

# AESPA: Accuracy Preserving Low-degree Polynomial Activation for Fast Private Inference

Jaiyoung Park  
Seoul National University  
jypark@scale.snu.ac.kr

Michael Jaemin Kim  
Seoul National University  
michael604@scale.snu.ac.kr

Wonkyung Jung  
Seoul National University  
wkjung@scale.snu.ac.kr

Jung Ho Ahn  
Seoul National University  
gajh@snu.ac.kr

**Abstract**—Hybrid private inference (PI) protocol, which synergistically utilizes both multi-party computation (MPC) and homomorphic encryption, is one of the most prominent techniques for PI. However, even the state-of-the-art PI protocols are bottlenecked by the non-linear layers, especially the activation functions. Although a standard non-linear activation function can generate higher model accuracy, it must be processed via a costly garbled-circuit MPC primitive. A polynomial activation can be processed via Beaver’s multiplication triples MPC primitive but has been incurring severe accuracy drops so far.

In this paper, we propose an accuracy preserving low-degree polynomial activation function (AESPA) that exploits the Hermite expansion of the ReLU and basis-wise normalization. We apply AESPA to popular ML models, such as VGGNet, ResNet, and pre-activation ResNet, to show classification accuracy comparable to those of the standard models with ReLU activation, achieving superior accuracy over prior low-degree polynomial studies. When applied to the all-ReLU baseline on the state-of-the-art Delphi PI protocol, AESPA shows up to  $61.4\times$  and  $28.9\times$  lower online latency and communication cost.

**Index Terms**—Private Inference, Homomorphic Encryption, Marty-party Computation

## I. INTRODUCTION

*Private inference (PI) protocols* [1]–[3], or hybrid PI protocols exploiting multi-party-computation (MPC [4]) and homomorphic encryption (HE [5]), are one of the most prominent approaches in machine-learning-as-a-service (MLaaS) to support sensitive data, such as medical images [6], [7]. In these protocols, MPC primitives and HE are combined in a complementary manner (e.g., HE is used for linear layers and MPC primitives used for non-linear layers). Although promising, the performance (i.e., serving latency) of the PI protocols is commonly bottlenecked by the non-linear layers of the machine learning (ML) models, especially activation functions [2], [3]. While standard activation functions such as ReLU can generate higher model accuracy, they rely on slow and communication/storage-hungry primitives such as garbled-circuit (GC) MPC primitive [8] due to the required bitwise operations. In contrast, the polynomial activation functions have an advantage in that they can be processed via less costly Beaver’s multiplication triples (BT) MPC primitive [9], but are limited by the lower model accuracy.

Prior works have tackled this challenge in various ways. First, one set of prior arts sought the effective approximations of the standard activation functions, such as ReLU, which minimize the model accuracy degradation and evaluation cost

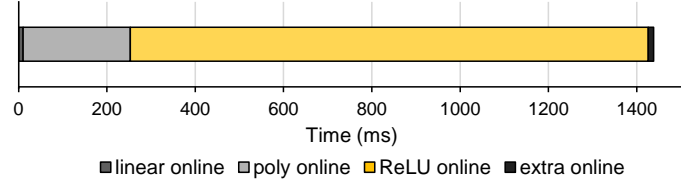


Fig. 1. The online latency breakdown of Delphi [2] for ResNet32, where 26 ReLU functions are approximated with quadratic polynomials and processed via Beaver’s multiplication triples, whereas 5 ReLUs are processed as via Garbled circuit. Experimental details in Section IV-A.

at the same time. [10]–[17] explored various polynomial activations to replace ReLU, especially based on numerical analysis such as Taylor polynomials or minimax approximation [18]. Circa [19] broke down ReLU into a piecewise linear function and a sign function to partly utilize BT, and approximating sign for reducing the cost of GC.

However, the ML models with such approximated activation often suffer from either limited trainability when the model is deep for a large task, low model accuracy, or high serving latency when pursuing high accuracy. [10] was limited to a shallow model for a simple task of MNIST. [17] minimized the accuracy drop for deeper models with larger tasks of CIFAR-10 [20] and ImageNet [21] by exploiting the minimax approximation function. However, it was limited from the long latency caused by using high polynomial degrees (e.g., 29) for approximation; a higher degree means that more MPC primitives must be used. Circa [19] demonstrated a better (lower) latency, but still leaves a large room for improvement.

Second, another set of studies [2], [22] utilized the neural architecture search (NAS) method [23] to find optimal activation layer locations (in terms of model accuracy and online inference latency) to substitute with the polynomial activation while leaving others with the original ReLU. However, despite the use of NAS, the acquired models still exhibit large room for improvement in both accuracy and latency because not all activation layers can be replaced with polynomials, and the remaining few activation layers take up the majority of the total online latency. Delphi [2], a state-of-the-art PI protocol utilizing NAS, greatly reduced the online latency, achieving 3.8 seconds and 65.7% accuracy for ResNet32 on CIFAR-100, when 26 out of 31 ReLUs are replaced into a quadratic approximation, for instance. However, up to 82% of the online latency is spent on GC for the five remaining ReLUs (see Figure 1); no more ReLUs can be replaced due to the sharp

drop in the model accuracy.

In this paper, we propose an Accuracy prEServing low-degree Polynomial Activation (AESPA) that can replace all ReLUs in the neural networks, using the *Hermite polynomial* concatenated with the *basis-wise normalization*. While prior studies [24]–[27] inspired our work, they were mainly in the theoretical domain. None had explored nor demonstrated the practicality of the Hermite polynomials in the PI context or was able to reach high accuracy as our method. Our composition of Hermite expansion and basis-wise normalization substantially mitigated vanishing (or exploding) gradient problems, which are frequently encountered with the polynomial activation functions, allowing us to achieve high model accuracy even using the low-degree polynomials.

We first present the organization of our Hermite polynomial with the basis-wise normalization (HerPN) block and show how it can be used to modify ResNet [28], pre-activation ResNet (PA-ResNet [29]), and VGG [30] (Section III). Then, we report the smaller (better) accuracy drop of the trained models using the HerPN blocks for CIFAR-10/100, and Tiny-ImageNet datasets [31] over the prior polynomial activation ML models of [11], [17]. Moreover, we demonstrate the effectiveness of the HerPN-modified ML models for a state-of-the-art PI protocol, Delphi, on the real machine and network (Section IV). By replacing all ReLUs with our HerPN blocks and processing them with BT, the online latency and communication for evaluating the ML models, such as ResNet18/32, PA-ResNet18/32, and VGG16, are reduced by up to  $61.4\times$  and  $28.9\times$ , respectively. The accuracies of the ML models with the HerPN block are also competitive with the original models with ReLU for CIFAR-10/100 and TinyImageNet.

