Athena: Accelerating Quantized Convolutional Neural Networks under Fully Homomorphic Encryption

Yinghao Yang
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
yangyinghao@ict.ac.cn

Xicheng Xu
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
xuxicheng20@mails.ucas.ac.cn

Liang Chang
University of Electronic Science and Technology of China
Chengdu, China
Zhongguancun Laboratory
Beijing, China
liangchang@uestc.edu.cn

Hang Lu
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
Zhongguancun Laboratory
Beijing, China
luhang@ict.ac.cn

Xiaowei Li
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
Zhongguancun Laboratory
Beijing, China
lxw@ict.ac.cn

Abstract

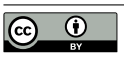
Deep learning under FHE is difficult due to two aspects: (1) formidable amount of ciphertext computations like convolutions, so frequent bootstrapping is inevitable which in turn exacerbates the problem; (2) lack of the support to various non-linear functions in terms of the diversity and accuracy. Previous work primarily used the CKKS-based approach, which requires large parameters and places a heavy burden on the hardware. In this paper, we propose Athena, including a novel framework targeting *quantized* convolutional neural networks under FHE, and a specialized accelerator to release the maximum potential of the framework. Unlike the classic CKKS-based approach, Athena only requires much smaller parameters, i.e., 2^{15} degree and approximately 5 MB ciphertext size. Athena uses a uniform representation, functional bootstrapping, to accurately support any type of activation functions, and is not limited to polynomial approximate fitted functions such as ReLU and sigmoid. We highlight the following results: (1) the accuracy varies by +0.01% /−0.24% compared with the plaintext quantized CNN; (2) the inference performance on the Athena accelerator achieves a speedup of 1.5× to 2.3×, an EDAP improvement of 3.8× to 9.9×, compared with state-of-the-art FHE accelerators.

Keywords

Fully homomorphic encryption (FHE), Quantized convolutional neural networks (CNNs), Accelerator architecture design

ACM Reference Format:
Yinghao Yang, Xicheng Xu, Liang Chang, Hang Lu, and Xiaowei Li. 2025. Athena: Accelerating Quantized Convolutional Neural Networks under Fully Homomorphic Encryption. In *58th IEEE/ACM International Symposium on Microarchitecture (MICRO '25), October 18–22, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3725843.3756103

The corresponding author is Hang Lu (luhang@ict.ac.cn). This work was completed at SKLP, Institute of Computing Technology, Chinese Academy of Sciences.



MICRO '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-1573-0/25/10
https://doi.org/10.1145/3725843.3756103

1 Introduction

The global tech industry recognizes the increasing importance of privacy-enhanced computation for securing personal information. Fully Homomorphic Encryption (FHE), at the forefront of mainstream privacy-enhanced computation, is a powerful methodology that enables computations on encrypted data without accessing sensitive or personal information [6, 11, 14, 40]. Recent research has focused on using FHE in cloud datacenters for machine learning inference. For example, Google has introduced the capability for video deblurring and object detection with FHE-based machine learning; Amazon has integrated FHE with its SageMaker endpoints [3] for the clients to perform secure and real-time inference. Supporting a variety of machine learning models and achieving fast and accurate inference on the encrypted data is highly desirable in the industry today.

Deploying complex Convolutional Neural Networks (CNNs) with large-scale is challenging under FHE due to its intractable computation. The reason lies in two aspects. Firstly, large-scale CNNs usually consist of tens to hundreds of convolution layers. Computing these layers under FHE will require frequent bootstrapping, a.k.a., the most complex and time-consuming operation. That is also why existing solutions only apply to very tiny CNNs [4, 9, 13, 15], because the chosen parameters are just enough to perform the target CNN without bootstrapping. Secondly, CNN is usually composed of substantial non-linear layers in addition to the linear layers. Existing arithmetic-based FHE algorithms like CKKS enables SIMD-like batched operations which is very suitable for the linear operations like convolution and the fully-connected. Non-linear operations such as ReLU, and Sigmoid require approximation methods, such as Taylor or Chebyshev expansions, to estimate their results. Even if the computations of the transformed SIMD-alike operation can be effectively dealt with, the accuracy may degrade because of the

error added by such approximation, especially when the layers of a particular CNN are much deeper.

Speaking of the error, the accuracy after decrypting the result compared with the plaintext CNN is significantly affected by the error. The FHE-based CNN will add error at each individual step during inference, i.e., boostrapping, keyswitch, rotation for the accumulation, etc. If we use the classic CKKS FHE, the accuracy is reflected by the precision of the binary decimals. Along with performing the convolutions and non-linear functions layer by layer, the error incrementally invades the binary decimals and ultimately degrades the precision of the results. The accuracy hence becomes highly unpredictable. Therefore, the parameter setting usually requires a large scaling factor and enough binary decimals to tolerate the precision loss during inference. However as mentioned before, large parameters lead to substantial hardware overhead in terms of on-chip storage and data movement bandwidth.

Numerous previous FHE accelerator designs aim to accommodate such *large-parameter* settings to mitigate the accuracy loss, while simultaneously achieving satisfactory speedup. This design paradigm is inevitably costly due to the considerable amount of chip resources and area required, especially when dealing with highly complex FHE operations like bootstrapping. Specifically, the speedup heavily relies on the on-chip storage capacity, i.e., the unconventional 256MB + 26MB for Craterlake [38]; 512MB and more for ARK [23] and BTS [24], due to the large size of the ciphertext and various public keys. A large on-chip data movement bandwidth is also imperative under such circumstances to provide the matching amounts of operands to the computing cores like NTT and automorphism units, i.e., 84TB/s for Craterlake, 92TB/s for ARK, and 330TB/s for BTS, as reported in the literature. HEAP [1] proposes an idea to accelerate CKKS bootstrapping across multiple FPGA cards using TFHE's programmable bootstrapping (PBS). However, it essentially trades increased computational overhead for parallelism, requiring more FPGA cards to provide sufficient compute resources. Cheetah [35] and [36] propose accelerator architectures for BFV FHE, primarily focusing on improving BFV efficiency without considering parameter settings in CNN implementations, thus still facing challenges similar to CKKS-based accelerators. Moreover, CNNs impose even more severe burdens on bootstrapping and non-linear functions. ResNet-20 [27, 28] requires bootstrapping at least twice per residue block and more than 19 times in total; ResNet-56 [27] is even deeper and more computationally complex compared to ResNet-20. Existing FHE accelerators find it challenging (or even impossible) to further enhance performance by simply adding more scratchpad resources, even slightly, making the deployment of CNNs under FHE a daunting task.

We propose Athena, a low-cost, fast, and accurate methodology for FHE-based quantized CNN acceleration. Athena is a universal acceleration framework designed to adapt to various model parameters and nonlinear functions, supporting a wide range of CNNs. Athena abandons the de-facto design paradigm of using the cryptosystem for CKKS-based FHE in floating-point CNN inference with *large-parameter* settings for bootstrapping and long multiplicative depth, by targeting quantized CNN inference under FHE. It accommodates various quantization precision and can flexibly adjust the mapping space of non-linear function to optimize system performance. Athena co-designs the software and hardware, encompassing the Athena framework with a five-step consecutive loop on the software side, and the Athena accelerator. In terms of architecture, Athena's inference framework is closely aligned with hardware architecture, focusing on reducing encryption parameters for better compatibility, which reduces the scratchpad capacity requirement by over 4×. We also design the "FBS and RNS Base changing unit" (FRU) and specific dataflow to support the computational bottlenecks. Quantization is a widely used and mature methodology. Despite its comparable (or sometimes even higher [31]) accuracy to its floating-point counterpart, it is very compatible with FHE, as the input activation, model weights, and their arithmetic are all integer-based. Thus, it provides a favorable condition for re-architecting the inference mechanism in an accuracy-controllable yet cost-effective manner with much *smaller* parameters (as detailed in Section 5.2). The contributions of this paper are listed as follows:

(1) *The Athena framework for the light-weight, lossless deep learning under FHE.* It utilizes the coefficient encoding-based optimization mechanism for efficient computation of linear layers. Additionally, it incorporates a noise control mechanism that includes modulus switching, ciphertext conversion (from RLWE to LWE and back), and a functional bootstrapping (FBS) mechanism that enables batch processing of precise non-linear functions and remapping. In theory, the Athena framework can support any non-linear function in CNNs, as detailed in Section 3.2.3, and can easily accommodate various models. The accuracy obtained using Athena is 94.65% and 94.66% in ciphertext for ResNet-56, compared with 94.89% and 94.79% in the plaintext quantized ResNet-56.

(2) *The Athena accelerator - a cost-effective hardware implementation specialized for the Athena framework.* The Athena accelerator is specifically designed for the Athena framework, featuring significantly smaller ciphertext degrees (and therefore smaller ciphertext sizes), public key sizes, and ciphertext moduli, leading to a 4× reduction in scratchpad capacity. We also design the Sample Extraction (SE) unit to effectively support the conversion of ciphertext formats within the Athena framework and instantiate customized modules FRU designed to reuse many computational cores through different data paths, addressing Athena's computational bottlenecks. Based on the versatile FRU and the design of the computational dataflow, the Athena accelerator can well match the inference framework to achieve excellent performance. We highlight the following results: 1.5× to 2.3× speedup compared with the state-of-the-art accelerators, together with 2.4× to 6.2× EDP improvement and over 1.53× area reduction.

2 CNNs under FHE

2.1 Challenges

2.1.1 Large-parameter Setting. In FHE, the parameter selection presents a tradeoff with implementation overhead. Table 1 compares six solutions from the literature. Early-stage pure leveled homomorphic encryption (LHE) [9, 13, 15] uses a small degree, such as 8192, and the ciphertext modulus, Q, never exceeds 220 bits. This conservative setting results in a solution incapable of bootstrapping, limiting its application to small-scale deep learning, such as CNNs with three or fewer layers [4, 26] trained on the MINIST dataset. More aggressive solutions [27, 28] use a larger degree (i.e.,

Table 1: Solutions for CNN under FHE. `L`: Linear; `NL`: Non-Linear; `E`: Evaluation; `B`: Bootstrapping; `Q`: Quantized; `NQ`: Non-Quantized; `FBS`: Functional Bootstrapping. We use the multiplicative depth to denote `E` and `B` levels for the CKKS-based FHE methods. For the LHE-based methods and our Athena, there is no multiplicative depth, which is marked by `NA`.

