what is captured by **Circuit** (6.9%). Evidently, the vast majority of the communication comes from the adder circuit during the A2B conversion, which is our main target for optimization.

3 Using Less Bits for DReLU

To further improve the performance of CrypTen, we need to optimize the A2B inside DReLU. As explained in Section 2.2, the A2B overhead is $O(NlogN)$, growing with the size of the secret share, N. In MPC protocols, N is usually large ($N = 64$ for CrypTen) to avoid arithmetic wrap-around (Knott et al., 2021) during linear layers.

Our core insight is that DReLU only determines whether the secret (x) is positive or negative, which can often be done *exactly correctly* by only looking at a small subset of the bits. We first show how we can use only a fraction of the bits and still produce *precise results*. Then, we show how we can even cut down more bits, if some *accuracy degradation* is allowed. Using fewer bits immediately improves the $O(NlogN)$ overhead of A2B.

For a high-level insight into our proposed optimization, consider secret shares of $x = 9$, $\langle x \rangle^Q = \{47, -38\}$. Note that adding the two secret shares retrieves the original secret. DReLU can be seen as a process of comparing the absolute values between the positive and the negative secret shares

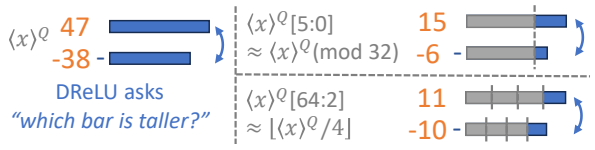

Figure 4. A motivating example of why dropping some high- and low-order bits still gives precise DReLU results.

(Figure 4, left). As it can be seen from the right of Figure 4, the comparison result would not change even when we discard some high-order bits, as it is equivalent to performing a modulo, or throwing away an equal amount from both values and doing the comparison (Figure 4, top right). Similarly, discarding some low-order bits is equivalent to performing a division and a floor operation, or dividing both values with the same value and comparing the quotient (Figure 4, bottom right). Both operations do not change the comparison result if done moderately. From the intuition, we propose to only use $\langle x\rangle^Q[k:m]$ during DReLU:

$$\mathrm{ReLU}(\langle x\rangle^Q) \approx \langle x\rangle^Q \times \mathrm{DReLU}(\langle x\rangle^Q[k:m]). \quad (3)$$

Equation 3 reduces the A2B communication overhead significantly from $O(NlogN)$ into $O((k-m)log(k-m))$. Again, as long as DReLU correctly estimates the sign, Equation 3 is not an approximation but an *exact computation*. Next, we theoretically show when Equation 3 is exact.

3.1 Theoretical Analysis

The insight from Figure 4 was not rigorous and does not always work. Here, we discuss when Equation 3 can be correct. Consider $\langle x\rangle_p^Q \in \mathbb{Z}/Q\mathbb{Z}$, arithmetic secret shares of $x \in \mathbb{Z}/Q\mathbb{Z}$, where $p \in \{0,1\}$. Assume $\langle x\rangle_p^Q$ is represented in an N-bit signed integer representation.

First, we state the condition when discarding the k-th bit and above is still correct. Proof is in Appendix:Section A.4.

Theorem 1. *For $k < N$, $\mathrm{DReLU}(\langle x\rangle^Q)$ is equal to $\mathrm{DReLU}(\langle x\rangle^Q[k:0])$ if $-2^{k-1} \le x < 2^{k-1}$.*

CrypTen (Knott et al., 2021) uses $N = 64$, while a floating point representation x_f is converted into an integer ring element with $x = \lfloor 2^{16} x_f \rceil$. As intermediate activations (x_f) in a DNN are usually close to zero, $x = \lfloor 2^{16} x_f \rceil$ only occupies a small subset of the full range represented by $N = 64$. For the datasets we studied, k between 18–22 was sufficient for $-2^{k-1} \le x < 2^{k-1}$ to always hold. The result indicates that 42–46 high-order bits (accounting for **66–72%**) of the secret shares can be safely discarded without causing **any mathematical errors**. N being much larger than a typical value of x is, in fact, deliberate; it prevents wrap-around errors during arithmetic operations (Mohassel & Zhang, 2017), and simply using a smaller N breaks the entire system. However, DReLU does not cause any wrap-around and does not need to operate with a large N.

Next, we discuss what happens when bits below the m-th bit are discarded. Proof is in Appendix:Section A.4.

Theorem 2. *If each party uses $\langle x\rangle_p^Q[N:m] \in \mathbb{Z}/2^{N-m}\mathbb{Z}$ for DReLU, the ReLU output is mostly equivalent to performing ReLU precisely and zeroing-out values below 2^m, except for rare cases where $x < -2^{N-1} + 2^{m+2}$.*

Theorem 2 shows that, unlike high-order bits, discarding low-order bits is *not safe*. Specifically, non-zero secrets smaller than 2^m becomes zero, and very large negative values ($x < -2^{N-1} + 2^{m+2}$) are flipped to become positive. Thus, if we want to maintain precisely the same computation, we must not discard any low-order bits.

However, Theorem 2 simultaneously shows how we can additionally improve the performance if some amount of accuracy degradation is tolerated. Note that the latter case is unlikely with a large N, and the former only zeros out positive values below 2^m. We note that the behavior is precisely *magnitude-based activation pruning*, which is empirically known to have little impact on the final model accuracy when used in moderation (Kurtz et al., 2020; Oh et al., 2021; Haberer & Landsiedel, 2022; Li et al., 2023b; Gupta et al., 2021). The rest of the ReLU outcomes are *not approximated and exact*. Thus, a careful choice of m is expected to not harm the model accuracy significantly.

Comparison with compression Our optimization is similar in spirit with compression or quantization (Han et al., 2016), in that we aim to reduce the number of bits communicated. However, we emphasize that our optimization *is not compression/quantization*. Traditional compression/quantization aims to make the value of the compressed result close to the original value, *i.e.*, $\mathrm{Compress}(\langle x\rangle_p^Q) \approx \langle x\rangle_p^Q$; however, as $\langle x\rangle_p^Q$ are random values fully occupying the N-bit representation space (very high entropy), it cannot be compressed much. In contrast, our proposed method does not preserve the values of the secret shares at all ($\langle x\rangle_p^Q[k:m] \neq \langle x\rangle_p^Q$), but it instead ensures that the DReLU result is similar before and after the bits are discarded ($\mathrm{DReLU}(\langle x\rangle^Q[k:m]) = \mathrm{DReLU}(\langle x\rangle^Q)$). Moreover, unlike compression and quantization which always induce errors, our optimizations do not introduce any errors (Theorem 1), or only incur errors for specific, predictable range of inputs (Theorem 2).

4 HummingBird System Design

HummingBird is an MPC framework that accelerates ReLUs using the insights from Section 3. HummingBird supports two modes. *HummingBird-eco* is the default mode,

which discards as many bits as possible while keeping the computation mathematically equivalent. HummingBird-eco does not discard any low-order bits and only discards high-order bits while meeting the requirement of Theorem 1. We additionally provide *HummingBird-b*, which trades off performance and accuracy. Given a budget b, HummingBird-b statically searches how many high- and low-order bits to discard for each ReLU layer to maximize accuracy.

HummingBird consists of an offline and an online phase. In the *offline phase*, a search engine finds bits to throw out (*i.e.*, select k, m in $\langle x \rangle^Q[k:m]$) for each ReLU layer to minimize accuracy degradation while maximizing performance. In the *online phase*, HummingBird uses an efficient runtime library that uses the searched parameters and runs DReLU with a subset of bits to achieve an end-to-end speedup. Figure 5 summarizes HummingBird.

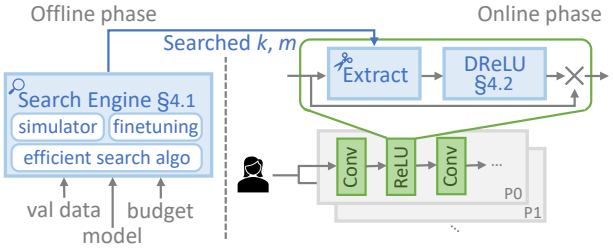

Figure 5. Overview of HummingBird.

4.1 Offline Phase: Finding Bits to Discard

HummingBird's offline search engine consists of three key components: (1) an efficient MPC simulator, (2) a search algorithm, and (3) a model finetuning process.

4.1.1 MPC Simulator

Evaluating any configuration on a real MPC setup during the search process is time-consuming. To save the search time, HummingBird performs the search on an efficient simulator instead. The simulator simply performs a single-node non-private ML inference for all layers except ReLU. Only for ReLU layers, the simulator simulates what HummingBird would do during a real MPC-based inference: converts the floating point values into an integer ring, generates secret shares, discards bits, and calculates DReLU using GMW.

Although the simulator does not simulate what HummingBird does exactly from the beginning to the end, we observed that the final accuracy trend from the simulator matches well with what we observe on a real MPC setup. At the same time, evaluating a configuration on a simulator is much more efficient than running a real MPC, because (1) all the other layers except for ReLU run a vanilla single-node inference and incur no additional MPC-related overhead, and (2) even for the ReLU layers, there is no communication overhead as both the parties are simulated on the same node. The efficient simulator allows the search engine to evaluate numerous configurations within a reasonable time.