The key contributions of this paper are as follows:

- We propose a novel low-degree polynomial activation function that utilizes the Hermite polynomial concatenated with the basis-wise normalization, which sustains the high accuracy of the ML model.
- We demonstrate superior accuracy over prior polynomial activation works on large datasets such as CIFAR-10/100.
- We showcase the effectiveness of our work in the PI context, in terms of latency and communication costs, on a real machine and network using the open-source Delphi PI protocol.

## II. BACKGROUND

### A. Private inference (PI) and hybrid PI protocols

Private inference (PI) refers to a set of techniques that enable machine learning (ML) inference without the need to reveal i) the private data of a client to a service provider or ii) the trained model of a service provider to a client. Fully-homomorphic-encryption (FHE) [5], [32]–[34], federated learning [35], [36], or works based on enclaves [37], [38] all pursue practical PI. However, none has proven to be dominantly superior to the others regarding the security level, latency, or accuracy of the supporting ML models. In particular, while FHE might be a promising option for the PI, it is limited by i) the extremely costly bootstrapping operation

to enable an unlimited depth of operations for deep ML models and ii) the limited applicability to arbitrary operations.

Hybrid *PI protocols* based on both MPC primitives and HE attempt to provide a prominent option for private inference [1]–[3]. These PI protocols often exploit the leveled-HE (LHE, which does not support bootstrapping) for ML *linear layers*. As LHE evaluates a single linear layer at a time, no bootstrapping operation is necessary, thus incurring a shorter computation time than FHE. A PI protocol often processes the non-linear activation function via MPC primitives. MPC primitives require minimal computation time at the cost of high communication costs between the client and the server. Our work focuses on the latter of processing activation functions with the MPC primitives.

### B. Cryptographic primitives

The hybrid PI protocols rely on multiple cryptographic primitives, either the ones from MPC or HE, to effectively support the private inference. Parameters about the finite field and data representation in cryptographic primitives are explained in Appendix B.

**Additive secret sharing** (SS [39]): SS is a cryptographic primitive that divides a secret value  $\langle x \rangle$  into multiple shares  $\langle x \rangle_1, \langle x \rangle_2, \dots, \langle x \rangle_n$  distributed to  $n$  parties so that a single party does not have complete knowledge of the original secret. In a two-party SS, the secret shares of the value  $x$  can be generated by randomly sampling a value  $r$  and distributing the shares  $\langle x \rangle_1 = r$  and  $\langle x \rangle_2 = x - r$  to each party. The reconstruction of the original secret is straightforward, by pooling shared values from the two parties and adding up:  $\langle x \rangle = \langle x \rangle_1 + \langle x \rangle_2$ .

**Beaver’s multiplication triples** (BT [9]): BT is a two-party protocol that can securely compute the product of the two secret-shared values. The whole protocol is divided into two steps: generation and multiplication procedures. For the first step to prepare for the multiplication, Beaver’s triples are generated; randomly sampled  $a$ ,  $b$ , and  $a \cdot b$  are secret-shared between two parties  $P_1$  and  $P_2$ . As a second step, when the actual input  $x$  and  $y$  are already secret-shared (namely  $P_1$  holds  $\langle x \rangle_1, \langle y \rangle_1$  and  $P_2$  holds  $\langle x \rangle_2, \langle y \rangle_2$ ), through Beaver’s multiplication procedure the SS of their product is generated and shared (namely  $\langle xy \rangle_1$  for  $P_1$  and  $\langle xy \rangle_2$  for  $P_2$ ). During this second step, one set of Beaver’s triples is consumed for every multiplication.

**Garbled circuit** (GC [8]): GC is a two-party protocol that enables garbler and evaluator to securely compute an arbitrary boolean circuit ( $C$ ) without revealing their private inputs. Initially, a garbler holds its private data  $x_g$  and boolean circuit  $C$ , while an evaluator holds its private data  $x_e$ . The protocol starts with the garbler encoding (garbling) the circuit to generate garbled circuit  $\hat{C}$ , which takes encoded inputs (labels) and evaluate the original circuit  $C$ . The garbler sends  $\hat{C}$  and encoded data  $L(x_g)$  to the evaluator. The label for the evaluator’s input,  $L(x_e)$ , can be computed by using the

TABLE I  
LATENCY AND COMMUNICATION COSTS OF BT PROCESSING POLYNOMIAL  
ACTIVATION AND GC PROCESSING ReLU. EACH IS MEASURED IN THE  
DELPHI PROTOCOL FOR RESNET32, AMORTIZED TO A SINGLE  
OPERATION.

Activation function	Time ( $\mu$ s)		Comm. (KB)	
	Offline	Online	Offline	Online
Polynomial activation	2.80	1.20	0.192	0.036
ReLU	60.60	20.22	19.088	1.184

Oblivious Transfer protocol<sup>1</sup> [40] with the garbler. Finally, the evaluator computes  $Eval(\hat{C}, L(x_g), L(x_e))$ , which outputs  $y = C(x_g, x_e)$ .

**Homomorphic encryption** (HE [5], [32]–[34]): HE is a set of public-key encryption schemes that allow arithmetic operations on the ciphertext without the need for decryption. Assuming messages  $m_1$  and  $m_2$ , a public-key  $pk$ , a secret key  $sk$ , an HE encryption scheme  $\mathbb{E}$ , and an HE decryption scheme  $\mathbb{D}$ , a homomorphic evaluation  $Eval$  satisfies the following upon some function of  $f$ :  $\mathbb{D}(sk, Eval(pk, \mathbb{E}(pk, m_1), \mathbb{E}(pk, m_2), f)) = f(m_1, m_2)$ . Due to the HE nature that errors accumulate over computations, only a limited number of HE operations ( $Eval$ ) can be executed upon an encrypted message. A bootstrapping operation can reset the accumulated errors, but only at the cost of high computational complexity. A leveled-HE refers to a set of HE schemes that do not support bootstrapping operation incurring lower cost but providing limited depth, while fully-HE refers to those that support the bootstrapping.

GC supports arbitrary functions such as ReLU, but only at the cost of high communication/storage for the garbled circuits and exchanged labels. BT supports multiplications with relatively lower-cost than GC but can only support polynomial activation functions, which leads to lower model accuracy. Table I summarizes the amortized latency and communication costs of BT and GC, while each processing polynomial activation and ReLU, respectively. HE is different in that it does not require multiple communication rounds or additional storage costs; however, it is limited in the depth of the computation and the supportable operations such as compare, which is necessary for ReLU.

### C. Delphi overview

Delphi [2] is a representative hybrid PI protocol based on the MPC primitives and HE. Delphi exploits LHE for the linear layers of an ML model and either GC or BT for the activation functions of the model. The inference under the Delphi protocol consists of offline and online phases (see Figure 5 in Appendix A). The offline phase is independent of the client’s actual input data and thus can be pre-processed. The online phase begins when the client sends their private data to the server and ends when the client learns the final inference results.