Method	CNN	Degree	Q / Δ (bits) (E / B) (level)	B & NL	Cipher. size	Key Size (rot+relin)	Dataset	Accuracy (%) (cipher / plain)
YASHE (L) (LHE)[13]	NQ	8192	191 / NA (NA / NA)	Seperated (Taylor)	375 KB	31.5 MB	MNIST	CryptoNets: 98.95 (99.0)
BGV (L) (LHE)[15]	NQ	8192	<220 / NA (NA / NA)	Seperated (Taylor)	448 KB	36.7 MB	MNIST	CryptoDL: 99.5 (99.7)
BFV (L) (LHE)[9]	Q	8192	219 / NA (NA / NA)	Seperated (Taylor)	448 KB	36.7 MB	CIFAR-10	Fast-CryptoNets: 86.76 (93.10)
CKKS (L+NL+B) (FHE)[28]	NQ	65536	1450 / 50 (11 / 13)	Seperated (Taylor)	27 MB	1.9 GB	CIFAR-10	ResNet-20: 92.43 (92.95)
CKKS (L+NL+B) (FHE)[27]	NQ	65536	1501 / 46 (16 / 14)	Seperated (Taylor)	32 MB	2.1 GB	CIFAR-10	ResNet-20: 91.31 (91.52) ResNet-56: 92.80 (93.07)
BFV(L)+FBS(NL+B) (FHE), **ours**	Q	32768	720 / NA **(19)**	Merged **(FBS)**	5.6 MB	720 MB	CIFAR-10	ResNet-20: **93.63 (93.84)** ResNet-56: **94.65 (94.89)**

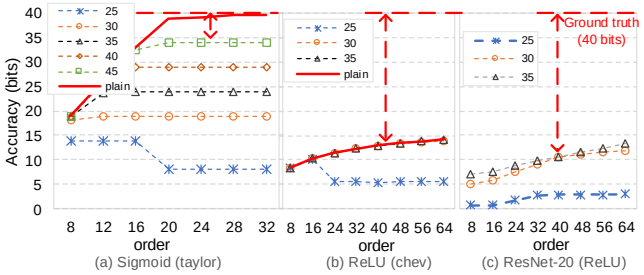


Figure 1: The impact of Δ to the non-linear activation functions and CNN model. The red line denotes the accuracy of the plaintext expansion. Other lines denote the decrypted accuracy of the ciphertext expansion under different Δ settings. X-axis is the orders of the expansion, and Y-axis is the accuracy in bits.

2^{16}) and a long modulus chain, or Q, in CKKS to accommodate normal multiplication and bootstrapping, enabling private inference of more complex CNNs like "ResNet-20". However, the total key size might be up to 50× larger than that of LHE, and one ciphertext might reach tens of megabytes. Several proposed hardware accelerators [22–24, 37, 38], although not specifically designed for deep learning, can support these large-parameter settings and deploy CNNs under FHE, but this increases ASIC chip size and reduces efficiency.

2.1.2 Accuracy. CKKS-based solutions employ series expansion to approximate the actual function value due to the non-linearity of activation functions. As shown in Fig. 1, We adopt commonly used Taylor and Chebyshev approximations of ReLU and Sigmoid as representative nonlinear functions, and use ResNet-20 with ReLU as a CNN benchmark. Bit accuracy is used to measure the difference between the computed results and the ground truth (40-bit) of function values and model output probabilities. The series expansion introduces a certain level of error, denoted by the red line. Ciphertext expansion under different Δ settings shows that more expansion orders (X-axis) improve accuracy, except when Δ = 25. However, there is a significant gap compared to the 40-bit ground truth, which is even larger for ReLU. Moreover, due to the approximation error of ReLU and its propagation across network layers, ResNet-20 shows degraded and unstable accuracy (measured by the average output probability) even when Δ is set to 30 or 35, performing worse than exact ReLU. When Δ = 25, the precision drops to around 2 bits, which is insufficient for practical use in typical CNN models. This is why some CKKS-based approaches set it as large as possible, i.e., 46 bits in [28] and 50 bits in [27]. This illustrates the limitations of the traditional extension-based approach to implementing nonlinear functions, i.e. it requires fine-tuning function design and selection of appropriate parameters, which need to be done by cryptography experts with expertise in cryptography. The lack of a unified design framework increases the complexity of FHE-based CNNs, thereby reducing their ease of use.

2.2 FHE's Affinity to Quantized CNNs

2.2.1 Controllable Accuracy with Small Parameters. Quantization aims to use integer precision for CNN inference while achieving negligible accuracy loss. Unlike [28] and [27], which leverage the Δ in CKKS to control decimal precision and tolerate precision loss due to noise, quantized CNN inherently possesses much lower "integer" precision, usually 8 bits, meaning there are no decimals in the plaintext. Therefore, no Δ setting is necessary, and we only need to ensure the 8-bit integer is accurate after dequantization and decryption, rather than maintaining around 30 bits of decimal accuracy as in the previous approach [27, 28]. This allows parameters such as "degree" and Q to be set much smaller without sacrificing accuracy, as the linear function result reflected in the intrinsic plaintext is an 8-bit integer multiply and accumulate (MAC). As shown in Table 1, our Athena, by targeting quantized CNN in FHE, only requires a 2^{15} degree and 720 bits of Q, but achieves even closer final accuracy compared to its plaintext counterpart. The key and ciphertext size also decrease significantly, to 3~6×, thereby alleviating the hardware burden.

Besides the benefit of small parameters, the MAC of the quantized CNN under FHE has fully controllable accuracy at each step during inference, due to the exclusion of decimals. Some early-stage

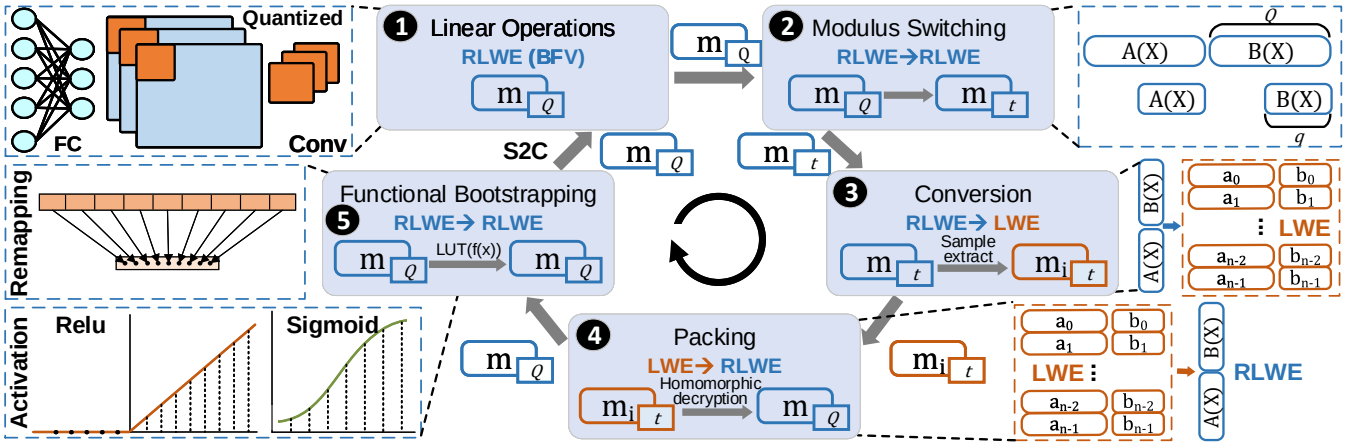


Figure 2: Athena framework. The five-step loop consecutively handles one linear layer (Conv or FC), and one non-linear layer (Activation) fused with remapping and bootstrapping at a time. LUT contains the encrypted activation values.

FHE algorithms, like BFV, which aim at integer-only homomorphic encryption, can be applied. Unlike CKKS, which relies on the approximate result of real numbers, BFV is designed for precise integer results and could be leveraged for encrypting quantized CNNs. A precise MAC contributes to maintaining the final accuracy.

2.2.2 Non-linearity and Bootstrapping Friendly. Non-linear functions and bootstrapping are the key design challenges in FHE-based CNN. CKKS-based solutions [27, 28] utilize the Taylor series expansion to approximate the results of non-linear functions, and the complex sine function to approximate the homomorphic modulo operation in bootstrapping. Some approaches resort to the CKKS/TFHE hybrid method to tackle the large overhead of bootstrapping and the inflexibility when dealing with arbitrary non-linear functions. For example, TOTA [42] leverages TFHE to perform blind rotation on the look-up table (LUT), mapping data to arbitrary function values. To maintain high accuracy, this approach requires multiple TFHE bootstrappings of data segments. Although TFHE bootstrapping is flexible, the possible permutation space of the LUT is proportional to the plaintext precision. For example, PEGASUS [30] uses 45-bit floating point precision, so its LUT has 2^{45} combinations, and the blind rotation requires passing through thousands of CMUX gates to maintain complete precision. Since the computational model of TFHE is SISD, using it to support high-precision activation and bootstrapping for large data in CNNs will introduce a huge computational overhead and significantly impact system performance.

Benefiting from the lower precision in the quantized CNN, the LUT space naturally has a reduced size (less than 2^{17} in Athena). This makes operations relevant to the LUT more cost-effective for hardware implementation and more flexible in supporting any type of non-linear functions, as we only need to encode a limited number of discrete values in the LUT. Additionally, quantization usually works with 8-bit or lower integers by remapping the MAC result back to 8 bits. It is also possible to further reduce the noise while controlling the plaintext within its pre-set modulus. Athena leverages these significant advantages to make a new solution for deep learning under FHE possible. The details will be elaborated in the next section.

3 Athena

3.1 Framework

In the practical industrial use, the weights of a CNN are always kept within the company domain. Therefore, only user privacy data, such as an image or audio fragment, is encrypted by FHE as input, while the weights remain in plaintext form [2, 5, 9, 18]. Athena targets quantized CNN inference under FHE. As a mature methodology in AI research, a considerable amount of work has been proposed for the high-accuracy model quantization [17, 19, 33, 43]. The general-purpose quantization procedure for CNN inference involves three classic steps: quantizing the activations of each layer into 8-bit (or lower) precision with scaling factors, issuing the multiply and accumulations, and dequantizing the results back to 8-bit integers (usually called remapping) [8]. Well-known deep learning frameworks like TensorRT, PyTorch, and Tensorflow have built-in quantization methods that follow this procedure, achieving comparable or slightly higher accuracy than non-quantized versions [31]. Within the scope of Athena, this procedure is issued under FHE. In each inference phase, such as the quantized convolution layer, the non-linear activation layer, and even the remapping, the intermediate activations are strictly kept under cover as FHE ciphertexts, ensuring accurate inference results and absolute user data security.