4.1.2 Efficient Search Algorithm

The goal of the search algorithm is to find the subset of bits in the secret shares to use for DReLU for each ReLU layer. For HummingBird-eco, the search engine does not throw away any low-order bits as it always introduces errors (Theorem 2). k-th and higher-order bits can be safely thrown away if $-2^{k-1} \leq x < 2^{k-1}$ (Theorem 1). The search engine runs a validation set through each layer, records the minimum and the maximum values of the output, and uses them to select k. The process is very efficient.

HummingBird-b takes in the relative amount of bits to retain as the budget b and finds a configuration that meets the budget while maximizing accuracy. For example, when the search budget is given as $b = 4/64$, it means the total number of bits used in each DReLU combined must be 4/64 or less than the original number of bits combined. The budget can be met by using only 4 bits among the 64-bit secret shares for all ReLU layers, or by retaining different numbers of bits for different layers (*e.g.*, retain only 2 bits for some layers and 8 bits for other layers). Different ReLU layers have different output dimensions — usually for CNNs, earlier ReLU layers have larger dimensions — and discarding bits from the earlier layers reduces the budget more quickly.

The search space for HummingBird-b grows exponentially with the number of ReLU layers and quickly becomes intractable. HummingBird-b enumerates all possible bit assignments to all ReLU layers using a depth-first-search (DFS). With l ReLU layers and N possible bits that can be assigned to each layer, the combinations of possible bit assignments are already $O(N^l)$. To make matters worse, each ReLU layer has to choose k and m that satisfies the number of assigned bits. For example, if one decides to retain 4 bits for all ReLU, each ReLU has to choose k and m from $(k, m) \in \{(4, 0), (5, 1), \ ..., \ (64, 60)\}$. This leads to a combined $O(N^{2l})$ search complexity. To navigate through the search space within a reasonable amount of time, HummingBird uses several optimizations: using locally optimal k and m values, early stopping for unlikely paths, and allowing a coarser search.

First, HummingBird visits each layer and greedily selects a locally optimal k and m values, instead of trying to find a global optimum. When a certain number of bits is assigned to a layer during the DFS, the search engine (1) uses the already-selected k and m for layers that have been visited, and (2) uses $k = N, m = 0$ (*i.e.*, no bit discarded) for layers that haven't been visited yet, and linearly searches for k and

m values for the current layer that yield the best validation accuracy. When the locally optimal values are found, those values are recorded, and the search moves on to the next layer. This greedy heuristic reduces the search complexity from $O(N^{2l})$ to $O(N^l)$.

We further reduce the $O(N^l)$ search space of the DFS by pruning branches that are likely to yield suboptimal configurations. When selecting the locally optimal k and m values for a certain layer, the search engine evaluates the validation accuracy while assuming unexplored layers will not discard any bits (discussed in the previous paragraph) — this validation accuracy serves as an upper bound of the accuracies any configurations following this branch will achieve. If this upper bound is already worse than a predefined threshold or the best configuration found so far, the search engine immediately stops exploring that branch. The search engine also tracks the total number of bits assigned to each layer and immediately stops when the budget is exceeded.

For additional efficiency, the search engine allows grouping multiple ReLUs and making them share the same parameters. For models with a repeating block structure, *e.g.*, ResNet (He et al., 2016), a natural choice is to group the ReLUs within the same block. All these optimizations combined allow our search engine to find a good configuration usually within several minutes, making the search practical.

When zero bit is assigned to a layer, that ReLU layer becomes an identity layer. HummingBird can be seen as a generalization of ReLU culling (Jha et al., 2021) which replaces a ReLU layer with an identity layer for performance.

4.1.3 Model Finetuning

When HummingBird-b degrades accuracy, we go through a model finetuning process to regain some of the accuracy. The finetuning process re-trains the model for a small number of epochs on the simulator, while using the selected k and m for each layer. We found that finetuning was not necessary for $b = 8/64$ and above as the approximation does not degrade the accuracy much; however, finetuning was essential for aggressive budgets below $b = 8/64$, where non-negligible accuracy drops occurred (Section 5.3).

4.2 Online Phase: Efficient DReLU with Less Bits

Using k and m found for each ReLU layer, HummingBird uses Equation 3 during online MPC inference. Note that k and m are selected during the offline phase using the validation data and are fixed during the online phase, not leaking any information about the online user data. With the reduced number of bits, HummingBird speeds up the DReLU process, especially the adder circuit (Section 2.3), with mainly two optimizations. First, it runs a circuit of depth $O(\lceil log(k-m) \rceil)$ instead of $O(logN)$. Second, it efficiently packs and unpacks the subset of bits into a 64-bit tensor before and after each communication to reduce the overhead. While the circuit depth change only impacts the adder circuit overhead (**Circuit**; Section 2.3), the reduced communication due to bitpacking also improves **Mult** and **B2A** from Section 2.3. We implemented HummingBird's online phase on top of CrypTen with Python. The added code accounts for less than 2% of the total execution time.

5 Evaluation Results

In this section, we answer the following questions: (1) How fast is HummingBird in different settings? (2) How much communication is reduced? (3) What are the major overheads of HummingBird? (4) How long is the search time? (5) How important is each component of HummingBird?

5.1 Evaluation Setup

We evaluated HummingBird in several representative setups. The first setup runs two parties on two nodes connected with a 10 Gbps LAN, each with one A100 GPU. The second setup is otherwise identical, but uses a less powerful V100 GPU. The third setup runs two parties on two A100 GPUs on a single node, representing a setup with a very high network bandwidth. We do not model the overhead of generating Beaver triplets, assuming they are generated and stored offline (Garimella et al., 2023) or sent by a trusted third party (TTP) asynchronously. Unlike in a client-server MPC setup where the clients have limited storage, we assume the servers have enough storage to hold pre-generated triplets.

Following prior works (Garimella et al., 2023; 2021a; Cho et al., 2022b), we evaluated HummingBird with ResNet18 and ResNet50 (He et al., 2016), models that are popular in MPC literature. Models like MobileNet (Sandler et al., 2018) have components not suitable for MPC (*e.g.*, ReLU6) and are not commonly used. We evaluated with CIFAR10 (Krizhevsky et al., 2009), CIFAR100 (Krizhevsky et al., 2009), and TinyImageNet (Stanford, 2023) datasets. All the models were trained to match the accuracy reported in prior works (Cho et al., 2022b), and details of the models are summarized in Appendix:Section A.1. For the search engine, we used a validation set of 1024 samples and grouped ReLUs following the five layer groups of ResNet. We used the search budget of 8/64 and 6/64.

5.2 HummingBird Performance Analysis

End-to-end performance improvement Figure 6 and 7 show the speedup of HummingBird over the baseline CrypTen in A100 and V100 GPUs. Both figures show that HummingBird improves the end-to-end performance significantly. Without adding any errors (HummingBird-eco), HummingBird improves the average performance

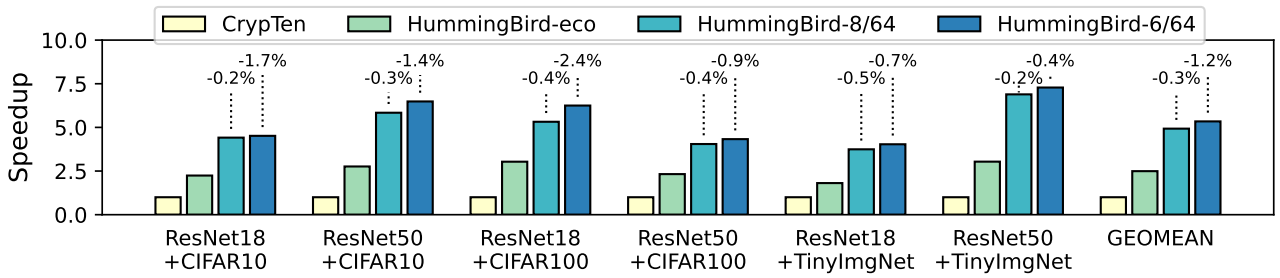

Figure 6. On A100 GPUs, HummingBird improves the end-to-end performance by 1.81–3.04× (HummingBird-eco), 3.74–6.89× (HummingBird-8/64), and 4.03–7.28× (HummingBird-6/64) over CrypTen. Any accuracy degradation is shown above the bar.