<sup>1</sup>Oblivious Transfer (OT) is also a type of MPC primitives. OT allows the sender to transfer only one of multiple possible data without knowing what has been transferred to the receiver.

**Linear layer:** During the offline phase, random matrices ( $r_i$  and  $s_i$  for  $i^{th}$  layer at the client and server) are generated. Using the encrypted  $\mathbb{E}(r_i)$  from the client, the server computes  $\mathbb{E}(W_i r_i - s_i)$  using the HE and returns it. The client decrypts and acquires  $W_i r_i - s_i$ . When the online phase starts, the server computes the plaintext operation using  $x_i - r_i$  (SS of input  $x_i$ ) to obtain  $W_i(x_i - r_i) + s_i$ , which results in the SS of  $W_i x_i$  between the client and the server.

**Activation layer:** Cryptographic primitives used for the activation layer depend on the type of the activation function; GC for ReLU or BT for polynomial activation, quadratic approximation in particular. In the case of i) *ReLU and GC*, the server acts as a garbler and the client acts as an evaluator. During the offline phase, the garbled circuit  $\hat{C}$  for ReLU, labels  $L(r_{i+1})$ , and  $L(W_i r_i - s_i)$  are sent from the server to the client. At the online phase,  $L(W_i(x_i - r_i) + s_i)$  is sent from the server to the client, where  $\hat{C}$  is evaluated and generates  $x_{i+1} - r_{i+1}$ , sending it back to the server. In contrast, for ii) *quadratic approximation and BT*, the server and client simply secret-share Beaver’s triples during the offline phase. At the online phase, they jointly compute and gain the secret share of  $x_{i+1}$  consuming a set of triples;  $\langle x_{i+1} \rangle_1$  and  $\langle x_{i+1} \rangle_2$  for the client and server. The client then sends  $\langle x_{i+1} \rangle_1 - r_{i+1}$  to the server, allowing the server to gain  $x_{i+1} - r_{i+1}$  for the next linear layer in the online phase.

Delphi employs NAS [23] to search for an optimal ML model that replaces some of its GC-processed ReLUs with BT-processed quadratic approximations, considering the tradeoff between the model accuracy and costs, including the online serving latency. The online latency is dominated by the activation layers in particular (see Figure 1). While the linear layer only requires fast plaintext computation, the activation layer demands high communication costs between the client and the server, which are especially higher in the case of GC compared to BT. However, one cannot naïvely replace ReLUs (processed via GC) with polynomial activations (processed via BT) to reduce the online latency because the accuracy of an ML model easily deteriorates when the ReLU functions are replaced by simple quadratic approximation functions. Delphi attempts to find a sweet spot amongst this tradeoff exploiting NAS.

Although promising, the latency of the obtained ML models still shows large room for improvements (taking 11.3 secs for inferring ResNet18 on CIFAR-10 in our setup) compared to 0.1354 msec [41] in a non-PI context. Multiple works including [22], [42]–[44] further optimized the training method, NAS structure, or even the polynomial activation function options, but these works have still achieved limited success in the latency or accuracy.

### D. Activation Function Approximation

Various prior arts have attempted to provide effective polynomial activation functions [10]–[17]. However, they either suffer from the inferior *accuracy* of the model or the high online service *latency* in the PI context. Especially when using low degree polynomials (2 or 3), they can only train shallow

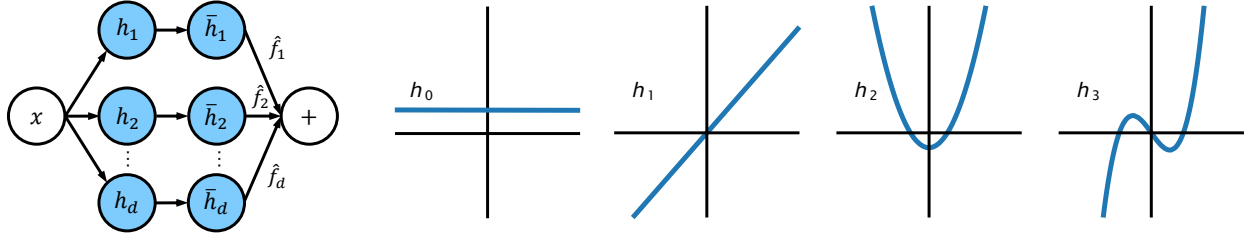


Fig. 2. HerPN block. For one input feature, several Hermite basis polynomials of input are evaluated. The computed Hermite polynomials experience separate basis-wise normalization and weighted summation using Hermite coefficients (left). The first four Hermite basis polynomials are depicted in the right.

neural networks consisting of fewer than ten layers. [11] proposed a QuaIL method that trains a polynomial activation-based model by optimizing the mean squared error (MSE) of an intermediate result from a pretrained ReLU-based model and report training deep networks using degree 2 polynomials. However, the reported accuracy is noticeably inferior to that of neural networks using ReLU. Contrarily, [17] proposed a high degree (e.g., 29) minimax polynomial approximation of ReLU and max-pooling with minimal accuracy drop for ImageNet. However, high degree polynomials are costly to evaluate [17] on the PI framework with higher communication and computational costs.

Circa [19] refactored the ReLU function into a piece-wise linear function and a sign function, utilizing BT for the former and GC for the latter. In particular, Circa further approximated the sign function with stochastic sign and trimming of the input, through which it minimized the size and thus cost of the boolean circuit to be evaluated via GC. Although Circa demonstrated superior (lower) latency over prior studies with comparable serving accuracy, there still exists large room for improvements.

### III. HERMITE POLYNOMIAL WITH BASIS-WISE NORMALIZATION

#### A. Orthogonal basis and Hermite polynomials

Our work utilizes Hermite expansion, a Fourier transform using Hermite polynomials as eigen-functions. Before we go deep into the construction of our activation function, we briefly describe the basic properties of orthogonal bases and Hermite polynomials. For a real interval  $[a, b]$ , let  $L^2([a, b], w(x))$  be the space of square-integrable functions with respect to a weight function  $w(x)$ . Square-integrable function is  $f : \mathbb{R} \rightarrow \mathbb{C}$  such that

$$\int_a^b |f(x)|^2 w(x) dx < \infty \quad (1)$$

Then  $L^2([a, b], w(x))$  is a Hilbert space where the inner product is defined. The inner product of two functions  $f, g \in L^2([a, b], w(x))$  is defined as follows:

$$\langle f, g \rangle = \int_a^b f(x) \overline{g(x)} w(x) dx \quad (2)$$

A finite set of polynomials  $\{p_1(x), p_2(x), \dots, p_n(x)\}$  forms an orthogonal basis if the set spans  $L^2([a, b], w(x))$  and  $\langle p_i, p_j \rangle = 0$  if  $i \neq j$ .