The Athena framework, as shown in Fig. 2, comprises five tightly coupled steps. Due to the negative impact of the added noise mentioned in the previous section, Athena proposes using the combination of modulus switching (Step ❷), ciphertext conversion (Step ❸ and ❹) to reduce the inherent noise introduced by FHE-specific operations like ciphertext multiplication and addition. To address the encrypted inference of non-linear activation functions, Athena proposes a uniform representation of these functions, a.k.a LUT, and collaboratively accomplishes bootstrapping and the activation function in tandem. This step, illustrated by Step ❺, is termed as functional bootstrapping (FBS). The core of the framework involves

Table 2: Ratios of valid data in the result polynomials under common convolution layer.

$(HW, C_{in}, C_{out}, W_k, stride, padding)$	**Cheetah** [16]	**Athena**
$(32^2, 3, 16, 3, 1, 1)$	25 %	50 %
$(32^2, 16, 16, 3, 1, 1)$	3.13 %	50 %
$(32^2, 16, 32, 1, 2, 0)$	1.56 %	25 %
$(16^2, 32, 32, 3, 1, 1)$	2.27 %	25 %
$(16^2, 32, 64, 1, 2, 0)$	0.78 %	6.25 %
$(8^2, 64, 64, 3, 1, 1)$	0.96 %	12.5 %

converting ciphertexts from RLWE to LWE, homomorphically decrypting, and packing back the LWE ciphertexts into one RLWE ciphertext. The uniform representation by LUT enables Athena to be applicable to any non-linear functions. In FBS, the remapping is merged inside the non-linear functions. In the next section, we delve into the details regarding its implications with the quantized CNN and the control and increment of noise.

3.2 Procedures

3.2.1 Encoding and Linear Functions. The classic encoding method (in "slots") requires a number of high overhead rotation operations for the accumulation in linear layers. To circumvent this issue and prepare for subsequent ciphertext conversion and noise refresh, we employ the coefficient encoding method proposed and utilized in [16, 21] for the implementation of the linear function and adjust the encoding sequence, which allows the result to be distributed compactly across the ciphertexts, minimizing the number of result ciphertexts for efficient implementation of the subsequent steps of the Athena framework (Step ❷ and Step ❸).

To compactly distribute the results of the computation into a small number of result ciphertexts, we need to adjust the encoding of the data. For the general case with input $M \in Z_p^{C_{in}*H*W}$ and kernel $K \in Z_p^{C_{out}*C_{in}*W_k*W_k}$, data location is arranged as $\hat{M}$ and $\hat{K}$ described in Eq. 1, where $T = HW(C_{out}C_{in} - 1) + W(W_k - 1) + W_k - 1$, $c \in [0, C_{in})$, $h \in [0, H)$, $w \in [0, W)$, $c' \in [0, C_{out})$, and $i, j \in [0, W_k)$. C_{in} and C_{out} represent the number of input and output channels, W and H represent the width and height of feature map, and W_k represents the size of the convolutional kernel. Here we assume $C_{out}C_{in}HW \leq N$ for simplicity. This extends the single-channel single-kernel example to multi-channels and multi-kernels of convolution, with results obtained in coefficients of $\hat{M} \cdot \hat{K}$. The FC layer is essentially the inner product operation, similar to convolution, completed by setting $W = W_k = C_{in} = 1$.

$$\begin{aligned} \hat{M}[cHW + hW + w] &= M[c, h, w] \\ \hat{K}[T - c'C_{in}HW - cHW - iW - j] &= K[c', c, i, j] \end{aligned} \tag{1}$$

When $C_{out}C_{in}HW > N$, the entire convolutional layer cannot be completed with a single ciphertext. Therefore, a batching strategy is employed to divide the feature map and convolution kernel into b_i based on the channel dimension. Unlike Cheetah [16], whose encoding aims to minimize computational overhead, Athena prioritizes the arrangement of features and kernels across different C_{out} dimensions, and adaptively selects H' and W' for feature partitioning. Table 2 compares the effective data ratios in the resulting polynomials for both methods under several common convolution parameter sets. Taking the $(32^2, 16, 16, 3, 1, 1)$ parameters as an example, Cheetah prioritizes packing features from 16 input channels into a single ciphertext during encoding. This results in the final output being distributed across 16 ciphertexts. In contrast, Athena's encoding prioritizes packing features corresponding to 16 output channels into a single ciphertext. Although this approach increases the number of ciphertext multiplications and additions, the final computed result is distributed across only one ciphertext, substantially improving the effective data ratio, which can facilitate the subsequent steps of the framework, such as sample extraction (Step ❷ in Fig. 2).

Algorithm 1: Sample Extraction

Input: BFV ciphertext $ct_{BFV} = (A, B)$; A and B are both coefficient vectors.
Result: N LWE ciphertexts $(ct_0, ct_1, \cdots, ct_{N-1})$

```
for i ⟵ 0, ··· , N − 1 do
    Initialize ct_i = (ā_i, b_i);
    for j ⟵ 0, ··· , N − 1 do
        ā_i [j] = j ≤ i ? A [i − j] : −A [N + i − j];
    end
    b_i = B [i];
end
return (ct_0, ct_1, ··· , ct_{N−1})
```

3.2.2 Noise Control and Ciphertext Conversion. Linear functions introduce noise and shrink the noise budget in the ciphertext, affecting subsequent calculations. After each linear layer, noise control is essential for accuracy. In BFV, the plaintext is scaled by the factor Δ during encryption, which means the valid values of the plaintext m_i are located at the MSBs of the ciphertext modulus Q. As shown in Fig. 3, e denotes the noise introduced by the linear functions, and the noise increment has consumed its payload $\Delta/2$. t is the plaintext modulus that frames the inner product of the linear functions within itself. Athena targets the quantized CNN and the inner product result might have much larger precision. Therefore, it requires proper t setting to prevent the inner product precision from exceeding the plaintext modulus itself. From our case study in Section 3.3, setting t to 17 bits is adequate for this purpose. In order to diminish the noise, Athena leverages "modulus switching" as shown in Step ❷ in Fig. 2. It switches the modulus of the ciphertext encrypting the inner product to a smaller modulus. Specifically, it converts the previous BFV ciphertext in Fig. 3 under Q to t, i.e., $ct_{BFV} = (a, b) = (a, -as + m + e_{ms}) \in \mathcal{R}_t^2$ following Eq. 2.

$$\begin{aligned} ct_{BFV} &= (a, b) \in \mathcal{R}_Q^2 \\ ModSwitch(ct_{BFV}, t) &:= (\lfloor a * t/Q \rceil, \lfloor b * t/Q \rceil) \end{aligned} \tag{2}$$

The modulus switching has a twofold impact. On the one hand, it removes the large noise in the Q/t range, refreshing the noise increment space in the ciphertext. On the other hand, it introduces some noise e_{ms} to the plaintext in t due to rounding operation in Eq. 2. Despite potentially affecting accuracy, the Athena framework is not susceptible to it (see Section 5.2 for the accuracy evaluation), as t is set as 17 bits and the noise only contaminates the LSBs in t. Since Athena deals with the quantized model and remaps the

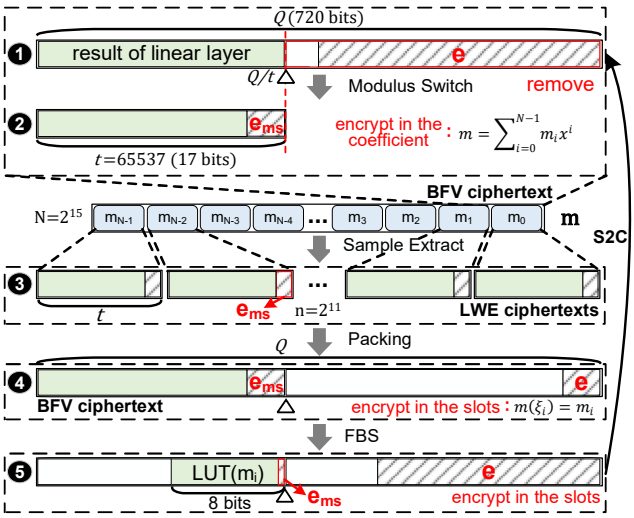


Figure 3: FBS and noise control. The modulus switching introduces e_{ms}, but eliminates the noise e introduced by linear functions. Sample extraction and packing refreshes the noise in the RLWE ciphertext. The headroom is enough for the next round Athena loop.

intermediate inner product back to int8 precision, the contaminated LSBs (1 bit at most) have minimal impact on the final accuracy. The detail will be introduced in Section 3.3.

After modulus switching, the noise is effectively reduced. However, the ciphertext's modulus is now t, much smaller than the original Q, limiting subsequent computations. Step ❸ and ❹ collaboratively resolve this issue. Step ❸ first converts the current BFV ciphertexts from RLWE to LWE form via "sample extraction", as described in Alg. 1. To decrease Step ❹'s overhead, we switch the degree of ciphertext from N to n using the method described in [12] before sample extraction. It is essentially the keyswitching operation. A single LWE ciphertext encrypts only one m_i of the plaintext; an BFV ciphertext corresponds to multiple LWE ciphertexts, as shown in Fig. 2 and Fig. 3. We could formalize the LWE ciphertext as $ct_i(lwe) = (\vec{a}_i, b_i) = (\vec{a}_i, -\vec{a}_i \cdot \vec{s} + m_i + e_i) \ mod \ t$, where $\vec{s}$ is the private key of the LWE ciphertexts, and $i \in [0, N)$. The noise e_{ms} is amortized into each LWE ciphertext as well according to the algorithm. Step ❹ packs the LWE ciphertexts back into one BFV ciphertext in RLWE form, with the large modulus Q for the subsequent non-linear functions. The packing involves the homomorphic decryption of the LWE ciphertext with the packing key encrypted private key s in RLWE form. The operation is a multiplication between the plaintext LWE matrix (a and b of LWE cipher) and the ciphertext vector (packing key) and the Baby-Step Gaint-Step (BSGS) algorithm [7] can be used to reduce the computational complexit. Taking an LWE ciphertext $ct_{lwe} = (\vec{a}, b)$ as an example, $\vec{a} \circ RLWE(s) + b = RLWE(a_0 * s_0 + a_1 * s_1 + \cdots + a_{n-1} * s_{n-1} + b) = RLWE(dec(ct_{lwe}))$, where $\circ$ represents inner product operation. After this step, the data is encrypted in the *slots* of the RLWE ciphertext, i.e. $ct_{BFV} = (a, b) = (a, -as + m + e) \in \mathcal{R}_Q^2$, where $m(\epsilon_i) = m_i$ and ϵ_i is the twiddle factor.