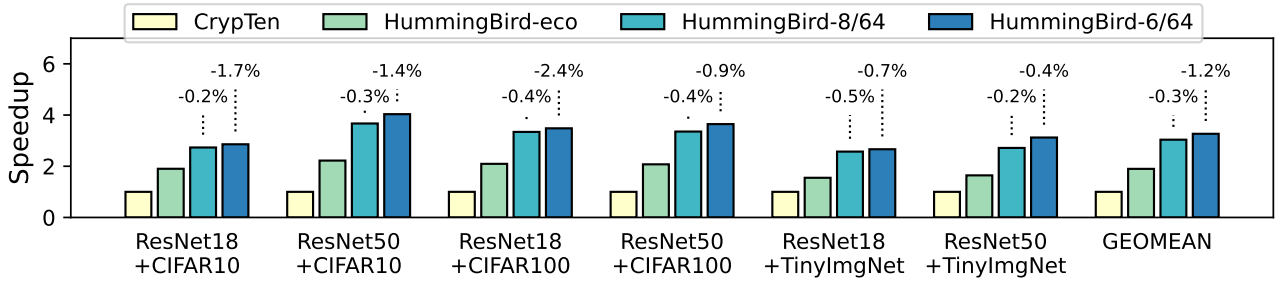

Figure 7. On V100 GPUs, HummingBird improves the end-to-end performance by 1.55–2.22× (HummingBird-eco), 2.57–3.67× (HummingBird-8/64), and 2.66–4.03× (HummingBird-6/64) over CrypTen. Any accuracy degradation is shown above the bar.

by 2.49× and 1.90× on A100 and V100, respectively. When some accuracy degradation is tolerated, the average performance improvement becomes 4.93× and 3.04× (-0.3%; HummingBird-8/64), and 5.34× and 3.26× (-1.2%; HummingBird-6/64), for A100 and V100. The performance improvement is less on the less powerful V100 GPUs because the linear layer computation (*e.g.*, convolution), which HummingBird does not accelerate, is slower on V100.

Performance improvement on different networks Figure 8 shows the average speedup across all the models/benchmarks for different network setups. High-BW represents an ideal setup with very high bandwidth. It is measured on two GPUs on a single node, connected with up to 16 Tbps link (NVIDIA, 2023). LAN reports a setup where two nodes are connected with a 10 Gbps LAN. WAN reports an analytical projection assuming a 320 Mbps bandwidth and a 70 ms round-trip latency, following Huang et al. (2022b). For WAN, we separately measured the communication round and time from the High-BW setup and scaled it according to the assumed bandwidth and latency.

Figure 8 shows that, as expected, HummingBird's performance benefit becomes more notable with limited networks. Compared to the 2.49–5.34× speedup of LAN, High-BW setup enjoyed less speedup of 2.03–4.12×, while the WAN setup enjoyed more speedup of 2.64–8.42×. High-BW and

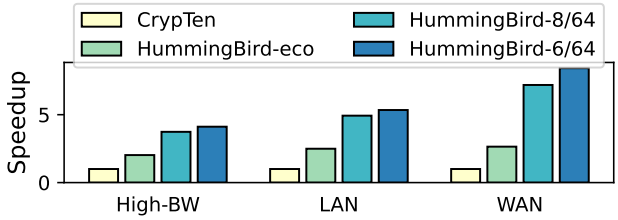

Figure 8. Speedup of HummingBird on different network setups. The bar shows the geometric mean across all the benchmarks. On WAN, HummingBird's speedup reaches 2.64–8.42×.

the LAN setup did not show significant differences although their bandwidth differed by multiple orders, because HummingBird was not able to fully utilize the 16 Tbps bandwidth anyways, using only around 20 Gbps.

Communication Figure 9 shows the total bytes communicated (bar plot) and the number of communication rounds (line plot). On average, HummingBird reduces the number of communication rounds by 1.12–1.56×, and reduces the total bytes communicated by 2.68–8.76×. Communication does not decrease proportionally with less budgets and starts to saturate, because there are communications that cannot be reduced by HummingBird (*e.g.*, **Mult** from Figure 3).

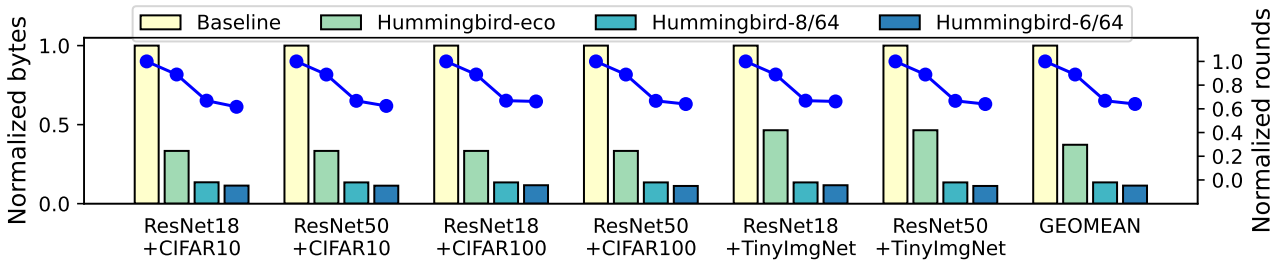

Figure 9. Normalized bytes that need to be communicated (bar) and the number of communication rounds (line). HummingBird reduces the number of communication rounds by 1.12–1.56× and total communicated bytes by 2.68–8.76×.

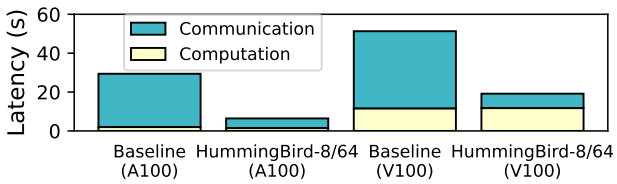

Figure 10. Overhead breakdown of CrypTen and Hummingbird-8/64. HummingBird reduces communication to a degree where the computation is no longer negligible.

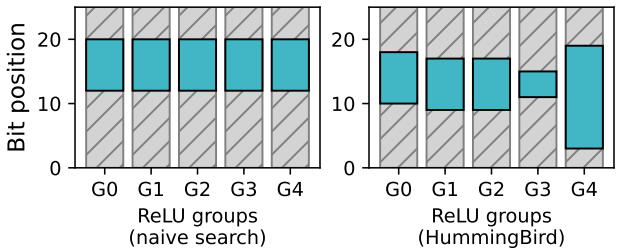

Figure 11. Retained (green) and discarded (grey hatched) bits for each ReLU group using different search strategies. HummingBird's search engine chooses different numbers and positions of bits for different ReLU groups.

Overhead breakdown Figure 10 shows the overhead breakdown of CrypTen and HummingBird-8/64, both on A100 and V100 GPUs. The breakdown shows that HummingBird reduces the communication overhead to a point where the computation overhead becomes non-negligible. With HummingBird-8/64, the portion of the communication overhead decreased from 93% to 78% (A100) and 78% to 39% (V100), respectively. For high-performance GPUs like A100, the major bottleneck is still communication (78%); however, for less-powerful V100 GPUs, HummingBird shifts the bottleneck to computation. The result shows why HummingBird's speedup is larger for A100. In V100, the computation overhead, which HummingBird does not accelerate, becomes non-negligible (61%).

Search overhead In most cases, HummingBird was able to find a satisfactory configuration in a few minutes. For large models (TinyImageNet with ResNet50), the search time sometimes reached an hour. The search time can be further reduced by using a smaller validation set or a coarser ReLU group. See Appendix:Table 2 for more details.

5.3 Ablation Studies

Effectiveness of the search engine HummingBird's search engine finds bits to discard (*i.e.*, k, m) per each ReLU group. A simple alternative approach would be to use the same k and m for all the ReLU layers. We found that such a naive alternative does not work well, incurring more than an 8% accuracy drop for the same search budget. Figure 11 visualizes the bits that are discarded (gray hatched) or retained (green) among the 64 bits for the two approaches with $b = 8/64$. Unlike the naive approach that discards the same bits for all the ReLU layers (Figure 11, left), HummingBird flexibly chooses to discard different amounts of bits for different layers (Figure 11, right), sometimes discarding more (G3) and sometimes less (G4). As different ReLU layers have different importance and characteristics, the search engine is crucial for achieving high accuracy.

Effectiveness of finetuning While finetuning was not necessary when the search budget was reasonably large (*e.g.*, HummingBird-8/64) as the accuracy degradation was already small, we found finetuning to be crucial when the search budget was small (*e.g.*, HummingBird-6/64), where non-negligible accuracy degradation occurred. Finetuning improves the model accuracy by 0.95–7.05% depending on the dataset and the model. See Appendix:Table 3 for details.

6 Additional Related Works

6.1 Alternative Approaches to Private Inference

Trusted execution environment (TEE) TEEs (Intel, 2021; ARM Ltd., 2021; AMD, 2023; Lee et al., 2020; Suh

et al., 2007; Lie et al., 2003; Lee et al., 2005; Bourgeat et al., 2019; Champagne & Lee, 2010; Chhabra et al., 2011; Costan et al., 2016; Evtyushkin et al., 2014; Fletcher et al., 2012; McKeen et al., 2013; Szefer & Lee, 2012; Lie et al., 2000; Watson et al., 2015; Yang et al., 2003) are hardware that ensures the authenticity of the software running on it, and the confidentiality and integrity of code and data used inside. Following the initial proposal from academia (Suh et al., 2007; Lie et al., 2003; Lee et al., 2005), most major vendors have TEEs in their commercial products (Intel, 2021; ARM Ltd., 2021; AMD, 2023; Lee et al., 2020; NVIDIA, 2022). TEEs for emerging heterogeneous accelerators are also being actively proposed (Jang et al., 2019a; Jiang et al., 2022; Zhao et al., 2022; Kang et al., 2021; Lee et al., 2022; Volos et al., 2018; Jang et al., 2019b; Zhu et al., 2020; Zuo et al., 2020; Hua et al., 2020). TEEs are efficient because they eliminate the need for any expensive HE or MPC operations, and are widely available in commodity off-the-shelf hardware. However, the security assurance from a TEE is generally considered to be weaker than cryptographic protection from HE/MPC. Although data inside a TEE should ideally be secure, TEE implementations may be vulnerable due to hardware/software bugs (Liu et al., 2021; Khandaker et al., 2020) or side channels (Wang et al., 2017).