Hermite polynomials are a type of orthogonal polynomials that arise in probability and physics. While there are other sets of polynomials that form an orthogonal basis such as Chebyshev or Laguerre, Hermite polynomials especially feature i) Gaussian weight functions defined in  $\mathbb{R}$  and ii) orderedness, as the low order parts of the Hermite expansion hold most of the information. Mathematically, probabilist's Hermite polynomials are given by:

$$H_n(x) = (-1)^n e^{\frac{x^2}{2}} \frac{d^n}{dx^n} e^{-\frac{x^2}{2}} \quad (3)$$

Then, the normalized Hermite polynomials  $h_n(x) = \frac{1}{\sqrt{n!}} H_n(x)$  form an orthonormal basis in the  $L^2(\mathbb{R}, e^{-x^2/2})$  in the sense that the series of the polynomial  $\{h_i\}_{i=0}^\infty$  are orthonormal:  $\langle h_i, h_j \rangle = \delta_{i,j}$ . Here  $\delta_{i,j} = 1$  if  $i = j$  and otherwise  $\delta_{i,j} = 0$ .

Finally, given a function  $f \in L^2(\mathbb{R}, e^{-x^2/2})$ , we have the Hermite expansion defined as the following:

$$f(x) = \sum_{i=0}^{\infty} \hat{f}_i h_i(x), \quad \hat{f}_i = \langle f, h_i \rangle \quad (4)$$

$\hat{f}_i$  is the  $i$ -th Hermite coefficient of  $f$ , defined as the inner product of the function  $f$  and the Hermite polynomial  $h_i$ .

While some prior works exploited Hermite polynomials in the neural networks, none had sought the use in the context of PI. [25]–[27] exploited the Hermite expansion of activation functions in a theoretical domain. [24] demonstrated that the Hermite expansion of ReLU with a soft-sign function shows fast training loss convergence for a pseudo labeling task.

#### B. The HerPN method

We propose the Hermite expansion of ReLU with basis-wise normalization (*HerPN*) block (see Figure 2) as a substitute for the ReLU and the normalization functions in neural networks. The  $n$ -th Hermite coefficient of the ReLU,  $\hat{f}_n$  is:

$$\hat{f}_n = \begin{cases} \frac{1}{\sqrt{2\pi}} & n = 0, \\ \frac{1}{2} & n = 1, \\ 0 & n \geq 2 \text{ and odd}, \\ \frac{((n-3)!!)^2}{\sqrt{2\pi n!}} & n \geq 2 \text{ and even} \end{cases}$$

Based on Formula 4, we use three bases with the highest degree of 2 ( $h_0$ ,  $h_1$ , and  $h_2$  in Figure 2) for the HerPN block. Using four bases shows similar or slightly lower accuracy compared to the case of using three.

TABLE II

AESPA ONLINE LATENCY AND TOTAL COMMUNICATION COST COMPARISON WITH THE BASELINE NEURAL NETWORKS IN DELPHI FOR CIFAR-100 (C100) AND TINYIMAGENET (TINY). AS THE NETWORKS ARE EQUALLY STRUCTURED WHILE PROCESSING CIFAR-10 AND CIFAR-100 EXCEPT LAST FULLY CONNECTED LAYER, WE ONLY DEMONSTRATE CIFAR-100 BETWEEN THE TWO. FULL ACCURACY RESULTS IN APPENDIX C.

Network	Accuracy(%)		Online Latency (ms)			Communication (MB)		
	Baseline	AESPA	Delphi	AESPA	Improv	Delphi	AESPA	Improv
VGG16-C100	73.45	71.99	6140.1	209.3	29.3×	5782.3	240.9	24.0×
ResNet18-C100	77.93	77.40	12281.8	291.6	42.1×	11699.5	533.8	21.9×
ResNet32-C100	71.66	63.98	6604.7	386.0	17.1×	6403.8	328.4	19.5×
PA-ResNet18-C100	76.95	76.31	12205.3	405.1	30.1×	11533.4	532.0	21.7×
PA-ResNet32-C100	70.21	67.85	7443.9	373.1	23.4×	7349.1	371.4	19.8×
VGG16-Tiny	60.80	58.84	24232.5	505.7	47.9×	22960.4	793.4	28.9×
ResNet18-Tiny	63.72	63.35	48887.4	814.8	60.0×	46685.7	2658.2	17.6×
ResNet32-Tiny	55.06	43.04	26320.9	621.7	42.3×	25408.8	1446.4	17.6×
PA-ResNet18-Tiny	61.95	61.50	48587.5	791.9	61.4×	46021.4	2015.8	22.8×
PA-ResNet32-Tiny	55.58	53.09	30535.8	647.1	47.2×	29189.9	1275.9	22.9×

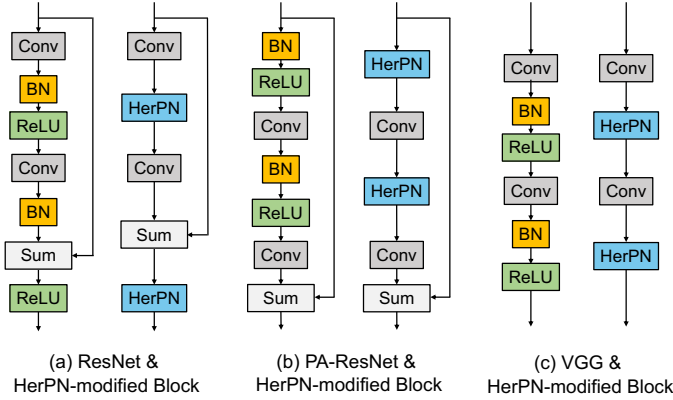


Fig. 3. ResNet, PA-ResNet, and VGG ML architectures that are modified using HerPN blocks.

We also employ basis-wise normalization concatenated to each basis, instead of a standard pre/post-activation normalization. Basis-wise normalization computes the mean and variance of each Hermite polynomial over the mini-batch of training data. We use Hermite coefficients as fixed weight and place scale and shift parameters after accumulating the Hermite polynomials. To sum up, HerPN can be computed on input  $x$  as follows:

$$f(x) = \gamma \sum_{i=0}^d \hat{f}_i \frac{h_i(x) - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \quad (5)$$

Basis-wise normalization is critical in attaining the desired accuracy of the model for the following reasons. First, without a proper normalization technique, the output of the polynomial activation during the forward or backward propagation can exponentially increase. Thus, the ranges of a mini-batch in each layer can change drastically. Such a characteristic exacerbates with the high-degree polynomials or deep neural networks, potentially leading to the exploding or vanishing gradient problem [10], [11].