3.2.3 Non-linear Functions and Functional Bootstrapping. Bootstrapping is well-known for its huge overhead , and is inevitable because the vast layers of CNN will consume the multiplicative depth rapidly in, for example, the CKKS-based scheme [27, 28]. In [27], bootstrapping is inserted right after each linear and non-linear functions, and the whole CNN inference attains nearly 0.6 hours (2271s for ResNet-20) on a standard high-end CPU as reported. Athena abandons the pure-CKKS based bootstrapping approach by replacing it with low-cost but accuracy-maintainable "table look-up" proposed in [29], as shown in ❺ in Fig. 2. Function bootstrapping accomplishes three key operations: the non-linear functions, the remapping which is performed in conjunction with the non-linear functions and the bootstrapping itself.

$$FBS(x) = LUT(0) - \sum_{i=0}^{t-1} x^i \sum_{k=0}^{t-1} LUT(k) k^{t-1-i} \tag{3}$$

FBS essentially embeds the required data mapping relationship, i.e., the lookup table (LUT), into a polynomial function using interpolation. This enables nonlinear mapping of encrypted data by evaluating a linear polynomial over ciphertexts. The core of this method is to construct a linear polynomial function, $FBS(x)$, based on a given discrete LUT, which is formally expressed in Eq. 3. Here, t represents the plaintext modulus, defining the mapping space of the LUT. The function $FBS(x)$ takes the form of a polynomial where x^i denotes different power terms, ensuring that $FBS(x) = LUT(x)$. Taking the ReLU function as an example under $t = 5$, the constructed polynomial is computed modulo t, satisfying the equivalences $-2 \equiv 3 \mod 5$ and $-1 \equiv 4 \mod 5$. The LUT mapping is $LUT(0) = ReLU(0) = 0, LUT(1) = 1, LUT(2) = 2, LUT(3) = LUT(-2) = 0, LUT(4) = 0$. Substituting this LUT into Eq. 3 yields the polynomial $FBS(x) = 3x + x^2 + 2x^4$ and $FBS(1) = 1 = LUT(1) \mod 5, FBS(3) = 0 = LUT(3) \mod 5$. Therefore, by computing $FBS(x)$, each ciphertext item m_i becomes $LUT(m_i)$, i.e. $Enc(FBS(m_i)) = Enc(LUT(m_i))$ as shown in Fig. 3.

Athena targets quantized CNNs under FHE, so remapping, which lowers the precision of the plaintext inner product back to quantitative range, is an essential step. Athena merges remapping with the non-linear function as: $LUT(x) = \lfloor Act(x \times scale) \rceil$, enabling simultaneous accomplishment of both the remapping and the non-linear function through one ciphertext evaluation. One of Athena's key features is its support for various non-linear functions, including commonly used ones such as *ReLU, Sigmoid, Gelu*, etc. We discuss some examples of non-linear functions that fit in the Athena framework as follows:

ReLU & Sigmoid alike. For single-input nonlinear functions, we can directly construct the function mapping relationship into a LUT. Taking ReLU as an example, the non-linear function can be performed homomorphically by constructing $LUT(x) = ReLU(x)$ and evaluating it by FBS only once.

Softmax alike. The evaluation of the multiple-inputs function softmax - $e^{x_i}/(\sum_{j=0}^{n-1} e^{x_j})$, is divided into three steps. Step ①: evaluate the function $f(x) = \lfloor e^x \rceil$. Note that in integer-only inference, the result of e^x would exceed the quantization bit widths. Therefore, we replace the LUT with $\lfloor e^x \times scale \rceil$ to keep the bit width within the appropriate range. Step ②: evaluate the inverse function of $scale \times \sum_{j=0}^{n-1} e^{x_j}$. Step ③: perform the ciphertext-ciphertext multiplication (CMult hereafter) to obtain the result $e^{x_i}/(\sum_{j=0}^{n-1} e^{x_j})$.

Table 3: Computational complexity analysis. N: polynomial degree; f: width/height of kernels; C: the number of input/output channels; p and r: degree of linear fit polynomials (ReLU and Boot); t: plaintext modulus.

Solutions	Operations	# of PMult	# of CMult	# of HRot
CKKS -based [27]	Conv	$O(f^2C)$	/	$O(f^2)+O(C)$
	ReLU	$O(p)$	$O(\sqrt{p})$	/
	Bootstrap	$O(\sqrt[3]{N})+O(r)$	$O(\sqrt{r})$	$O(\sqrt[3]{N})$
Athena	Conv	$O(C)$	/	/
	Packing	$O(C)$	/	$O(C)$
	FBS	$O(t)$	$O(\sqrt{t})$	/
	S2C	$O(\sqrt[3]{N})$	/	$O(\sqrt[3]{N})$

Average-pooling. Average-pooling, which is straightforward, can be completed by evaluating the function $LUT(x) = \lfloor x/k \rceil$ in FBS, where k is the kernel size.

Max-pooling. Max-pooling is implemented using FBS with the max-tree method proposed by PEGASUS [30]. For a max-pooling layer with the kernel size k, $O(k)$ times LUT lookups are required.

Between Step ❺ and ❶, the Slot-to-Coefficient (S2C) operation transforms the FBS result, encrypted in slots, back into the coefficients (the m_is in Fig. 3), preparing for the subsequent loop in the Athena framework.

3.3 Specifics

Computational complexity analysis. Table 3 compares the computational complexity between Athena framework and conventional CKKS-based CNN implementation [27]. Unlike traditional designs, Athena omits ReLU and bootstrapping operations, as its FBS integrates both functionalities. The Packing and S2C in Athena serve as bridges between coefficient-encoded convolution and the FBS operation. As shown in the table, Athena's convolution avoids homomorphic rotation (HRot) operations due to the use of coefficient encoding, reducing computational overhead. Additionally, the complexity of Packing and S2C in Athena is lower than that in CKKS-based schemes for both CMult and PMult operations. Thus, the dominant computational cost in Athena lies in the FBS module, making its efficiency a critical factor for overall performance.

FBS essentially performs homomorphic table lookup, the number of entries in the LUT determines the number of terms in $FBS(x)$, i.e., smaller LUTs necessitate simpler polynomials. We adopt an optimized algorithm [34] to implement FBS. As shown in Alg. 2, the computation of the polynomial $FBS(x)$ is implemented using Baby-Step Giant-Step (BSGS) approach. Essentially, the original polynomial is partitioned into sub-polynomials and computed as a summation of products with corresponding powers of x. For example, $FBS(x) = 2x + 3x^2 + 4x^3 + 5x^4 = (2+3x)x + (4+5x)x^3$. The complexity of scalar-ciphertext multiplication (SMult hereafter) and homomorphic addition (HAdd hereafter) operations (line 4 and 6) is $O(t)$, and the complexity of CMult (line 6) operation is $O(\sqrt{t})$. Additionally, as the results of different layers have varying MAC results, (as shown in Fig. 4), we can construct a matching small LUT for layers. Therefore, the FBS module in Athena is designed to be flexible. For models with smaller intermediate values or lower quantization precision, its computational complexity is lower. It is worth noting that beyond computational complexity, Athena's key advantage lies in its smaller encryption parameters, which result in reduced computation overhead and improved hardware efficiency. This software-hardware co-design enables effective acceleration.

Algorithm 2: FBS

Input: BFV ciphertext ct; Polynomial $FBS(x) = \sum_{i=0}^{t-1} c_i x^i$.
Result: BFV ciphertext ct_{res}

1 $bs \leftarrow \lceil \sqrt{t} \rceil; gs \leftarrow \lceil \frac{t}{bs} \rceil$; **for** $i \longleftarrow 1, \cdots, gs$ **do**
2 **for** $j \longleftarrow 1, \cdots, bs$ **do**
3 $tmp \leftarrow HAdd(tmp, SMult(ct_j, c_{i*bs+j}))$;
4 **end**
5 $ct_{res} \leftarrow HAdd(ct_{res}, CMult(tmp, ct_{i*bs}))$;
6 **end**
 // ct_k is the k-th power of ct
7 Return ct_{res}

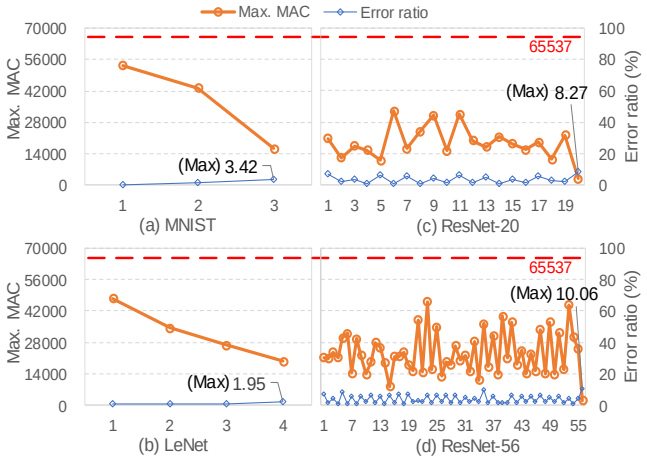


Figure 4: The rationality of parameter t setting (orange line) and the data error ratio introduced by noise e_{ms} (blue line).