Fully homomorphic encryption (FHE) FHE is a cryptographic technique that allows certain computations directly on an encrypted ciphertext. Using FHE, servers can collect user data in a ciphertext form and run computation (*e.g.*, DNN inference) directly on the ciphertext (Gentry, 2009). While the first HE schemes and systems were very slow, subsequent works accelerated HE-based private inference heavily on CPUs (EPFL-LDS, 2021), GPUs (Jung et al., 2021; Kim et al., 2023), FPGAs (Roy et al., 2019; Riazi et al., 2020; Agrawal et al., 2023), and custom accelerators (Samardzic et al., 2021; Kim et al., 2022b;a). Unlike MPC, FHE cannot evaluate ReLU precisely and must approximate it with a high-degree polynomial (Kim et al., 2022b). While recent advances in algorithms and hardware accelerators significantly reduced the latency of FHE, the throughput is still limited: using CIFAR10 and ResNet20, recent studies report a throughput of 8 samples/s on a custom accelerator (Kim et al., 2022a) and 0.7 samples/s on an A100 GPU (Kim et al., 2023), which are orders of magnitude less than what HummingBird achieves.

Instance encoding Instance encoding (Carlini et al., 2020) refers to a general concept where the client encodes the input into a noisy encoding, such that reconstructing the original input is statistically hard while some useful downstream inference or training is still possible. Similar concepts have been explored under many different names across communities, including split inference (Kang et al., 2017; Vepakomma et al., 2021; Titcombe et al., 2021; He et al., 2020; Mireshghallah et al., 2020; 2021), split learning (Vepakomma et al., 2020; 2018; Poirot et al., 2019), vertical federated learning (vFL; Thapa et al. (2022); Yang et al. (2019); Li et al. (2022)), learnable encryption (Yala et al., 2021; Xiao & Devadas, 2021; Xiang et al., 2020), private image publication (Fan, 2018; 2019), *etc.* Instance encoding is usually efficient computation-wise, as no cryptographically-heavy operation is needed. However, these approaches lack a strong theoretical guarantee on their claimed privacy-enhancing properties (Carlini et al., 2020; 2021), unlike MPC or FHE which are cryptographically secure. A few recent studies provided a theoretical analysis of instance encoding, using tools like (metric) differential privacy (Fan, 2018; 2019), Fisher information leakage (Maeng et al., 2022; 2023), or PAC theory (Xiao & Devadas, 2022). These theories' guarantees are still much weaker compared to MPC. For example, although instance encoding can make input reconstruction statistically more difficult, it still leaks a non-trivial amount of information about the input.

6.2 Additional Related Works on MPC

Section 2.1 summarizes popular client-server and multi-server MPC protocols. Many of these simultaneously introduce orthogonal approaches to accelerate ReLU, which are often complementary to ours. Some of the popular approaches include replacing ReLU with an identity function (Jha et al., 2021; Cho et al., 2022b), replacing ReLU with a polynomial (Park et al., 2022; Mishra et al., 2020; Lou et al., 2021; Gilad-Bachrach et al., 2016; Garimella et al., 2021b), and using a neural architecture search to find a model with fewer ReLUs (Jha et al., 2021; Jha & Reagen, 2023; Cho et al., 2022a). As most of these works were not able to fully replace all the ReLUs, HummingBird can still accelerate their remaining ReLUs. Other works focused on applying MPC to more complex models other than CNNs, including RNNs (Rathee et al., 2021), (vision) Transformers (Li et al., 2023a; Wang et al., 2022; Zhang et al., 2023; Zeng et al., 2023; Pang et al., 2023) and recommendation models (Lam et al., 2023). As these works still uses ReLU (Li et al., 2023a; Wang et al., 2022; Zhang et al., 2023; Zeng et al., 2023) and other comparisons internally (Rathee et al., 2021; Pang et al., 2023), HummingBird is still relevant. Zhou et al. (2022) and Huang et al. (2022a) explore accelerating MPC with the help of customized TEEs.

7 Conclusion

MPC-based private inference is very slow, due to the significant communication overhead it incurs. After applying best-effort optimizations, the majority ($> 93\%$) of the remaining overhead comes from ReLUs. In this work, we theoretically show that most of the bits in the secret shares can be removed during ReLU evaluation with little to no

impact on accuracy for many popular protocols. Leveraging the finding, we propose HummingBird, an efficient MPC framework that uses a reduced number of bits during ReLU evaluation. HummingBird carefully selects the bits to retain for each layer and uses an efficient runtime library, reducing the communication overhead by up to 8.76× and achieving up to 8.42× end-to-end speedup over CrypTen.

ACKNOWLEDGEMENTS

We thank the reviewers of MLSys for their feedback. We also thank Brandon Reagan and his team at NYU for the insightful discussions.

REFERENCES

Agrawal, R., de Castro, L., Yang, G., Juvekar, C., Yazicigil, R. T., Chandrakasan, A. P., Vaikuntanathan, V., and Joshi, A. FAB: an fpga-based accelerator for bootstrappable fully homomorphic encryption. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2023, Montreal, QC, Canada, February 25 - March 1, 2023*, pp. 882–895. IEEE, 2023. doi: 10.1109/HPCA56546.2023.10070953. URL https://doi.org/10.1109/HPCA56546.2023.10070953.

Alliance, M. Powering secure computation, together. https://www.mpcalliance.org/, 2023.

Amazon. Amazon echo & alexa devices. https://www.amazon.com/smart-home-devices/b?ie=UTF8&node=9818047011, 2023.

AMD. AMD Secure Encrypted Virtualization (SEV). https://www.amd.com/en/developer/sev.html, 2023.

ARM Ltd. TrustZone for cortex-m. https://www.arm.com/why-arm/technologies/trustzone-for-cortex-m, 2021.

Beaver, D. Efficient multiparty protocols using circuit randomization. In Feigenbaum, J. (ed.), *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pp. 420–432. Springer, 1991. doi: 10.1007/3-540-46766-1_34. URL https://doi.org/10.1007/3-540-46766-1_34.

Bourgeat, T., Lebedev, I. A., Wright, A., Zhang, S., Arvind, and Devadas, S. MI6: secure enclaves in a speculative out-of-order processor. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2019, Columbus, OH, USA, October 12-16, 2019*, pp. 42–56. ACM, 2019. doi: 10.1145/3352460.3358310. URL https://doi.org/10.1145/3352460.3358310.

Byali, M., Chaudhari, H., Patra, A., and Suresh, A. FLASH: fast and robust framework for privacy-preserving machine learning. *Proc. Priv. Enhancing Technol.*, 2020(2):459–480, 2020. doi: 10.2478/popets-2020-0036. URL https://doi.org/10.2478/popets-2020-0036.

Carlini, N., Deng, S., Garg, S., Jha, S., Mahloujifar, S., Mahmoody, M., Song, S., Thakurta, A., and Tramèr, F. An attack on instahide: Is private learning possible with instance encoding? *CoRR*, abs/2011.05315, 2020. URL https://arxiv.org/abs/2011.05315.

Carlini, N., Garg, S., Jha, S., Mahloujifar, S., Mahmoody, M., and Tramèr, F. Neuracrypt is not private. *CoRR*, abs/2108.07256, 2021. URL https://arxiv.org/abs/2108.07256.

Champagne, D. and Lee, R. B. Scalable architectural support for trusted software. In Jacob, M. T., Das, C. R., and Bose, P. (eds.), *16th International Conference on High-Performance Computer Architecture (HPCA-16 2010), 9-14 January 2010, Bangalore, India*, pp. 1–12. IEEE Computer Society, 2010. doi: 10.1109/HPCA.2010.5416657. URL https://doi.org/10.1109/HPCA.2010.5416657.

Chandran, N., Gupta, D., Rastogi, A., Sharma, R., and Tripathi, S. Ezpc: Programmable and efficient secure two-party computation for machine learning. In *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*, pp. 496–511. IEEE, 2019. doi: 10.1109/EuroSP.2019.00043. URL https://doi.org/10.1109/EuroSP.2019.00043.

Chaudhari, H., Choudhury, A., Patra, A., and Suresh, A. ASTRA: high throughput 3pc over rings with application to secure prediction. In Sion, R. and Papamanthou, C. (eds.), *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop, CCSW@CCS 2019, London, UK, November 11, 2019*, pp. 81–92. ACM, 2019. doi: 10.1145/3338466.3358922. URL https://doi.org/10.1145/3338466.3358922.