Second, without the basis-wise normalization, the post-activation value may be dominated by a single highest or lowest order Hermite basis due to the differences in the size of values from the polynomial degrees. The standard post- or pre-

activation normalization does not sufficiently alleviate these problems as they overlook the scale difference between the basis expressions. Basis-wise normalization is advantageous as it keeps each basis zero-centered and consistently keeps the range of the intermediate values throughout the layers.

Using the HerPN block, we can directly replace the batch-normalization (BN) and ReLU blocks in the popular ML architectures such as pre-activation ResNet (PA-ResNet [29]) and VGGNet [30] (Figure 3(b), (c)). We cannot directly apply HerPN to ResNet [28] because a ReLU function exists after the skip-connection. We thus modify the ResNet model as shown in Figure 3(a). Although we only demonstrate the HerPN block based on the Hermite expansion of the ReLU function, our method can also be applied to other activation functions, such as ELU [45], SELU [46], Swish [47], GeLU [48], and Mish [49].

## IV. EVALUATION

### A. Experimental setup

**Benchmarks and datasets:** We tested the private inference of our trained neural networks on CIFAR-10/100 [20] and TinyImageNet [31] datasets. CIFAR-10/100 (C10/100) datasets consist of  $32 \times 32$  50,000 training and 10,000 test images. The only difference between the two datasets is the number of classes (10 for C10 and 100 for C100). TinyImageNet dataset has higher resolution of  $64 \times 64$  and consists of 100,000 training images and 10,000 test images with 200 output classes.

We performed experiments on VGG16, ResNet18/32, and PA-ResNet18/32. We applied our HerPN method to these networks and tailored them to the evaluated datasets by adjusting the pooling or fully-connected layers prior to the final softmax as [11]. We first compared the performance of our HerPN-based ML models with the baseline Delphi (Section IV-B). We also compared our HerPN-based models with the prior polynomial activation works of Lee et.al. [17] and QuaIL [11] using CIFAR-10 (Section IV-C).

**System setup:** We used the AWS system to evaluate the effectiveness of our HerPN method on Delphi, obtaining the latency and communication/storage usage measurements. The

hardware configurations of the client and the server are both AWS c5.4xlarge instances for CIFAR-10/100, which consist of Intel Xeon 8000 series CPU at 3.0 GHz with 32 GB of RAM. We further employed AWS c5.9xlarge instances equipped with the same CPU, each having a 72GB of RAM for TinyImageNet. The client and server instances were both located in the ap-northeast-2 regions. The communication link between the client and server was in the LAN setting. We used the open-source Delphi version using the SEAL library [50] for HE and the fancy garbling [51] for GC. We report the average latency and communication cost results of 10 different experiments, which shows less than 5% fluctuation in their results.

### B. Effectiveness of the HerPN-based models on the Delphi protocol

ML models with HerPN blocks demonstrate superior online latency (ranging from  $17.1\times$  to  $61.4\times$ ) and communication costs (ranging from  $17.6\times$  to  $28.9\times$ ) for all the evaluated ML models compared to the baseline Delphi (see Table II). AESPA shows large runtime improvement over the prior PI protocols. SAFENet [22], which further employed an additional channel-wise NAS, demonstrated  $2.5\times/2.46\times$  reduction in online latency for ResNet32/C100 and VGG16/C10 over the Delphi protocol. Circa, a state-of-the-art protocol that utilized stochastic ReLU, demonstrated  $2.6\times/2.6\times$  runtime improvement for the same ResNet32/C100 and VGG16/C10, while those of AESPA are  $17.1\times$  and  $29.3\times$  over Delphi.

This large reduction of costs is primarily attributed to the fact that HerPN block can replace all ReLUs in the original Delphi, allowing the full usage of the BT instead using the costly GC primitive. Although we only evaluated the HerPN-modified neural networks on Delphi, AESPA are applicable to other PI protocols [3], [10] that are limited by activation functions.

### C. Comparison with the prior activation functions

ML models with the HerPN blocks show accuracies comparable to those of the baseline neural networks with ReLU. Table II reports the accuracy of the trained baseline and HerPN-based models (Appendix C shows the full result). In the case of VGG16, ResNet18 and PA-ResNet18/32, our work shows comparable accuracy. HerPN-based ResNet32 shows a large accuracy drop on CIFAR-100 and TinyImageNet because HerPN cannot natively support the original ResNet with the skip connection. However, such drawbacks can be nullified considering that the accuracy of the HerPN-based PA-ResNet32 is higher than that of the original ResNet32.

Table III compares the modified VGGNet and ResNet implemented by the prior polynomial activation works and HerPN on CIFAR-10. The accuracies of the HerPN-based models are consistently higher than QuaIL, which uses the same degree of quadratic polynomials. Compared to [17], as the degree of polynomials is far smaller, our work has significantly less computational cost on activation functions.

TABLE III  
ML MODEL ACCURACY AND DEGREE COMPARISON OF AESPA WITH PRIOR POLYNOMIAL ACTIVATION WORKS ON CIFAR-10.

Method	Neural Network	Accuracy	Degree
AESPA	VGG16	92.38%	2
	ResNet18	94.96%	2
	ResNet32	93.83%	2
[17]	VGG16	91.87%	29
	ResNet32	89.23%	29
QuaIL	VGG16	82.25%	2
	ResNet18	83.61%	2
	ResNet32	71.81%	2

Yet, AESPA shows accuracy consistently higher than that of [17] on CIFAR-10.

### D. Effectiveness of basis-wise normalization

Through an ablation study regarding the normalization technique for the HerPN block, we identified that our basis-wise normalization shows a clear advantage in terms of model accuracy over other options. Besides the baseline ResNet32 with ReLU activation and the HerPN-based ResNet32, we trained two additional ResNet32 models that exploit the Hermite expansion of the ReLU as the activation function, but adopt standard i) *pre-activation* normalization and ii) *post-activation* normalization, instead of our basis-wise normalization. Figure 4 illustrates the loss value, train accuracy, and test accuracy of the four trained models. In the case of the pre-activation normalization, the loss does not converge to zero and fluctuates during the training. While the post-activation normalization does achieve high train accuracy, the HerPN-based case shows much higher *test* accuracy with a minimum accuracy drop compared to the baseline case with ReLU.

## V. RELATED WORK

### A. Reducing the serving cost of activation functions for PI protocols

Gazelle [3] solely relies on GC and additive masking in calculating the non-linear functions. Delphi reduces the number of ReLUs relying on GC via NAS. Given the pre-trained neural network, Delphi replaces some ReLUs with quadratic approximation and retrain the neural network, which allows them to utilize a less costly BT primitive. The actual replacement process exploits a planner conducting population-based training (PBT [52]) that maximizes the number of ReLU functions to be replaced with quadratic approximation and minimizes the accuracy drop. SAFENet [22] extends the replacement policy of Delphi. SAFENet introduces a more fine-grained channel-wise replacement that picks activation between zero pruning and polynomials of degree 2 or 3 for each channel.