Parameter settings and noise analysis. The parameters we selected are as follows: for RLWE, degree $N = 2^{15}$, ciphertext modulus $\log_2 Q = 720$, plaintext modulus $t = 65537$, The setting of t can hold the "maximum" MAC results (left Y-axis of Fig. 4) for each layer of CNN models under the w7a7 quantization. It guarantees the correctness of the ciphertext computation based on BFV; for LWE, degree $n = 2048$, modulus $q = t = 65537$; These parameters guarantee > 128 bits security. Parameter settings are tied to noise control, and the noise analysis has two aspects: (1) computational correctness. In Athena, the noise budget of the BFV ciphertext is consumed by linear operation, packing, FBS, and S2C. It's necessary to ensure that the noise growth does not exceed $\Delta/2$, where $\Delta = Q/t$. CMult, which is equivalent to plaintext-ciphertext multiplication (PMult hereafter), will introduce $\log_2 N + \log_2 t$ bits noise per computational depth, while SMult and HAdd will contribute $\log_2 t$ bits and 1 bit of noise growth, respectively. Table 4 shows the noise budget consumed by Athena. The total noise, which is ≤ 706 bits and less than $\Delta/2$, ensures computational correctness. (2) Impact on model accuracy. As outlined in Section 3.2.2, after switching the modulus to t, a small noise e_{ms}, following the $\mathcal{N}(0, (\frac{t\sigma}{Q})^2 + \frac{\|s\|^2+1}{12})$

Table 4: The maximum noise (in bits) introduced by different steps of Athena. C_{in} is the number of channels of input data in convolution layer.

	PMult (depth)	CMult (depth)	SMult (depth)	HAdd (depth)	Noise (bits)
Linear	1	0	0	$\log_2 C_{in}(6)$	37
Packing	1	0	0	12	43
FBS	0	17	1	15	558
S2C	2	0	0	6	68
Total	**4**	**17**	**1**	**39**	**706**

distribution, is introduced in the result of the linear layer. Here, σ is the standard deviation of the ciphertext before modulus switching, $\mathbf{s}$ is the secret key and $\|\mathbf{s}\| = \sqrt{\sum_{i=0}^{N-1} s_i^2}$. In practice, e_{ms} typically falls within about 4 bits. Remapping further compresses e_{ms} by multiplying it by *scale* as shown in ❺ in Fig. 3, and after $\lfloor x \times scale \rceil$, e_{ms} introduces a maximum error of ±1 to the remapping result. Also, only a small portion of the output data from the linear layer will be affected. Fig. 4 demonstrates the ratio of data with errors to the total per layer (right Y-axis). Most layers have error rates of less than 6%, or even 1%, with the highest not exceeding 11%. Such minor errors have a minimal impact on model accuracy, as demonstrated in Section 5.2.

3.4 Generalization

General QCNNs encompass linear, nonlinear, and re-quantization (remapping) operations. In Athena, the coefficient encoding based on BFV serves as a general method for linear functions. FBS can implement precise arbitrary nonlinear functions while also handling remapping. Different QCNN models can be implemented by simply performing the appropriate linear layer data position mapping based on the Athena framework according to their model parameters and activation functions, as well as adjusting the LUT function of the FBS. Therefore, Athena can support various neural networks. Furthermore, quantizing CNNs to enhance inference efficiency is widely accepted in the AI community. NVIDIA has proved using its PTQ or QAT could achieve comparable accuracy with its non-quantized counterpart [31]. Numerous quantized models and methodologies are available in the literature. Therefore, focusing on QCNNs under FHE does not restrict Athena's application scope.

4 Accelerator

4.1 Overall Architecture

Fig. 5 depicts the overall Athena accelerator architecture. It contains five computational units (CUs), i.e., Automorphism, (I)NTT, Pseudo-random number generator (PRNG) , Sample Extract (SE), and the FRU, which denotes the "FBS and RNS Base changing unit". The PRNG is utilized to halve the storage and bandwidth requirements of keyswitch keys, as with CraterLake [38] and SHARP [22]. SE is used to convert ciphertext formats between different FHE schemes (Step ❸ in Fig. 2). The FRU is capable of supporting modulo addition, modulo multiplication and base conversion through resource multiplexing. In addition, the accelerator is divided into two regions, i.e. Region 0 and Region 1, which interact with each other through an inter-region buffer. Region 0 contains all types of CUs to support the full range of FHE operations, while Region 1 is an FRU array, primarily utilized for accelerating a substantial

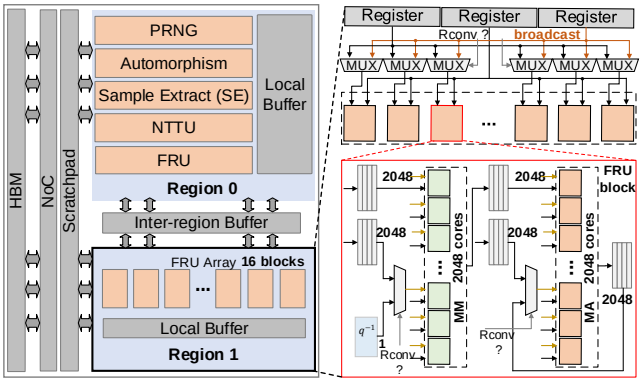


Figure 5: The overall architecture of the Athena accelerator.

number of PMult and HAdd within the Athena framework. The primary objective of region division is to facilitate the design of inter-regional data flows to efficiently support the bottleneck, i.e., FBS, of Athena framework (detailed in 4.3). All the CUs have their own local buffers, and the scratchpad is connected to the buffer via NoC. The parallelism of the accelerator is 2048.

Unlike some recently proposed architectures like ARK or CraterLake, the Athena accelerator does not instantiate large scratchpad memory because the optimized Athena framework has smaller parameters and unique non-linear and bootstrapping manner, i.e., FBS. The performance of FBS relies on the FRU Array, which has 2048×16 Modular Adders (MAs) and Modular Multipliers (MMs), and optimized data flow design based on the regional division. Importantly, the FRU is a versatile unit. It is employed not only for modulo addition and modulo multiplication, but also for base conversion within the Athena framework.

The design of the Athena accelerator architecture is based on two key observation: (1) the FBS involves a consecutive multiplication, accounts for the first-order overhead. Evidently, the order of SMult and CAdd is $O(N)$, which is much larger than the order of CMult, $O(\sqrt{N})$. Therefore, in our quantized inference framework Athena, NTT is no longer the biggest bottleneck, while the computational efficiency of MM and MA significantly affects the system performance; (2) FBS in Athena framework has obvious opportunities for pipelined computation, i.e., independent batch PMult and HMult operations. Therefore, allocating a large volume of shared MA/MMs in FRU and dividing the compute region will significantly improve the system performance.

4.2 Microarchitecture

4.2.1 (I)NTT and Automorphism Units. Unlike previous accelerator designs that cater to large parameters to cover the worst case, Athena only requires a polynomial with a degree of 2^{15}. We adopt the radix-8 NTT unit design, proposed in Poseidon [41], that can complete 3 NTT iterations in one core. A polynomial with a degree of 2^{15} requires merely $(log_2(2^{15})/3 = 5)$ iterations to finish, maximizing the advantages of the radix-8 NTT. In the Athena accelerator, we deploy 256 NTT units to process 2048 data in parallel and use a fully-pipelined manner to increase the throughput of the NTT cores. Specifically, Radix-8 NTT reduces the overhead of modular arithmetic units by fusing iterations and enhancing throughput. For $N = 2^{15}$, NTT requires only 5 iterations, and with 2048 hardware

parallelism, each iteration needs 2048 modular units, compared to 3072 units required by the traditional Radix-2 approach (1024 * 3). The design of the Automorphism unit is relatively mature. It essentially involves index mapping overhead and has lower computational requirements. Similar to Poseidon, we use data read and write control to implement the Automorphism. Our framework, Athena, has a fixed polynomial degree N of 2^{15}, and the parallelism l of a single core is set to 256, which can minimize the latency of $2(l + N/l)$. We deploy 8 cores to unify the total parallelism of 2048.

4.2.2 FRU. FBS in Athena involves substantial SMult and HAdd, essentially MA and MM computations, necessitating significant hardware power support. FHE schemes like CKKS and BFV are known for their NTT and RNS base changing bottlenecks. Therefore, general-purpose accelerators like CraterLake and SHARP [22] focus on designing optimized NTT units and using highly parallel units dedicated to RNS changing, allocating less power for MA and MM. Although the RNS changing unit comprises numerous MA and MM cores, its data flow supports only the RNS base changing. For instance, the CRB unit in CraterLake uses data broadcasting to share ciphertext data among 60 sets of 2048 parallel MAC blocks, setting another input to a constant register. The BConv unit in SHARP, using a systolic array design, facilitates ciphertext data flow between the PEs of the MACs to achieve data sharing. Despite having less IO overhead compared to CraterLake, the data flow scheduling is similarly singular.

As an accelerator dedicated to Athena, balancing the efficient RNS base changing and FBS computations is imperative. Therefore, we design a versatile unit, the FRU, that supports both operations by sharing high parallelism MM and MA cores through fine data flow scheduling. As depicted in Fig. 5, the accelerator contains a total of 17 FRU blocks, with one in Region 0 and 16 in Region 1, each with 2048 MM and MA cores. Two data paths are planned in FRU for the FBS and RNS base changing operations. As shown on the right side of Fig. 5, the FRU block requires two sets of input vector data for modular addition or modular multiplication, completing the FBS-related computation. For RNS base changing, one input data changes to the broadcasted data q^{-1} and the accumulation register (the output of MA cores). The "Rconv" signal is used to the selection of the different inputs in FRU block. Additionally, MA and MM are cascaded together, allowing the output of the MM core to serve as the input of the MA core and combined with its output as the input of the MA core for the MAC computation. In addition to the versatile FRUs, we also design efficient computational data flows based on two regions to match the computational characteristics of the FBS and improve its performance (detailed in Section 4.3).

4.2.3 Sample Extraction Unit. The unique SE unit in the Athena accelerator can convert RLWE ciphertexts into LWE ciphertexts. This process requires less computation but involves extensive transformation of data location. To extract $ct_i(lwe)$, as shown in Alg. 1, we need to cyclically shift all coefficients of the ciphertext term a of BFV by $i + 1$ positions and negate the $(i + 1)$-th to N-th element. Although the process can be implemented with $log_2 N$ complexity using the barrel shifter, it is inefficient for a large number of continuous shifting. In Athena, we fill the slots of a single ciphertext to the maximum with the results of the linear layer using a batch strategy. This requires close to N extraction operations for that

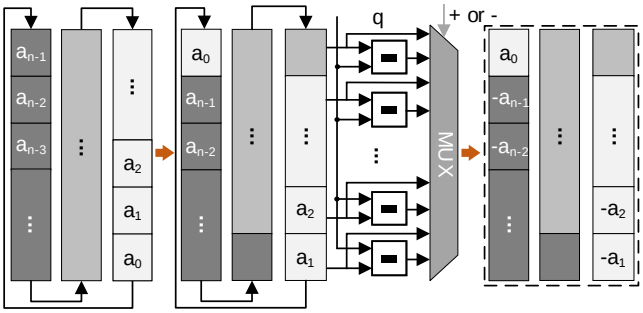


Figure 6: The sample extraction micro-architecture.