Chaudhari, H., Rachuri, R., and Suresh, A. Trident: Efficient 4pc framework for privacy preserving machine learning. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. URL https://www.ndss-symposium.org/ndss-paper/trident-efficient-4pc-framework-for-privacy-preserving-machine-learning/.

Chhabra, S., Rogers, B., Solihin, Y., and Prvulovic, M. Secureme: a hardware-software approach to full system security. In Lowenthal, D. K., de Supinski, B. R., and McKee, S. A. (eds.), *Proceedings of the 25th International Conference on Supercomputing, 2011, Tucson, AZ, USA, May 31 - June 04, 2011*, pp. 108–119. ACM, 2011. doi: 10.1145/1995896.1995914. URL https://doi.org/10.1145/1995896.1995914.

Cho, M., Ghodsi, Z., Reagen, B., Garg, S., and Hegde, C. Sphynx: A deep neural network design for private inference. *IEEE Secur. Priv.*, 20(5):22–34, 2022a. doi: 10.1109/MSEC.2022.3165475. URL https://doi.org/10.1109/MSEC.2022.3165475.

Cho, M., Joshi, A., Reagen, B., Garg, S., and Hegde, C. Selective network linearization for efficient private inference. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 3947–3961. PMLR, 2022b. URL https://proceedings.mlr.press/v162/cho22a.html.

Costan, V., Lebedev, I. A., and Devadas, S. Sanctum: Minimal hardware extensions for strong software isolation. In Holz, T. and Savage, S. (eds.), *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pp. 857–874. USENIX Association, 2016. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan.

Demmler, D., Schneider, T., and Zohner, M. ABY - A framework for efficient mixed-protocol secure two-party computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*. The Internet Society, 2015. URL https://www.ndss-symposium.org/ndss2015/aby---framework-efficient-mixed-protocol-secure-two-party-computation.

EPFL-LDS. Lattigo v2.3.0. https://github.com/ldsec/lattigo, 2021.

Evtyushkin, D., Elwell, J., Ozsoy, M., Ponomarev, D. V., Abu-Ghazaleh, N. B., and Riley, R. Iso-x: A flexible architecture for hardware-managed isolated execution. In *47th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2014, Cambridge, United Kingdom, December 13-17, 2014*, pp. 190–202. IEEE Computer Society, 2014. doi: 10.1109/MICRO.2014.25. URL https://doi.org/10.1109/MICRO.2014.25.

Fan, L. Image pixelization with differential privacy. In Kerschbaum, F. and Paraboschi, S. (eds.), *Data and Applications Security and Privacy XXXII - 32nd Annual IFIP WG 11.3 Conference, DBSec 2018, Bergamo, Italy, July 16-18, 2018, Proceedings*, volume 10980 of *Lecture Notes in Computer Science*, pp. 148–162. Springer, 2018. doi: 10.1007/978-3-319-95729-6_10. URL https://doi.org/10.1007/978-3-319-95729-6_10.

Fan, L. Differential privacy for image publication. In *Theory and Practice of Differential Privacy (TPDP) Workshop*, volume 1, pp. 6, 2019.

Fletcher, C. W., Dijk, M. v., and Devadas, S. A secure processor architecture for encrypted computation on untrusted programs. In *Proceedings of the seventh ACM workshop on Scalable trusted computing*, pp. 3–8, 2012.

Garimella, K., Jha, N. K., Ghodsi, Z., Garg, S., and Reagen, B. Cryptonite: Revealing the pitfalls of end-to-end private inference at scale. *CoRR*, abs/2111.02583, 2021a. URL https://arxiv.org/abs/2111.02583.

Garimella, K., Jha, N. K., and Reagen, B. Sisyphus: A cautionary tale of using low-degree polynomial activations in privacy-preserving deep learning. *CoRR*, abs/2107.12342, 2021b. URL https://arxiv.org/abs/2107.12342.

Garimella, K., Ghodsi, Z., Jha, N. K., Garg, S., and Reagen, B. Characterizing and optimizing end-to-end systems for private inference. In Aamodt, T. M., Jerger, N. D. E., and Swift, M. M. (eds.), *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, ASPLOS 2023, Vancouver, BC, Canada, March 25-29, 2023*, pp. 89–104. ACM, 2023. doi: 10.1145/3582016.3582065. URL https://doi.org/10.1145/3582016.3582065.

Gentry, C. Fully homomorphic encryption using ideal lattices. In Mitzenmacher, M. (ed.), *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pp. 169–178. ACM, 2009. doi: 10.1145/1536414.1536440. URL https://doi.org/10.1145/1536414.1536440.

Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K. E., Naehrig, M., and Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 201–210. JMLR.org,

2016. URL http://proceedings.mlr.press/v48/gilad-bachrach16.html.

Goldreich, O. Secure multi-party computation. *Manuscript. Preliminary version*, 78(110), 1998.

Goldreich, O., Micali, S., and Wigderson, A. How to play any mental game or A completeness theorem for protocols with honest majority. In Aho, A. V. (ed.), *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pp. 218–229. ACM, 1987. doi: 10.1145/28395.28420. URL https://doi.org/10.1145/28395.28420.

Gupta, A., Dar, G., Goodman, S., Ciprut, D., and Berant, J. Memory-efficient transformers via top-k attention. In Moosavi, N. S., Gurevych, I., Fan, A., Wolf, T., Hou, Y., Marasovic, A., and Ravi, S. (eds.), *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing, SustaiNLP@EMNLP 2021, Virtual, November 10, 2021*, pp. 39–52. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.sustainlp-1.5. URL https://doi.org/10.18653/v1/2021.sustainlp-1.5.

Haberer, J. and Landsiedel, O. Activation sparsity and dynamic pruning for split computing in edge ai. In *Proceedings of the 3rd International Workshop on Distributed Machine Learning*, pp. 30–36, 2022.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1510.00149.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

He, Z., Zhang, T., and Lee, R. B. Attacking and protecting data privacy in edge–cloud collaborative inference systems. *IEEE Internet of Things Journal*, 8(12):9706–9716, 2020.

Ho, T.-T., Tran, K.-D., and Huang, Y. Fedsgdcovid: Federated sgd covid-19 detection under local differential privacy using chest x-ray images and symptom information. *Sensors*, 22(10):3728, 2022.

Home, G. Brands you love, united with google home. https://home.google.com/explore-devices/, 2023.

Hua, W., Umar, M., Zhang, Z., and Suh, G. E. Guardnn: Secure DNN accelerator for privacy-preserving deep learning. *CoRR*, abs/2008.11632, 2020. URL https://arxiv.org/abs/2008.11632.

Huang, P., Hoang, T., Li, Y., Shi, E., and Suh, G. E. Efficient privacy-preserving machine learning with lightweight trusted hardware. *arXiv preprint arXiv:2210.10133*, 2022a.

Huang, P., Hoang, T., Li, Y., Shi, E., and Suh, G. E. Stamp: Lightweight tee-assisted mpc for efficient privacy-preserving machine learning. *arXiv preprint arXiv:2210.10133*, 2022b.

Huang, Z., Lu, W., Hong, C., and Ding, J. Cheetah: Lean and fast secure two-party deep neural network inference. In Butler, K. R. B. and Thomas, K. (eds.), *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pp. 809–826. USENIX Association, 2022c. URL https://www.usenix.org/conference/usenixsecurity22/presentation/huang-zhicong.

Intel. Intel® Software Guard Extensions. https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html, 2021.

Jang, I., Tang, A., Kim, T., Sethumadhavan, S., and Huh, J. Heterogeneous isolated execution for commodity gpus. In Bahar, I., Herlihy, M., Witchel, E., and Lebeck, A. R. (eds.), *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*, pp. 455–468. ACM, 2019a. doi: 10.1145/3297858.3304021. URL https://doi.org/10.1145/3297858.3304021.

Jang, I., Tang, A., Kim, T., Sethumadhavan, S., and Huh, J. Heterogeneous isolated execution for commodity gpus. In Bahar, I., Herlihy, M., Witchel, E., and Lebeck, A. R. (eds.), *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*, pp. 455–468. ACM, 2019b. doi: 10.1145/3297858.3304021. URL https://doi.org/10.1145/3297858.3304021.

Jha, N. K. and Reagen, B. Deepreshape: Redesigning neural networks for efficient private inference. *CoRR*, abs/2304.10593, 2023. doi: 10.48550/arXiv.

2304.10593. URL https://doi.org/10.48550/arXiv.2304.10593.

Jha, N. K., Ghodsi, Z., Garg, S., and Reagen, B. Deepreduce: Relu reduction for fast private inference. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4839–4849. PMLR, 2021. URL http://proceedings.mlr.press/v139/jha21a.html.

Jiang, J., Qi, J., Shen, T., Chen, X., Zhao, S., Wang, S., Chen, L., Zhang, G., Luo, X., and Cui, H. CRONUS: fault-isolated, secure and high-performance heterogeneous computing for trusted execution environment. In *55th IEEE/ACM International Symposium on Microarchitecture, MICRO 2022, Chicago, IL, USA, October 1-5, 2022*, pp. 124–143. IEEE, 2022. doi: 10.1109/MICRO56248.2022.00019. URL https://doi.org/10.1109/MICRO56248.2022.00019.