The key weakness of these replacement-based approaches is that even with the NAS-based planner, polynomial approximation results in a severe accuracy drop after a certain approximation ratio. CIFAR-100 accuracy of ResNet32 using Delphi’s planner provides less than a 2% accuracy drop within



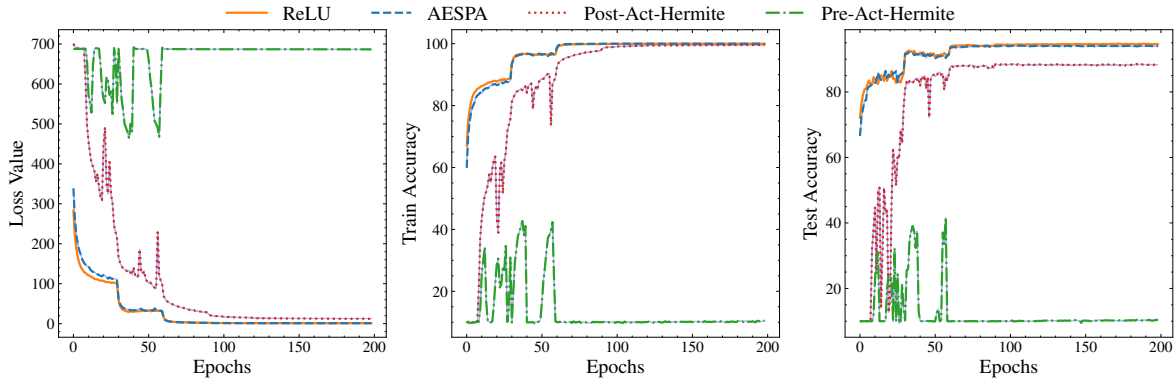


Fig. 4. Comparing ML models on CIFAR-10 using different normalization techniques: i) baseline original ResNet32 with ReLU, ii) HerPN-based model, and ResNet32 with Hermite polynomial activation but with iii) post-act normalization, and iv) pre-act normalization.

26 ReLU function approximations out of 31 ReLU functions. However, the accuracy of a network applying approximation to 27 or more ReLU functions drops quickly compared to the network without approximation. Considering that even a small number of ReLU activation layers with GC incur the majority of the overall cost, the fact that not all activation layers can be approximated is a critical bottleneck.

Another approach is to design neural architectures from scratch optimized for the ReLU counts. [44] searches for neural architectures that can efficiently minimize the number of invoking ReLUs, rather than total FLOPs in typical NAS [23]. However, despite using these optimizations, the remaining ReLUs account for most of the online latency.

### B. Polynomial activation function works

Since CryptoNet [10], there have been challenges to search for polynomial activations for PI. [12]–[17] exploit classical numerical approximations to implement polynomials that can act as the original activation function. The resulting polynomial activations are based on Taylor, Chebyshev, or MinMax approximations. These works have demonstrated superior accuracy to the square function. However, only two of them demonstrated the effectiveness of their work on deep neural networks.

First, [17] approximates a ReLU function with a high degree polynomial that successfully preserves the original model accuracy and proved its trainability on deep neural networks such as ResNet50 [29] and ImageNet [21]. However, the high degree of the polynomial leads to longer latency on the hybrid PI protocol framework even with the BT because it necessitates a greater number of MPC primitive usages.

[11] uses quadratic polynomial approximations and introduces QuaIL. QuaIL is an alternative training method that minimizes the MSE loss between the intermediate representations of a pre-trained ReLU-based neural network and a target quadratic approximation-based neural network. With QuaIL, one can avoid exploding-gradient problems as they train a single layer at a moment while the others are frozen. However, QuaIL failed to achieve competitive accuracy because even

small differences in the intermediate representation can cause divergence in subsequent layers.

Each study showed the potential of polynomials as activation functions. However, there has been no prior work demonstrating accuracy competitive to the baseline ReLU using low-degree polynomials.

## VI. CONCLUSION

In this paper, we have proposed an accuracy preserving low-degree polynomial activation function, AESPA. AESPA leverages the Hermite polynomial concatenated with basis-wise normalization, which retains the high accuracy of the standard ML models using ReLU. Our proposal is especially effective in the PI context because i) the polynomial activation enables less expensive BT primitive instead of GC, ii) the low degree of our work reduces the number of MPC primitives required for serving, and iii) AESPA demonstrates a high ML model accuracy even without the need of complex NAS or training methods. We also showcase the effectiveness of our work with the open-source PI protocol, Delphi, on a real machine and network for VGGNet, ResNet, and pre-activation ResNet; the online latency and the communication between the client and the server are reduced by up to  $61.4\times$  and  $28.9\times$ , respectively.

## REFERENCES

- [1] J. Liu, M. Juuti, Y. Lu, and N. Asokan, “Oblivious neural network predictions via minion transformations,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, 2017. [Online]. Available: <https://doi.org/10.1145/3133956.3134056>
- [2] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, “Delphi: A cryptographic inference service for neural networks,” in *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, 2020. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity20/presentation/mishra>
- [3] C. Juvekar, V. Vaikuntanathan, and A. P. Chandrakasan, “GAZELLE: A low latency framework for secure neural network inference,” in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, 2018. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar>
- [4] A. C. Yao, “Protocols for secure computations (extended abstract),” in *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*. IEEE Computer Society, 1982. [Online]. Available: <https://doi.org/10.1109/SFCS.1982.38>