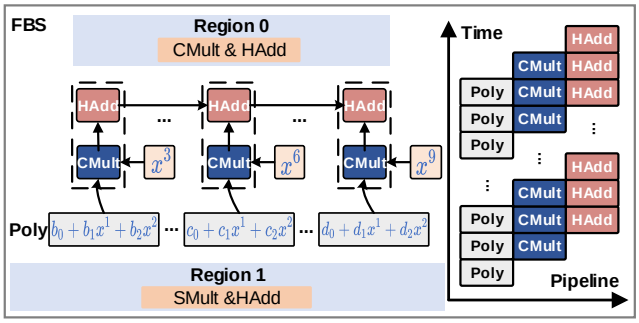


Figure 7: The computational dataflow design for FBS.

ciphertext. Therefore, in the SE unit, as shown in Fig. 6, we use a simple register shifter. Each cycle is shifted by one position, and the elements are selected positively or negatively according to the identification signals. In addition, since the input data must be in the corresponding modulus , only the subtractor is required for a negative value. In this way, the average cycle of a sample extract is close to 1.

4.3 Dataflow Design for FBS

The computational model of FBS as the bottleneck of the Athena framework is rich in opportunities for pipelined computation. Therefore, we combine the hardware design (two computational regions) to design an efficient computational flow to improve the performance of FBS. As shown in Alg. 2 in Section 3.3, FBS is performed by batch ciphertext-plaintext multiplication and addition (SMult and HAdd, i.e., *bs*) and single ciphertext multiplication (CMult, i.e., *gs*) with alternating computations between them, and *bs* and *gs* create excellent pipelining opportunities. The *bs* computations (Poly), primarily SMult and HAdd operations, are independent from *gs* computations which involve CMult. As shown in Fig. 7, We assign SMult and HAdd in *bs* to Region 1, and CMult in *gs* to Region 0. Accumulation (HAdd) is handled by a accumulating circuit in Region 0, enabling a fully pipelined FBS implementation. It is noteworthy that the key to efficient pipelining is to ensure that the computational latency of the two regions is similar. This is achieved by balancing the computational latency by allocating the CUs of the two regions according to the computational complexity of the Poly segmentation and CMult in the FBS. In Athena accelerator the parallelism of 2048 in Region 0 and the 16-block FRU array in Region 1 are capable of achieving this latency balance.

5 Evaluation

5.1 Experimental Setup

Platform. The major logic units of Athena are implemented in RTL using the ASAP7 7.5-track 7nm predictive process design kit (PDK) [10], and the SRAM components are evaluated by a cache modeling tool FinCACTI [39]. The area and power of two HBM modules are estimated based on prior works [20, 32]. Table **??** summarizes the resource utilization of Athena. We measure the benchmark runtime with a cycle-level simulator, and use the activity-level energy consumption from the synthesized components for energy evaluation.

Baseline. In our experiments, we compare Athena with the state-of-the-art ASIC-based FHE accelerators e.g., BTS [24], Craterlake [38], ARK [23] and SHARP [22]. These accelerators only report on ResNet-20. We normalize the computational complexity of other benchmarks to that of ResNet-20 as the performance estimation on these accelerators. For fairness, we also evaluate the performance of Athena framework on these accelerators, but the results prove that only the Athena accelerator achieves the best performance and efficiency (see Section 5.2.2).

Benchmark. We use the following four CNN benchmarks: (1) `MNIST`. It depicts the inference of a simple CNN [4] with one convolution layer and two fully connected layers. (2) `LeNet`. It depicts the classic CNN for the digit recognition task proposed in [26]. It includes two convolutional, two fully-connected and two max-pooling layers. We replace the original square activation function with ReLU to verify the efficacy of Athena in supporting more complex non-linear functions. (3) `ResNet-20`. It is implemented in [27, 28] trained with the CIFAR-10 dataset [25], which is employed by Craterlake and SHARP as well. It has 19 convolution layers and one fully-connected layer. The activation functions are uniformly ReLU. (4) `ResNet-56`. Compared with ResNet-20, ResNet-56 has similar backbone but deeper depth. In evaluating Athena, all the benchmarks are quantized into 7-bit weight and 7-bit activation termed as **(Athena-w7a7)**, and 6-bit weight and 7-bit activation termed as **(Athena-w6a7)**. In evaluating other FHE accelerators, we still use the floating-point CNN encrypted by CKKS, termed as the **CKKS-based**.

5.2 Performance

5.2.1 Accuracy. Athena focuses on the quantized model inference. Although a small amount of noise is introduced in the process, as elaborated in section 3.2.2, its impact on the model's accuracy is minimal. This is because the noise affects the LSBs of the data and is further reduced during remapping. Additionally, the model's noise immunity can be improved by introducing noise during the network training process. We first perform generic full-precision training (palin-G), followed by quantization-aware training to obtain quantized models (plain-Q). LeNet and MNIST are trained on MNIST dataset, and ResNet-20/56 on CIFAR-10. The quantized weights are then used in our Athena framework, where encrypted test images are processed via homomorphic inference. Accuracy is measured over all 10,000 test images from each dataset to evaluate ciphertext inference performance. The accuracy of CKKS-only implementations is the data reported in the source paper. As shown in Table 5, the CKKS-only implementation has a precision reduction of about 0.2% compared with plaintext. While the two quantized modes of Athena, i.e., w7a7 and w6a7, produce an error of less than 0.07% on LeNet and MNIST, and even achieve higher accuracy than quantized plain-Q on MNIST. For ResNet-20 and ResNet-56, Athena-w7a7 has a similar error rate. In addition, the accuracy of the quantized model (plain-Q) is very close to that of the high-precision generic model (plain-G), with only a 0.01%–0.07% difference. This also objectively indicates that Athena's ciphertext-domain inference incurs minimal accuracy loss, owing to the quantized framework's ability to precisely support nonlinear functions and bootstrapping.

Table 5: The accuracy of CNNs under plaintext and ciphertext inference. plain-G and plain-Q denote plaintext accuracy before and after quantization, respectively.

Method / Model	CKKS-based (%)		Athena (%)				
			plain-G	w7a7		w6a7	
	plain	cipher		plain-Q	cipher	plain-Q	cipher
MNIST	99.7	99.5 (-0.2)	98.34	98.27	98.28 (+0.01)	98.37	98.30 (-0.07)
LeNet	/	/	99.09	99.04	99.00 (-0.04)	99.03	99.00 (-0.03)
ResNet-20	91.52	91.31 (-0.21)	93.85	93.84	93.63 (-0.21)	93.51	93.45 (-0.06)
ResNet-56	93.27	93.07 (-0.2)	94.96	94.89	94.65 (-0.24)	94.79	94.66 (-0.13)

Table 6: Full-system performance comparison with SOTA accelerator prototypes. We use the actual benchmark execution time in "ms" as the metric.

	LeNet	MNIST	ResNet-20	ResNet-56
CraterLake [38]	182	35	321	946
ARK [23]	71	14	125	368
BTS [24]	1084	206	1910	5627
SHARP [22]	56	11	99	292
Athena-w7a7	**26.6**	**9.2**	**65.5**	**198.7**
Athena-w6a7	**24.1**	**7.3**	**54.9**	**157.8**

5.2.2 Speedup. We compare Athena's performance with state-of-the-art ASIC accelerators. Athena refers to the performance of the model running on our quantized framework on the accompanying accelerator, while other accelerators run the corresponding CKKS-based model. As shown in Table 6, Athena-w7a7 outperforms BTS [24] by 40× for LeNet and 29× for ResNet-20. Compared to CraterLake [38], Athena achieves a 3.8× to 6.8× speedup for all the selected models. Compared to SOTA ARK [23] and SHARP [22], the Athena also has a better performance, i.e. up to 2.66× and 2.1× speedup respectively. Athena-w6a7 has an overall advantage over SHARP because the cumulative results under the w6a7 quantize mode are smaller, and the LUT size of FBS can be flexibly adjusted to reduce the overhead of non-linear layers, which highlights Athena's flexibility and efficiency.

Athena requires dedicated accelerator support, and existing CKKS accelerators are not applicable to Athena. We evaluate the performance of the models under the Athena framework on SHARP and CraterLake to illustrate this point. They lack SE units for the sample extract operation of Athena, so we assume that they deploy

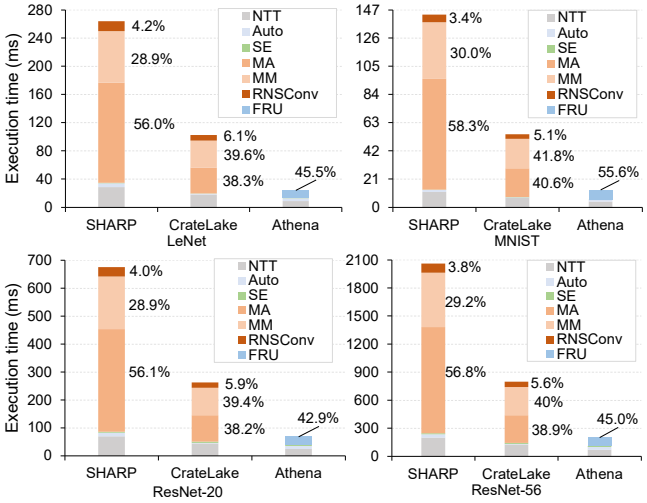


Figure 8: We deploy the Athena framework on the existing FHE accelerators and the Athena accelerator. Since Athena differs from CKKS-based implementations, existing accelerators are not suitable for Athena. For the Athena-specific modules like FRU and SE, we use MA/MM and RNSConv to substitute in SHARP and CraterLake.