Jung, W., Kim, S., Ahn, J. H., Cheon, J. H., and Lee, Y. Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021 (4):114–148, 2021. doi: 10.46586/tches.v2021.i4.114-148. URL https://doi.org/10.46586/tches.v2021.i4.114-148.

Juvekar, C., Vaikuntanathan, V., and Chandrakasan, A. P. GAZELLE: A low latency framework for secure neural network inference. In Enck, W. and Felt, A. P. (eds.), *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pp. 1651–1669. USENIX Association, 2018. URL https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar.

Kang, L., Xue, Y., Jia, W., Wang, X., Kim, J., Youn, C., Kang, M. J., Lim, H. J., Jacob, B. L., and Huang, J. Iceclave: A trusted execution environment for in-storage computing. In *MICRO '21: 54th Annual IEEE/ACM International Symposium on Microarchitecture, Virtual Event, Greece, October 18-22, 2021*, pp. 199–211. ACM, 2021. doi: 10.1145/3466752.3480109. URL https://doi.org/10.1145/3466752.3480109.

Kang, Y., Hauswald, J., Gao, C., Rovinski, A., Mudge, T., Mars, J., and Tang, L. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

Khandaker, M. R., Cheng, Y., Wang, Z., and Wei, T. COIN attacks: On insecurity of enclave untrusted interfaces in SGX. In Larus, J. R., Ceze, L., and Strauss, K. (eds.), *ASPLOS '20: Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, March 16-20, 2020*, pp. 971–985. ACM, 2020. doi: 10.1145/3373376.3378486. URL https://doi.org/10.1145/3373376.3378486.

Kim, D., Park, J., Kim, J., Kim, S., and Ahn, J. H. Hyphen: A hybrid packing method and optimizations for homomorphic encryption-based neural networks. *CoRR*, abs/2302.02407, 2023. doi: 10.48550/arXiv.2302.02407. URL https://doi.org/10.48550/arXiv.2302.02407.

Kim, J., Lee, G., Kim, S., Sohn, G., Rhu, M., Kim, J., and Ahn, J. H. ARK: fully homomorphic encryption accelerator with runtime data generation and inter-operation key reuse. In *55th IEEE/ACM International Symposium on Microarchitecture, MICRO 2022, Chicago, IL, USA, October 1-5, 2022*, pp. 1237–1254. IEEE, 2022a. doi: 10.1109/MICRO56248.2022.00086. URL https://doi.org/10.1109/MICRO56248.2022.00086.

Kim, S., Kim, J., Kim, M. J., Jung, W., Kim, J., Rhu, M., and Ahn, J. H. BTS: an accelerator for bootstrappable fully homomorphic encryption. In Salapura, V., Zahran, M., Chong, F., and Tang, L. (eds.), *ISCA '22: The 49th Annual International Symposium on Computer Architecture, New York, New York, USA, June 18 - 22, 2022*, pp. 711–725. ACM, 2022b. doi: 10.1145/3470496.3527415. URL https://doi.org/10.1145/3470496.3527415.

Knott, B., Venkataraman, S., Hannun, A. Y., Sengupta, S., Ibrahim, M., and van der Maaten, L. Crypten: Secure multi-party computation meets machine learning. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 4961–4973, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/2754518221cfbc8d25c13a06a4cb8421-Abstract.html.

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

Kumar, N., Rathee, M., Chandran, N., Gupta, D., Rastogi, A., and Sharma, R. Cryptflow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pp. 336–353. IEEE, 2020. doi: 10.1109/SP40000.2020.00092. URL https://doi.org/10.1109/SP40000.2020.00092.

Kundu, S., Lu, S., Zhang, Y., Liu, J. T., and Beerel, P. A. Learning to linearize deep neural networks for secure and efficient private inference. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=BGF9IeDfmlH.

Kurtz, M., Kopinsky, J., Gelashvili, R., Matveev, A., Carr, J., Goin, M., Leiserson, W. M., Moore, S., Shavit, N., and Alistarh, D. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5533–5543. PMLR, 2020. URL http://proceedings.mlr.press/v119/kurtz20a.html.

Lam, M., Johnson, J., Xiong, W., Maeng, K., Gupta, U., Rhu, M., Lee, H. S., Reddi, V. J., Wei, G., Brooks, D., and Suh, G. E. Gpu-based private information retrieval for on-device machine learning inference. *CoRR*, abs/2301.10904, 2023. doi: 10.48550/arXiv.2301.10904. URL https://doi.org/10.48550/arXiv.2301.10904.

Lee, D., Kohlbrenner, D., Shinde, S., Asanovic, K., and Song, D. Keystone: an open framework for architecting trusted execution environments. In Bilas, A., Magoutis, K., Markatos, E. P., Kostic, D., and Seltzer, M. I. (eds.), *EuroSys '20: Fifteenth EuroSys Conference 2020, Heraklion, Greece, April 27-30, 2020*, pp. 38:1–38:16. ACM, 2020. doi: 10.1145/3342195.3387532. URL https://doi.org/10.1145/3342195.3387532.

Lee, R. B., Kwan, P. C. S., McGregor, J. P., Dwoskin, J. S., and Wang, Z. Architecture for protecting critical secrets in microprocessors. In *32st International Symposium on Computer Architecture (ISCA 2005), 4-8 June 2005, Madison, Wisconsin, USA*, pp. 2–13. IEEE Computer Society, 2005. doi: 10.1109/ISCA.2005.14. URL https://doi.org/10.1109/ISCA.2005.14.

Lee, S., Kim, J., Na, S., Park, J., and Huh, J. TNPU: supporting trusted execution with tree-less integrity protection for neural processing unit. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2022, Seoul, South Korea, April 2-6, 2022*, pp. 229–243. IEEE, 2022. doi: 10.1109/HPCA53966.2022.00025. URL https://doi.org/10.1109/HPCA53966.2022.00025.

Li, D., Wang, H., Shao, R., Guo, H., Xing, E. P., and Zhang, H. MPCFORMER: fast, performant and provate transformer inference with MPC. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023a. URL https://openreview.net/pdf?id=CWmvjOEhgH-.

Li, J., Rakin, A. S., Chen, X., He, Z., Fan, D., and Chakrabarti, C. Ressfl: A resistance transfer framework for defending model inversion attack in split federated learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 10184–10192. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00995. URL https://doi.org/10.1109/CVPR52688.2022.00995.

Li, Z., You, C., Bhojanapalli, S., Li, D., Rawat, A. S., Reddi, S. J., Ye, K., Chern, F., Yu, F. X., Guo, R., and Kumar, S. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b. URL https://openreview.net/pdf?id=TJ2nxciYCk-.

Lie, D., Thekkath, C., Mitchell, M., Lincoln, P., Boneh, D., Mitchell, J., and Horowitz, M. Architectural support for copy and tamper resistant software. *Acm Sigplan Notices*, 35(11):168–177, 2000.

Lie, D., Mitchell, J. C., Thekkath, C. A., and Horowitz, M. Specifying and verifying hardware for tamper-resistant software. In *2003 IEEE Symposium on Security and Privacy (S&P 2003), 11-14 May 2003, Berkeley, CA, USA*, pp. 166. IEEE Computer Society, 2003. doi: 10.1109/SECPRI.2003.1199335. URL https://doi.org/10.1109/SECPRI.2003.1199335.

Liu, J., Juuti, M., Lu, Y., and Asokan, N. Oblivious neural network predictions via minionn transformations. In Thuraisingham, B., Evans, D., Malkin, T., and Xu, D. (eds.), *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 619–631. ACM, 2017. doi: 10.1145/3133956.3134056. URL https://doi.org/10.1145/3133956.3134056.

Liu, W., Chen, H., Wang, X., Li, Z., Zhang, D., Wang, W., and Tang, H. Understanding tee containers, easy to use? hard to trust. *arXiv preprint arXiv:2109.01923*, 2021.

Lou, Q., Shen, Y., Jin, H., and Jiang, L. Safenet: A secure, accurate and fast neural network inference. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Cz3dbFm5u-.

Maeng, K., Guo, C., Kariyappa, S., and Suh, E. Measuring and controlling split layer privacy leakage using fisher information. *arXiv preprint arXiv:2209.10119*, 2022.

Maeng, K., Guo, C., Kariyappa, S., and Suh, G. E. Bounding the invertibility of privacy-preserving instance encoding using fisher information. *arXiv preprint arXiv:2305.04146*, 2023.

McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C. V., Shafi, H., Shanbhogue, V., and Savagaonkar, U. R. Innovative instructions and software model for isolated execution. In Lee, R. B. and Shi, W. (eds.), *HASP 2013, The Second Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, June 23-24, 2013*, pp. 10. ACM, 2013. doi: 10.1145/2487726.2488368. URL https://doi.org/10.1145/2487726.2488368.

Meta. Meta portal go. https://www.meta.com/portal/products/portal-go/, 2023a.

Meta. The value of secure multi-party computation. https://privacytech.fb.com/multi-party-computation/, 2023b.