- [5] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, 2009. [Online]. Available: <https://doi.org/10.1145/1536414.1536440>
- [6] N. Agrawal, A. S. Shamsabadi, M. J. Kusner, and A. Gascón, "QUOTIENT: two-party secure neural network training and prediction," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, 2019. [Online]. Available: <https://doi.org/10.1145/3319535.3339819>
- [7] M. S. Riaz, M. Samragh, H. Chen, K. Laine, K. E. Lauter, and F. Koushanfar, "XONN: xnor-based oblivious deep neural network inference," in *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, 2019. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity19/presentation/riazi>
- [8] A. C. Yao, "How to generate and exchange secrets (extended abstract)," in *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, 1986. [Online]. Available: <https://doi.org/10.1109/SFCS.1986.25>
- [9] D. Beaver, "Precomputing oblivious transfer," in *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, vol. 963, 1995. [Online]. Available: [https://doi.org/10.1007/3-540-44750-4\\_8](https://doi.org/10.1007/3-540-44750-4_8)
- [10] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. E. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, vol. 48, 2016. [Online]. Available: <http://proceedings.mlr.press/v48/gilad-bachrach16.html>
- [11] K. Garimella, N. K. Jha, and B. Reagen, "Sisyphus: A Cautionary Tale of Using Low-Degree Polynomial Activations in Privacy-Preserving Deep Learning," *CoRR*, vol. abs/2107.12342, 2021. [Online]. Available: <https://arxiv.org/abs/2107.12342>
- [12] T. Ishiyama, T. Suzuki, and H. Yamana, "Highly accurate CNN inference using approximate activation functions over homomorphic encryption," in *2020 IEEE International Conference on Big Data (IEEE BigData 2020), Atlanta, GA, USA, December 10-13, 2020*, 2020. [Online]. Available: <https://doi.org/10.1109/BigData50022.2020.9378372>
- [13] S. Obla, X. Gong, A. Aloufi, P. Hu, and D. Takabi, "Effective activation functions for homomorphic evaluation of deep neural networks," *IEEE Access*, vol. 8, 2020. [Online]. Available: <https://doi.org/10.1109/ACCESS.2020.3017436>
- [14] E. Hesamifard, H. Takabi, and M. Ghasemi, "Deep neural networks classification over encrypted data," in *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy, CODASPY 2019, Richardson, TX, USA, March 25-27, 2019*, 2019. [Online]. Available: <https://doi.org/10.1145/3292006.3300044>
- [15] P. Thaine, S. Gorbunov, and G. Penn, "Efficient evaluation of activation functions over encrypted data," in *2019 IEEE Security and Privacy Workshops, SP Workshops 2019, San Francisco, CA, USA, May 19-23, 2019*, 2019. [Online]. Available: <https://doi.org/10.1109/SPW.2019.00022>
- [16] Q. Lou and L. Jiang, "HEMET: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, 2021*. [Online]. Available: <http://proceedings.mlr.press/v139/lou21a.html>
- [17] J. Lee, E. Lee, J. Lee, Y. Kim, Y. Kim, and J. No, "Precise approximation of convolutional neural networks for homomorphically encrypted data," *CoRR*, vol. abs/2105.10879, 2021. [Online]. Available: <https://arxiv.org/abs/2105.10879>
- [18] L. Veidinger, "On the numerical determination of the best approximations in the chebyshev sense," *Numerische Mathematik*, vol. 2, no. 1, 1960.
- [19] Z. Ghodsi, N. K. Jha, B. Reagen, and S. Garg, "Circa: Stochastic relus for private deep learning," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.
- [20] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
- [21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, 2015. [Online]. Available: <https://doi.org/10.1007/s11263-015-0816-y>
- [22] Q. Lou, Y. Shen, H. Jin, and L. Jiang, "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*, 2021. [Online]. Available: <https://openreview.net/forum?id=Cz3dbFm5u->
- [23] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. [Online]. Available: <https://openreview.net/forum?id=r1Ue8Hcxg>
- [24] V. S. Lokhande, S. Tasneeyapant, A. Venkatesh, S. N. Ravi, and V. Singh, "Generating accurate pseudo-labels in semi-supervised learning and avoiding overconfident predictions via hermite polynomial activations," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 2020, pp. 11 432–11 440.
- [25] R. Ge, J. D. Lee, and T. Ma, "Learning one-hidden-layer neural networks with landscape design," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: <https://openreview.net/forum?id=BkwHObbRZ>
- [26] N. Agarwal, P. Awasthi, and S. Kale, "A deep conditioning treatment of neural networks," in *Algorithmic Learning Theory, 16-19 March 2021, Virtual Conference, Worldwide*, ser. Proceedings of Machine Learning Research, vol. 132, 2021. [Online]. Available: <http://proceedings.mlr.press/v132/agarwal21b.html>
- [27] A. Panigrahi, A. Shetty, and N. Goyal, "Effect of activation functions on the training of overparametrized neural nets," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020. [Online]. Available: <https://openreview.net/forum?id=rkgfdeBYvH>
- [28] K. He, X. Zhang, S. Ren, and J. Sun, "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*, 2016. [Online]. Available: <https://doi.org/10.1109/CVPR.2016.90>
- [29] —, "Identity mappings in deep residual networks," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, vol. 9908, 2016. [Online]. Available: [https://doi.org/10.1007/978-3-319-46493-0\\_38](https://doi.org/10.1007/978-3-319-46493-0_38)
- [30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: <http://arxiv.org/abs/1409.1556>
- [31] L. Yao and J. Miller, "Tiny imagenet classification with convolutional neural networks," *CS 231N*, vol. 2, no. 5, p. 8, 2015.
- [32] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, no. 1, 2020. [Online]. Available: <https://doi.org/10.1007/s00145-019-09319-x>
- [33] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, vol. 10624, 2017. [Online]. Available: [https://doi.org/10.1007/978-3-319-70694-8\\_15](https://doi.org/10.1007/978-3-319-70694-8_15)
- [34] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, 2012. [Online]. Available: <http://eprint.iacr.org/2012/144>
- [35] K. A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and Y. Roselander, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*, 2019. [Online]. Available: <https://proceedings.mlsys.org/book/271.pdf>
- [36] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, 2016. [Online]. Available: <http://arxiv.org/abs/1610.05492>
- [37] V. Costan and S. Devadas, "Intel SGX Explained," *IACR Cryptol. ePrint Arch.*, 2016. [Online]. Available: <http://eprint.iacr.org/2016/086>



- [38] B. Ngabonziza, D. Martin, A. Bailey, H. Cho, and S. Martin, "Trustzone explained: Architectural features and use cases," in *2nd IEEE International Conference on Collaboration and Internet Computing, CIC 2016, Pittsburgh, PA, USA, November 1-3, 2016*. IEEE Computer Society, 2016. [Online]. Available: <https://doi.org/10.1109/CIC.2016.065>
- [39] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, 1979. [Online]. Available: <http://doi.acm.org/10.1145/359168.359176>
- [40] M. O. Rabin, "How to exchange secrets with oblivious transfer," *IACR Cryptol. ePrint Arch.*, 2005. [Online]. Available: <http://eprint.iacr.org/2005/187>
- [41] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré, and M. Zaharia, "Dawnbench: An end-to-end deep learning benchmark and competition," *NIPS ML Systems Workshop*, 2017. [Online]. Available: <https://dawn.cs.stanford.edu/benchmark/papers/nips17-dawnbench.pdf>
- [42] M. Cho, Z. Ghodsi, B. Reagen, S. Garg, and C. Hegde, "Sphynx: Relu-efficient network design for private inference," vol. abs/2106.11755, 2021. [Online]. Available: <https://arxiv.org/abs/2106.11755>
- [43] Z. Ghodsi, A. K. Veldanda, B. Reagen, and S. Garg, "Cryptonas: Private inference on a relu budget," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [44] N. K. Jha, Z. Ghodsi, S. Garg, and B. Reagen, "Deepreduce: Relu reduction for fast private inference," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139, 2021. [Online]. Available: <http://proceedings.mlr.press/v139/jha21a.html>
- [45] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. [Online]. Available: <http://arxiv.org/abs/1511.07289>
- [46] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017.
- [47] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*, 2018. [Online]. Available: <https://openreview.net/forum?id=Hkuq2EkPf>
- [48] D. Hendrycks and K. Gimpel, "Bridging nonlinearities and stochastic regularizers with gaussian error linear units," *CoRR*, vol. abs/1606.08415, 2016. [Online]. Available: <http://arxiv.org/abs/1606.08415>
- [49] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *CoRR*, vol. abs/1908.08681, 2019. [Online]. Available: <http://arxiv.org/abs/1908.08681>
- [50] "Microsoft SEAL (release 3.6)," <https://github.com/Microsoft/SEAL>, Nov. 2020, microsoft Research, Redmond, WA.
- [51] "Swanky," <https://github.com/GaloisInc/swanky>, 2019, galois, Inc, Portland, OR.
- [52] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks," *CoRR*, vol. abs/1711.09846, 2017. [Online]. Available: <http://arxiv.org/abs/1711.09846>
- [53] "Delphi," <https://github.com/mc2-project/delphi>, Aug. 2021, rISELab, UC Berkeley, Berkeley, CA.