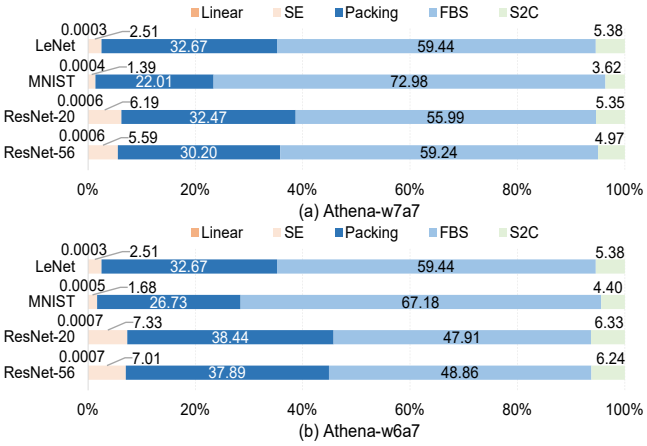


Figure 9: Execution time breakdown for two Athena representatives.

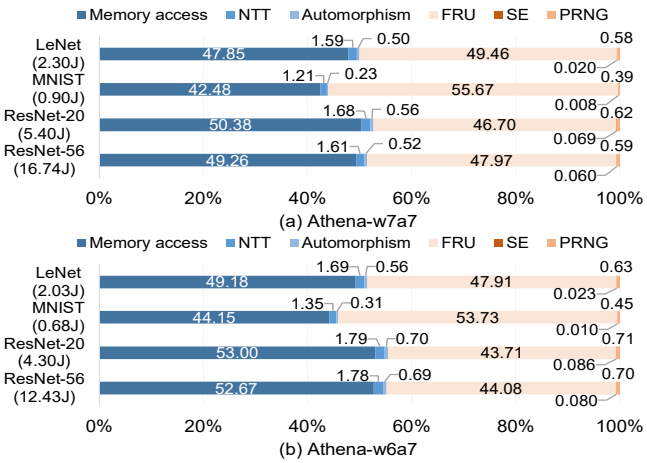


Figure 10: Full-system energy consumption and breakdown.

the same SE hardware units for ease of comparison. As shown in Fig. 8, the performance of CraterLake and SHARP is at least 3.8× and 9.9× lower than that of Athena accelerator. The main reason is that these accelerators only focus on NTT and RNSConv, which are known bottlenecks in CKKS applications, and cannot efficiently accelerate Athena's unique FBS operations. FBS contains a large number of MAs and MMs, so their MM and MA overheads are substantial and account for more than 77% and 84% of the total. Please note that the result reflects the challenge of matching Athena's workload rather than the merits of the CraterLake and SHARP architectures themselves. CraterLake's higher Athena performance results from having many multiplication and addition units, which partially ease Athena's bottleneck. SHARP's algorithmic and architectural co-design is highly innovative and performant in CKKS-based CNN inference. The Athena accelerator deploys versatile units, FRU, which supports RNSConv and FBS, with an optimized data flow design, thereby greatly improving performance. This also demonstrates the advantages of co-design between Quantized Athena framework and its accelerators.

The execution time breakdown of selected models is shown in Fig. 9. The non-linear part, i.e., FBS, accounts for the largest proportion, up to 72%, on all benchmarks. Activation, pooling, and softmax operators consume most of the execution time. Two points to note are: (1) ResNet-20 and ResNet-56 have a lower proportion of non-linear compared to the smaller models LeNet and MNIST. This is because the softmax operation has a higher number of activation operators, resulting in a correspondingly higher proportion of activation than the other two benchmarks. (2) LeNet uses the max-pooling layer instead of the average pooling layer used in ResNet-20 and ResNet-56. This requires more FBS operations and results in higher pooling consumption.

5.3 Energy

5.3.1 Consumption and Breakdown. In addition to performance, we also evaluate the full-system energy consumption of the Athena accelerator when running the four selected CNN models under two quantized modes, i.e., Athena-w7a7 and Athena-w6a7. As shown in Fig. 10, memory access accounts for about 50% of the energy consumption. Among the computation units, FRU consumes the largest proportion of energy due to its larger power requirement and the frequent use of FRU by the Athena bottleneck - FBS. This is consistent with the analysis in Fig. 9. Compared to Athena-w7a7, Athena-w6a7 has a similar memory access share, but the energy proportion of the FRU is slightly reduced. This is because the Athena-w6a7 linear layer has a smaller cumulative result, thus requiring a smaller LUT table, which results in reduced FBS computation.

5.3.2 Efficiency. We use the energy-delay product (EDP) and energy-delay-area product (EDAP) as efficiency metrics. Table 7 lists the EDP of the two modes of Athena and the previous accelerator

Table 7: Efficiency analysis. We use Energy Delay Product (EDP) as the metric (energy × time). Lower is better.

	LeNet	MNIST	ResNet-20	ResNet-56
CraterLake [38]	3.73	0.42	11.61	100.86
ARK [23]	0.64	0.138	1.99	17.25
BTS [24]	193.46	6.987	600.6	5213
SHARP [22]	0.31	0.012	0.96	8.36
Athena-w7a7	**0.056**	**0.008**	**0.35**	**3.32**
Athena-w6a7	**0.050**	**0.005**	**0.24**	**1.96**

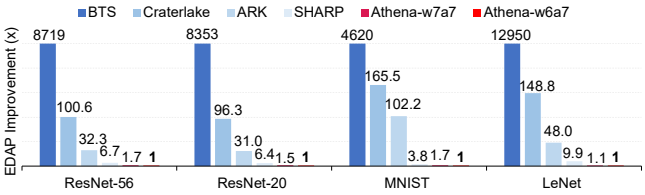


Figure 11: Energy-Delay-Area product (EDAP).

Table 8: Memory-related comparison.

Mem / Accelerator	HBM		Scratchpad	
	Cap.	BW	Cap.	BW
CraterLake [38]	16 GB	1 TB/s	256+26 MB	84 TB/s
ARK [23]	16 GB	1 TB/s	512+76 MB	92 TB/s
BTS [24]	16 GB	1 TB/s	512+22 MB	330 TB/s
SHARP [22]	16 GB	1 TB/s	180+18 MB	72 TB/s
Athena	16 GB	1 TB/s	45+15 MB	180 TB/s

ASICs. Athena has the highest efficiency among all the baselines for LeNet, ResNet-20, and ResNet-56. Compared to BTS, we achieve an efficiency improvement of over 8000×, attributable to the high-performance CNN inference framework and the coupled accelerator. Although SHARP focuses on optimizing the FHE parameter and hardware utilization, Athena still outperforms it by over 3.8×. The EDAP is depicted in Fig. 11. Compared to the EDP, the EDAP performance of Athena is better, benefiting from the significant area advantage of the Athena accelerator over other FHE accelerators.

5.4 On/Off-chip Memory

We compare the scratchpad and bandwidth profile of the Athena accelerator with other accelerators, as shown in Table 8. We use HBMs with capacities and bandwidths of 16 GB and 1 TB/s, respectively, to satisfy the accelerator's demand for off-chip memory. Existing accelerators like CraterLake and ARK deploy up to 256MB or even 512MB of scratchpad to support the hardware acceleration units, leading to a large area and power consumption overhead. Benefiting from the small encryption parameters of the Athena framework, its ciphertext size is less than that of the CKKS application by 4×. Therefore, we only need about 45MB of scratchpad to provide sufficient bandwidth for the acceleration unit. In terms of on-chip bandwidth, since our FRU array needs to process a large number of MAs and MMs in parallel, it requires a maximum bandwidth of 180TB/s, which is higher than other accelerators except

Table 9: Area and power breakdown (@1 GHz, 7nm).

Component	Area [mm^2]	Peak Power [W]
Automorphism	3.8	3.0
PRNG	1.2	1.9
NTT	4.51	3.9
SE	0.32	0.94
FRU	42.6	89.1
NoC	5.9	7.8
Register Files (15MB)	8.4	4.9
Scratchpad SRAM (45MB)	20.1	4.8
HBM (2 × HBM2E) (Cap. 16 GB / BW 1 TB/s)	29.6	31.8
Sum	**116.4**	**148.1**
CraterLake [38]	222.7 (7nm)	~207
ARK [23]	418.3	281.3
BTS [24]	373.6	133.8
SHARP [22]	178.8	/

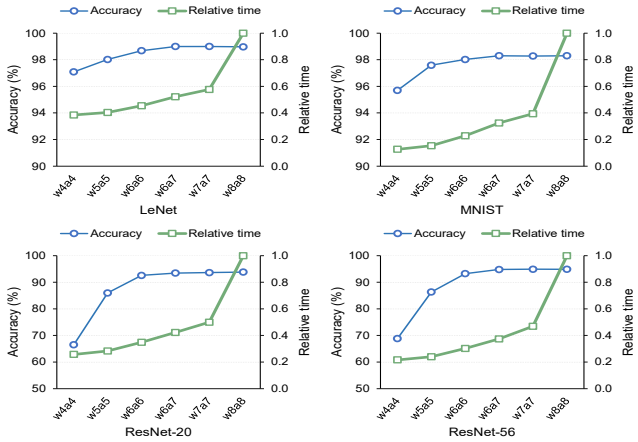


Figure 12: Sensitivity analysis of quantitative precision.

CraterLake, which requires a bandwidth support of 330TB/s due to its up to CRB unit with a parallelism of 2048*60.

5.5 Area and Power

We implement the Athena accelerator in RTL and synthesize it in a commercial 7nm technology node using SOTA tools, including the SRAM compiler. As shown in Table **??**, the Athena accelerator has a size of 116.4mm^2 and a power consumption of 148.1W, with the computation units accounting for a larger proportion. FRU is the compute unit with the highest overhead because it is designed for the high-overhead FBS operations in Athena. Meanwhile, on-chip storage has a lower overhead, benefiting the quantized framework due to the smaller size of ciphertext. This differs from existing accelerators, which need to allocate a large amount of storage resources to boost performance to match the large parameter requirements of Cryptosystem for CKKS-based applications. Therefore, the Athena accelerator surpasses ARK with improvements of 3.59× in area and 1.89× in power. Compared to the optimized SHARP, the area overhead of Athena is still 1.53× smaller.