Mireshghallah, F., Taram, M., Ramrakhyani, P., Jalali, A., Tullsen, D., and Esmaeilzadeh, H. 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.

Mireshghallah, F., Taram, M., Jalali, A., Elthakeb, A. T., Tullsen, D. M., and Esmaeilzadeh, H. Not all features are equal: Discovering essential features for preserving prediction privacy. In Leskovec, J., Grobelnik, M., Najork, M., Tang, J., and Zia, L. (eds.), *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pp. 669–680. ACM / IW3C2, 2021. doi: 10.1145/3442381.3449965. URL https://doi.org/10.1145/3442381.3449965.

Mishra, P., Lehmkuhl, R., Srinivasan, A., Zheng, W., and Popa, R. A. Delphi: A cryptographic inference service for neural networks. In Capkun, S. and Roesner, F. (eds.), *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pp. 2505–2522. USENIX Association, 2020. URL https://www.usenix.org/conference/usenixsecurity20/presentation/mishra.

Mohassel, P. and Rindal, P. Aby^3: A mixed protocol framework for machine learning. In Lie, D., Mannan, M., Backes, M., and Wang, X. (eds.), *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pp. 35–52. ACM, 2018. doi: 10.1145/3243734.3243760. URL https://doi.org/10.1145/3243734.3243760.

Mohassel, P. and Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 19–38. IEEE Computer Society, 2017. doi: 10.1109/SP.2017.12. URL https://doi.org/10.1109/SP.2017.12.

NVIDIA. NVIDIA CONFIDENTIAL COMPUTING. https://www.nvidia.com/en-us/data-center/solutions/confidential-computing/, 2022.

NVIDIA. NVLink and NVSwitch. https://www.nvidia.com/en-us/data-center/nvlink/, 2023.

Oh, C., So, J., Kim, S., and Yi, Y. Exploiting activation sparsity for fast CNN inference on mobile gpus. *ACM Trans. Embed. Comput. Syst.*, 20(5s):77:1–77:25, 2021. doi: 10.1145/3477008. URL https://doi.org/10.1145/3477008.

Pang, Q., Zhu, J., Möllering, H., Zheng, W., and Schneider, T. BOLT: privacy-preserving, accurate and efficient inference for transformers. *IACR Cryptol. ePrint Arch.*, pp. 1893, 2023. URL https://eprint.iacr.org/2023/1893.

Park, J., Kim, M. J., Jung, W., and Ahn, J. H. AESPA: accuracy preserving low-degree polynomial activation for fast private inference. *CoRR*, abs/2201.06699, 2022. URL https://arxiv.org/abs/2201.06699.

Patra, A. and Suresh, A. BLAZE: blazing fast privacy-preserving machine learning. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. URL https://www.ndss-symposium.org/ndss-paper/blaze-blazing-fast-privacy-preserving-machine-learning/.

Poirot, M. G., Vepakomma, P., Chang, K., Kalpathy-Cramer, J., Gupta, R., and Raskar, R. Split learning for collaborative deep learning in healthcare. *arXiv preprint arXiv:1912.12115*, 2019.

Rathee, D., Rathee, M., Kumar, N., Chandran, N., Gupta, D., Rastogi, A., and Sharma, R. Cryptflow2: Practical 2-party secure inference. In Ligatti, J., Ou, X., Katz, J., and Vigna, G. (eds.), *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pp. 325–342. ACM, 2020.

doi: 10.1145/3372297.3417274. URL https://doi.org/10.1145/3372297.3417274.

Rathee, D., Rathee, M., Goli, R. K. K., Gupta, D., Sharma, R., Chandran, N., and Rastogi, A. Sirnn: A math library for secure RNN inference. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pp. 1003–1020. IEEE, 2021. doi: 10.1109/SP40001.2021.00086. URL https://doi.org/10.1109/SP40001.2021.00086.

Riazi, M. S., Weinert, C., Tkachenko, O., Songhori, E. M., Schneider, T., and Koushanfar, F. Chameleon: A hybrid secure computation framework for machine learning applications. In Kim, J., Ahn, G., Kim, S., Kim, Y., López, J., and Kim, T. (eds.), *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pp. 707–721. ACM, 2018. doi: 10.1145/3196494.3196522. URL https://doi.org/10.1145/3196494.3196522.

Riazi, M. S., Laine, K., Pelton, B., and Dai, W. HEAX: an architecture for computing on encrypted data. In Larus, J. R., Ceze, L., and Strauss, K. (eds.), *ASPLOS '20: Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, March 16-20, 2020*, pp. 1295–1309. ACM, 2020. doi: 10.1145/3373376.3378523. URL https://doi.org/10.1145/3373376.3378523.

Roy, S. S., Turan, F., Järvinen, K., Vercauteren, F., and Verbauwhede, I. Fpga-based high-performance parallel architecture for homomorphic computing on encrypted data. In *25th IEEE International Symposium on High Performance Computer Architecture, HPCA 2019, Washington, DC, USA, February 16-20, 2019*, pp. 387–398. IEEE, 2019. doi: 10.1109/HPCA.2019.00052. URL https://doi.org/10.1109/HPCA.2019.00052.

Samardzic, N., Feldmann, A., Krastev, A., Devadas, S., Dreslinski, R. G., Peikert, C., and Sánchez, D. F1: A fast and programmable accelerator for fully homomorphic encryption. In *MICRO '21: 54th Annual IEEE/ACM International Symposium on Microarchitecture, Virtual Event, Greece, October 18-22, 2021*, pp. 238–252. ACM, 2021. doi: 10.1145/3466752.3480070. URL https://doi.org/10.1145/3466752.3480070.

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 4510–4520. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00474. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html.

Stanford. [TinyImageNet download link]. http://cs231n.stanford.edu/tiny-imagenet-200.zip, 2023.

Suh, G. E., O'Donnell, C. W., and Devadas, S. Aegis: A single-chip secure processor. *IEEE Design & Test of Computers*, 24(6):570–580, 2007.

Szefer, J. and Lee, R. B. Architectural support for hypervisor-secure virtualization. *ACM SIGPLAN Notices*, 47(4):437–450, 2012.

Tan, S., Knott, B., Tian, Y., and Wu, D. J. Cryptgpu: Fast privacy-preserving machine learning on the GPU. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pp. 1021–1038. IEEE, 2021. doi: 10.1109/SP40001.2021.00098. URL https://doi.org/10.1109/SP40001.2021.00098.

Thapa, C., Chamikara, M. A. P., Camtepe, S., and Sun, L. Splitfed: When federated learning meets split learning. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 8485–8493. AAAI Press, 2022. URL https://ojs.aaai.org/index.php/AAAI/article/view/20825.

Titcombe, T., Hall, A. J., Papadopoulos, P., and Romanini, D. Practical defences against model inversion attacks for split neural networks. *arXiv preprint arXiv:2104.05743*, 2021.

Vepakomma, P., Gupta, O., Swedish, T., and Raskar, R. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

Vepakomma, P., Singh, A., Gupta, O., and Raskar, R. Nopeek: Information leakage reduction to share activations in distributed deep learning. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pp. 933–942. IEEE, 2020.

Vepakomma, P., Singh, A., Zhang, E., Gupta, O., and Raskar, R. Nopeek-infer: Preventing face reconstruction attacks in distributed inference after on-premise training. In *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, pp. 1–8. IEEE, 2021.

Volos, S., Vaswani, K., and Bruno, R. Graviton: Trusted execution environments on gpus. In Arpaci-Dusseau, A. C. and Voelker, G. (eds.), *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018*, pp. 681–696. USENIX Association, 2018. URL https://www.usenix.org/conference/osdi18/presentation/volos.

Wagh, S., Gupta, D., and Chandran, N. Securenn: 3-party secure computation for neural network training. *Proc. Priv. Enhancing Technol.*, 2019(3):26–49, 2019. doi: 10.2478/popets-2019-0035. URL https://doi.org/10.2478/popets-2019-0035.

Wagh, S., Tople, S., Benhamouda, F., Kushilevitz, E., Mittal, P., and Rabin, T. Falcon: Honest-majority maliciously secure framework for private deep learning. *Proc. Priv. Enhancing Technol.*, 2021(1):188–208, 2021. doi: 10.2478/popets-2021-0011. URL https://doi.org/10.2478/popets-2021-0011.

Wang, W., Chen, G., Pan, X., Zhang, Y., Wang, X., Bindschaedler, V., Tang, H., and Gunter, C. A. Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX. In Thuraisingham, B., Evans, D., Malkin, T., and Xu, D. (eds.), *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 2421–2434. ACM, 2017. doi: 10.1145/3133956.3134038. URL https://doi.org/10.1145/3133956.3134038.

Wang, Y., Suh, G. E., Xiong, W., Lefaudeux, B., Knott, B., Annavaram, M., and Lee, H. S. Characterization of mpc-based private inference for transformer-based models. In *International IEEE Symposium on Performance Analysis of Systems and Software, ISPASS 2022, Singapore, May 22-24, 2022*, pp. 187–197. IEEE, 2022. doi: 10.1109/ISPASS55109.2022.00025. URL https://doi.org/10.1109/ISPASS55109.2022.00025.