APPENDIX A  
AN ILLUSTRATION OF THE DELPHI PROTOCOL

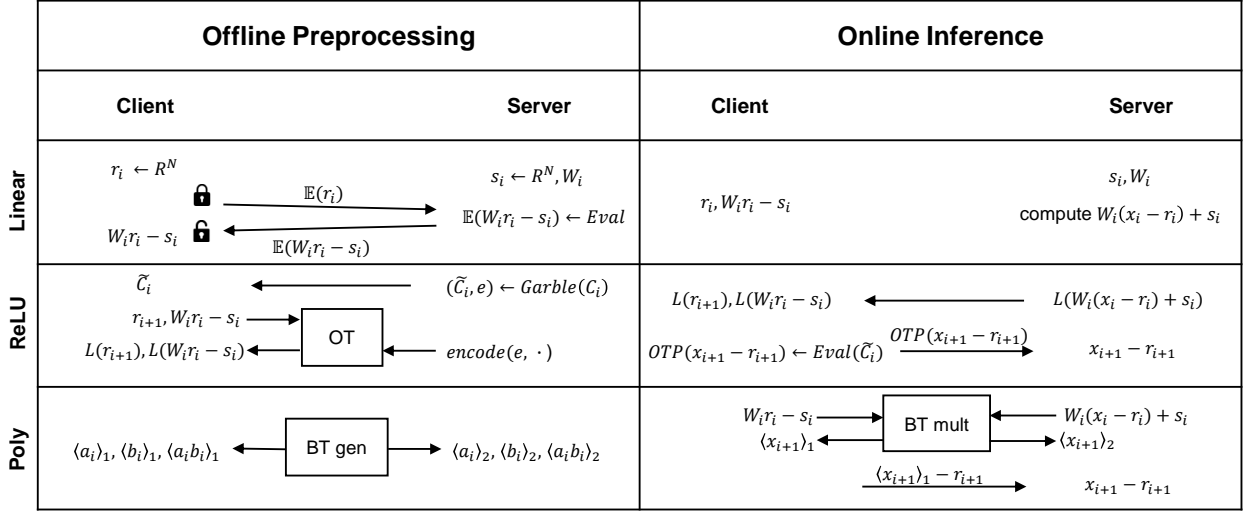


Fig. 5. An illustration of the Delphi protocol. Delphi uses LHE for convolution and FC in linear layers, and GC for ReLU or BT for polynomial activation in non-linear layers.

APPENDIX B  
PARAMETERS FOR CRYPTOGRAPHIC PRIMITIVES

**Choice of prime for prime field:** In Delphi, the underlying MPC cryptographic primitives have two additional parameters: i) *prime number for prime field* and ii) *data representation*. All the operations in the cryptographic primitives are defined on this data representation and prime finite field. This parameter setting ensures no overflow and underflow to occur in a single layer evaluation. Throughout this paper, we have used the latest version of Delphi’s configuration [53], which uses 41-bit prime of 2061584302081 for prime finite fields and 11bit fixed-point data representation.

APPENDIX C  
TRAINING RESULTS

Models presented in the paper are trained normally using PyTorch. More precisely, we trained all networks for 200 epochs using SGD with cosine annealing scheduler, 0.1 initial learning rate, 100 batch size, 0.0005 weight decay, 0.9 momentum. Table IV summarizes training accuracy of AESPA and baseline ReLU networks in CIFAR-10/100 and TinyImageNet. We adopt standard data augmentations: random crops, random flips, random rotation and normalization. Soft labeling is used to obtain higher accuracy.

TABLE IV  
ACCURACY RESULT OF AESPA ON CIFAR-10 (C10), CIFAR-100 (C100) AND TINYIMAGENET (TINY), COMPARED TO THE STANDARD ML MODELS WITH RELU AND BATCH-NORMALIZATION.

Dataset	Neural Network	ReLU Acc	AESPA Acc
C10	VGG16	93.95%	92.38%
	ResNet18	95.57%	94.96%
	ResNet32	93.85%	93.83%
	PA-ResNet18	94.75%	94.59%
	PA-ResNet32	93.21%	91.79%
C100	VGG16	73.45%	71.99%
	ResNet18	77.93%	77.40%
	ResNet32	71.66%	63.98%
	PA-ResNet18	76.95%	76.31%
	PA-ResNet32	70.21%	67.85%
Tiny	VGG16	60.80%	58.84%
	ResNet18	63.72%	63.35%
	ResNet32	55.06%	43.04%
	PA-ResNet18	61.95%	61.50%
	PA-ResNet32	55.58%	53.09%

APPENDIX D  
COMPARISON WITH PRIOR WORK

Table V and VI shows the accuracy and speedup result for VGG16 on CIFAR-10 and ResNet32 on CIFAR-100. While AESPA shows almost an order of magnitude speedup over all prior works, AESPA’s accuracy result is not the highest. However, considering the larger network such as PA-ResNet18 in our experiments, AESPA outperforms all prior works in terms of accuracy and speedup.

TABLE V  
THE VGG16 RESULT ON CIFAR-10.

<b>VGG16</b>	<b>Accuracy</b>	<b>speedup</b>
Delphi	88.1%	1.2×
SAFENet	88.9%	2.5×
Circa	93.8%	2.6×
AESPA	92.4%	24.0×

TABLE VI  
THE RESNET32 RESULT ON CIFAR-100.

<b>ResNet32</b>	<b>Accuracy</b>	<b>speedup</b>
Delphi	67.3%	1.3×
SAFENet	67.5%	2.5×
Circa	66.4%	2.6×
AESPA	64.0%	19.5×