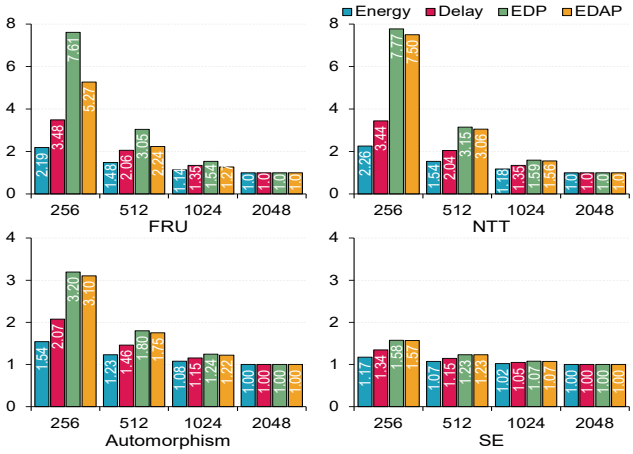


Figure 13: Sensitivity analysis of lanes.

5.6 Sensitivity

Fig. 12 illustrates Athena's inference accuracy and performance across various quantization precisions (w4a4 to w8a8). Regarding accuracy, while it improves with higher precision, significant gains plateau at w6a7, with minimal improvement beyond this point. In terms of performance, degradation accelerates after w6a6, with the most substantial drop occurring between w7a7 and w8a8, nearly doubling. This is due to increased overhead in the FBS as higher precision increases the cumulative results of linear layers. Despite this, Athena maintains strong performance even at the highest precision. To better balance accuracy and performance, we choose w6a7 and w7a7 as quantitative parameters for evaluation.

We also explore the performance scaling of the key computation units, i.e., NTT, FRU, Automorphism, and SE, on lanes from 256 to 2048, normalized to 2048. The metrics we use include delay, energy, EDP, and EDAP. As depicted in Fig. 13, the FRU significantly impacts on system performance, indicating Athena's need for efficient acceleration support for the bottleneck, i.e., FBS. Besides FRU, system performance is also more sensitive to the NTT unit, which is consistent with the general understanding that NTT operation has a high overhead in FHE. The SE unit has less impact on the system because it mainly involves data shifting and reading, rather than intensive computation. Compared to the SE unit, the Automorphism unit has a slightly greater influence on the efficiency of the accelerator due to more frequent rotation operations.

6 Conclusion

In this paper, we present Athena, a quantized deep learning inference framework under FHE, and design a coupled hardware accelerator for it. The framework supports high-accuracy CNN inference and arbitrary non-linear functions. The Athena process is simple and fixed, eliminating the complex parameter configuration to ensure accuracy. We implement the Athena prototype and evaluate its performance. The results demonstrate the effectiveness of this FHE-based quantized CNN inference system, with higher accuracy, faster speed, and smaller chip area. We hope this work can inspire new ideas for future FHE-based deep learning acceleration.

References

[1] Rashmi Agrawal, Anantha Chandrakasan, and Ajay Joshi. 2024. Heap: A fully homomorphic encryption accelerator with parallelized bootstrapping. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 756–769.

[2] Ahmad Al Badawi, Chao Jin, Jie Lin, Chan Fook Mun, Sim Jun Jie, Benjamin Hong Meng Tan, Xiao Nan, Khin Mi Mi Aung, and Vijay Ramaseshan Chandrasekhar. 2020. Towards the alexnet moment for homomorphic encryption: Hcnn, the first homomorphic cnn on encrypted data with gpus. *IEEE Transactions on Emerging Topics in Computing* 9, 3 (2020), 1330–1343.

[3] Amazon. [n. d.]. SageMaker with FHE. https://aws.amazon.com/cn/blogs/machine-learning/enable-fully-homomorphic-encryption-with-amazon-sagemaker-endpoints-for-secure-real-time-inferencing..

[4] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. 2021. TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption. arXiv:2104.03152 [cs.CR]

[5] Hervé Chabanne, Roch Lescuyer, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. 2019. Recognition over encrypted faces. In *Mobile, Secure, and Programmable Networking: 4th International Conference, MSPN 2018, Paris, France, June 18-20, 2018, Revised Selected Papers 4*. Springer, 174–191.

[6] Hao Chen, Kim Laine, and Peter Rindal. 2017. Fast private set intersection from homomorphic encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1243–1255.

[7] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2018. Bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part I 37*. Springer, 360–384.

[8] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085* (2018).

[9] Edward Chou, Josh Beal, Daniel Levy, Serena Yeung, Albert Haque, and Li Fei-Fei. 2018. Faster cryptonets: Leveraging sparsity for real-world encrypted inference. *arXiv preprint arXiv:1811.09953* (2018).

[10] Lawrence T Clark, Vinay Vashishtha, David M Harris, Samuel Dietrich, and Zunyan Wang. 2017. Design flows and collateral for the ASAP7 7nm FinFET predictive process design kit. In *2017 IEEE international conference on microelectronic systems education (MSE)*. IEEE, 1–4.

[11] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. 2018. PIR-PSI: scaling private contact discovery. *Cryptology ePrint Archive* (2018).

[12] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P Smart. 2013. Field switching in BGV-style homomorphic encryption. *Journal of Computer Security* 21, 5 (2013), 663–684.

[13] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*. PMLR, 201–210.

[14] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. 2019. Logistic regression on homomorphic encrypted data at scale. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 9466–9471.

[15] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. 2017. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189* (2017).

[16] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. 2022. Cheetah: Lean and fast secure {Two-Party} deep neural network inference. In *31st USENIX Security Symposium (USENIX Security 22)*. 809–826.

[17] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. 2020. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv preprint arXiv:2006.10518* (2020).

[18] Takumi Ishiyama, Takuya Suzuki, and Hayato Yamana. 2020. Highly accurate CNN inference using approximate activation functions over homomorphic encryption. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 3989–3995.

[19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2704–2713.

[20] Norman P Jouppi, Doe Hyun Yoon, Matthew Ashcraft, Mark Gottscho, Thomas B Jablin, George Kurian, James Laudon, Sheng Li, Peter Ma, Xiaoyu Ma, et al. 2021. Ten lessons from three generations shaped google's tpuv4i: Industrial product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 1–14.

[21] Jae Hyung Ju, Jaiyoung Park, Jongmin Kim, Donghwan Kim, and Jung Ho Ahn. 2023. Neujeans: Private neural network inference with joint optimization of convolution and bootstrapping. *arXiv preprint arXiv:2312.04356* (2023).

[22] Jongmin Kim, Sangpyo Kim, Jaewan Choi, Jaiyoung Park, Donghwan Kim, and Jung Ho Ahn. 2023. SHARP: A Short-Word Hierarchical Accelerator for Robust and Practical Fully Homomorphic Encryption. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 1–15.

[23] Jongmin Kim, Gwangho Lee, Sangpyo Kim, Gina Sohn, John Kim, Minsoo Rhu, and Jung Ho Ahn. 2022. ARK: Fully Homomorphic Encryption Accelerator with Runtime Data Generation and Inter-Operation Key Reuse. *arXiv preprint arXiv:2205.00922* (2022).

[24] Sangpyo Kim, Jongmin Kim, Michael Jaemin Kim, Wonkyung Jung, John Kim, Minsoo Rhu, and Jung Ho Ahn. 2022. BTS: An accelerator for bootstrappable fully homomorphic encryption. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*. 711–725.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[27] Eunsang Lee, Joon-Woo Lee, Junghyun Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and Woosuk Choi. 2022. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In *International Conference on Machine Learning*. PMLR, 12403–12422.

[28] Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, et al. 2022. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access* 10 (2022), 30039–30054.

[29] Zeyu Liu and Yunhao Wang. 2023. Amortized functional bootstrapping in less than 7 ms, with $\tilde{O}(1)$ polynomial multiplications. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 101–132.

[30] Wen-jie Lu, Zhicong Huang, Cheng Hong, Yiping Ma, and Hunter Qu. 2021. PEGASUS: bridging polynomial and non-polynomial evaluations in homomorphic encryption. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1057–1073.

[31] NVIDIA. [n. d.]. 8-bit inference with TensorRT. https://www.cse.iitd.ac.in/~rijurekha/course/tensorrt.pdf.

[32] Mike O'Connor, Niladrish Chatterjee, Donghyuk Lee, John Wilson, Aditya Agrawal, Stephen W Keckler, and William J Dally. 2017. Fine-grained DRAM: Energy-efficient DRAM for extreme bandwidth systems. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. 41–54.

[33] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. 2018. Value-aware quantization for training and inference of neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 580–595.

[34] Michael S Paterson and Larry J Stockmeyer. 1973. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.* 2, 1 (1973), 60–66.

[35] Brandon Reagen, Woo-Seok Choi, Yeongil Ko, Vincent T Lee, Hsien-Hsin S Lee, Gu-Yeon Wei, and David Brooks. 2021. Cheetah: Optimizing and accelerating homomorphic encryption for private inference. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 26–39.

[36] Sujoy Sinha Roy, Furkan Turan, Kimmo Jarvinen, Frederik Vercauteren, and Ingrid Verbauwhede. 2019. FPGA-based high-performance parallel architecture for homomorphic computing on encrypted data. In *2019 IEEE International symposium on high performance computer architecture (HPCA)*. IEEE, 387–398.

[37] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Srinivas Devadas, Ronald Dreslinski, Christopher Peikert, and Daniel Sanchez. 2021. F1: A fast and programmable accelerator for fully homomorphic encryption. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 238–252.

[38] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Nathan Manohar, Nicholas Genise, Srinivas Devadas, Karim Eldefrawy, Chris Peikert, and Daniel Sanchez. 2022. CraterLake: a hardware accelerator for efficient unbounded computation on encrypted data.. In *ISCA*. 173–187.

[39] Alireza Shafaei, Yanzhi Wang, Xue Lin, and Massoud Pedram. 2014. FinCACTI: Architectural analysis and modeling of caches with deeply-scaled FinFET devices. In *2014 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 290–295.

[40] Liyan Shen, Xiaojun Chen, Dakui Wang, Binxing Fang, and Ye Dong. 2018. Efficient and private set intersection of human genomes. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 761–764.

[41] Yinghao Yang, Huaizhi Zhang, Shengyu Fan, Hang Lu, Mingzhe Zhang, and Xiaowei Li. 2023. Poseidon: Practical Homomorphic Encryption Accelerator. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 870–881.

[42] Zhaomin Yang, Xiang Xie, Huajie Shen, Shiying Chen, and Jun Zhou. 2021. TOTA: fully homomorphic encryption with smaller parameters and stronger security. *Cryptology ePrint Archive* (2021).

[43] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. 2021. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*. PMLR, 11875–11886.