Watson, R. N. M., Woodruff, J., Neumann, P. G., Moore, S. W., Anderson, J., Chisnall, D., Dave, N. H., Davis, B., Gudka, K., Laurie, B., Murdoch, S. J., Norton, R. M., Roe, M., Son, S. D., and Vadera, M. CHERI: A hybrid capability-system architecture for scalable software compartmentalization. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pp. 20–37. IEEE Computer Society, 2015. doi: 10.1109/SP.2015.9. URL https://doi.org/10.1109/SP.2015.9.

Xiang, L., Zhang, H., Ma, H., Zhang, Y., Ren, J., and Zhang, Q. Interpretable complex-valued neural networks for privacy protection. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=S1xFl64tDr.

Xiao, H. and Devadas, S. Dauntless: Data augmentation and uniform transformation for learning with scalability and security. *IACR Cryptol. ePrint Arch.*, pp. 201, 2021. URL https://eprint.iacr.org/2021/201.

Xiao, H. and Devadas, S. Pac security: Automatic privacy measurement and control of data processing. *arXiv preprint arXiv:2210.03458*, 2022.

Yala, A., Esfahanizadeh, H., D'Oliveira, R. G. L., Duffy, K. R., Ghobadi, M., Jaakkola, T. S., Vaikuntanathan, V., Barzilay, R., and Médard, M. Neuracrypt: Hiding private health data via random neural networks for public training. *CoRR*, abs/2106.02484, 2021. URL https://arxiv.org/abs/2106.02484.

Yang, J., Zhang, Y., and Gao, L. Fast secure processor for inhibiting software piracy and tampering. In *Proceedings of the 36th Annual International Symposium on Microarchitecture, San Diego, CA, USA, December 3-5, 2003*, pp. 351–360. IEEE Computer Society, 2003. doi: 10.1109/MICRO.2003.1253209. URL https://doi.org/10.1109/MICRO.2003.1253209.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

Yao, A. C. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pp. 160–164. IEEE Computer Society, 1982. doi: 10.1109/SFCS.1982.38. URL https://doi.org/10.1109/SFCS.1982.38.

Zeng, W., Li, M., Xiong, W., Tong, T., Lu, W.-j., Tan, J., Wang, R., and Huang, R. Mpcvit: Searching for accurate and efficient mpc-friendly vision transformer with heterogeneous attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5052–5063, 2023.

Zhang, Y., Chen, D., Kundu, S., Li, C., and Beerel, P. A. Sal-vit: Towards latency efficient private inference on vit using selective attention search with a learnable softmax approximation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5116–5125, 2023.

Zhao, M., Gao, M., and Kozyrakis, C. Shef: shielded enclaves for cloud fpgas. In Falsafi, B., Ferdman, M., Lu, S., and Wenisch, T. F. (eds.), *ASPLOS '22: 27th*

ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, 28 February 2022 - 4 March 2022, pp. 1070–1085. ACM, 2022. doi: 10.1145/3503222.3507733. URL https://doi.org/10.1145/3503222.3507733.

Zhou, X., Xu, Z., Wang, C., and Gao, M. PPMLAC: high performance chipset architecture for secure multi-party computation. In Salapura, V., Zahran, M., Chong, F., and Tang, L. (eds.), *ISCA '22: The 49th Annual International Symposium on Computer Architecture, New York, New York, USA, June 18 - 22, 2022*, pp. 87–101. ACM, 2022. doi: 10.1145/3470496.3527392. URL https://doi.org/10.1145/3470496.3527392.

Zhu, J., Hou, R., Wang, X., Wang, W., Cao, J., Zhao, B., Wang, Z., Zhang, Y., Ying, J., Zhang, L., and Meng, D. Enabling rack-scale confidential computing using heterogeneous trusted execution environment. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pp. 1450–1465. IEEE, 2020. doi: 10.1109/SP40000.2020.00054. URL https://doi.org/10.1109/SP40000.2020.00054.

Zuo, P., Hua, Y., Liang, L., Xie, X., Hu, X., and Xie, Y. Sealing neural network models in secure deep learning accelerators. *CoRR*, abs/2008.03752, 2020. URL https://arxiv.org/abs/2008.03752.

A Appendix

A.1 Baseline Models

For CIFAR10, we replaced the max pooling with average pooling, following (Garimella et al., 2023; 2021a). For the rest, we simply removed max pooling (as average pooling did not work well), following (Cho et al., 2022b).

Table 1. Baseline model accuracy.

Dataset	Model	Accuracy
CIFAR10	ResNet18	92.78%
	ResNet50	93.15%
CIFAR100	ResNet18	77.98%
	ResNet50	79.36%
Tiny-ImageNet	ResNet18	65.46%
	ResNet50	66.87%

A.2 HummingBird Search Time

Table 2 summarizes the search time of the HummingBird search engine.

Table 2. HummingBird's configuration search time.

Dataset	Model	Search budget 8/64	Search budget 6/64
CIFAR10	ResNet18	5m 34s	4m 28s
	ResNet50	6m 10s	5m 47s
CIFAR100	ResNet18	5m 37s	4m 19s
	ResNet50	18m 32s	18m 34s
Tiny-ImageNet	ResNet18	13m 1s	11m 34s
	ResNet50	42m 3s	1h 8m

A.3 Effectiveness of Finetuning

Table 3 summarizes the model accuracy before and after finetuning.

Table 3. Impact of finetuning (FT) on HummingBird-6/64.

Dataset	Model	Before FT	After TF
CIFAR10	ResNet18	90.09%	91.04%
	ResNet50	87.6%	91.12%
CIFAR100	ResNet18	73.04%	75.57%
	ResNet50	72.45%	78.49%
Tiny-ImageNet	ResNet18	60.21%	64.79%
	ResNet50	59.82%	66.47%

A.4 Proofs

We only provide proofs for a 2-party case ($p \in \{0, 1\}$), while Theorem 1 can be extended to more parties.

Proof for Theorem 1

Proof. $\langle x \rangle^Q[k:0]$ can be seen as secret shares of $x[k:0]$ in $\mathbb{Z}/2^k\mathbb{Z}$. This is because $\langle x \rangle^Q[k:0] \equiv \langle x \rangle^Q \pmod{2^k}$ and $x[k:0] \equiv x \pmod{2^k}$, and thus, applying $\pmod{2^k}$ to both sides of

$$\langle x \rangle_0^Q + \langle x \rangle_1^Q \equiv x \pmod{2^N}$$

results in

$$\langle x \rangle_0^Q[k:0] + \langle x \rangle_1^Q[k:0] \equiv x[k:0] \pmod{2^k}.$$

Applying DReLU to $\langle x \rangle^Q[k:0]$ on a smaller ring $\mathbb{Z}/2^k\mathbb{Z}$ will simply output secret shares indicating whether its secret ($x[k:0]$) is positive. Thus, $\mathrm{DReLU}(\langle x \rangle^Q) = \mathrm{DReLU}(\langle x \rangle^Q[k:0])$ if and only if their secrets ($x[k:0] \in \mathbb{Z}/2^k\mathbb{Z}$ and $x \in \mathbb{Z}/Q\mathbb{Z}$) have the same sign bits, *i.e.*, $x[k-1] = x[N-1]$. This is always the case if (but not only if) $-2^{k-1} \leq x < 2^{k-1}$. □

Proof for Theorem 2

Proof. Note that $\langle x \rangle^Q[N:m] = \lfloor \frac{\langle x \rangle^Q}{2^m} \rfloor$. Consequently,

$$\begin{aligned}
&\langle x \rangle_0^Q[N:m] + \langle x \rangle_1^Q[N:m] \\
&\equiv \lfloor \frac{\langle x \rangle_0^Q}{2^m} \rfloor + \lfloor \frac{\langle x \rangle_1^Q}{2^m} \rfloor \\
&\equiv \begin{cases} \lfloor \frac{x}{2^m} \rfloor \pmod{2^{N-m}}, & \text{or} \\ \lfloor \frac{x}{2^m} \rfloor - 1 \pmod{2^{N-m}}. \end{cases}
\end{aligned}$$

In other words, $\langle x \rangle^Q[N:m] \in \mathbb{Z}/2^{N-m}\mathbb{Z}$ are secret shares of either $\lfloor \frac{x}{2^m} \rfloor$ or $\lfloor \frac{x}{2^m} \rfloor - 1$ in $\mathbb{Z}/2^{N-m}\mathbb{Z}$. The sign of the former is always the same as x (here, for simplicity we consider zero as positive, which does not make any difference for ReLU), so applying DReLU yields the same sign as x. The latter can cause the sign to flip if (1) $0 < x < 2^m$ ($\lfloor \frac{x}{2^m} \rfloor$ smaller than 1), or (2) $\lfloor \frac{x}{2^m} \rfloor - 1 < -2^{N-m-1}$ (underflow).

The first case incorrectly consider secrets in $0 < x < 2^m$ as negative and output secret shares of zero, which will cause the corresponding ReLU result to become zero. The behavior is equivalent to magnitude-based activation pruning with a threshold 2^m. The second case becomes rare when N is large enough, compare to the values of x. □