Jaguar: Fast Private Neural Network Inference with Power-of-Two Homomorphic Arithmetic

Yewon Jeong* **Nayoung Jung*** **Hyeri Roh** **Woo-Seok Choi†**
Department of Electrical and Computer Engineering, ISRC, Seoul National University
{yellowon518,18170lanige,hrroh,wooseokchoi}@snu.ac.kr
*Equal contribution.

Abstract

Hybrid HE/2PC private neural network inference remains bottlenecked by prime-modulus homomorphic arithmetic in convolution and by a precision flow that runs ReLU at doubled bitwidth before invoking a separate truncation protocol. We present **Jaguar**, a system built on a single design choice—a *power-of-two ciphertext ring*—that addresses both. The choice enables SPA-CONV, a coefficient-domain convolution kernel that replaces NTT-centric polynomial multiplication with Scalar–Polynomial Accumulation, and an *exact* ciphertext-side truncation by local right shifts that lets ReLU run directly at the target fixed-point precision and eliminates the post-ReLU truncation protocol. Where NTT remains genuinely useful—at the client, for the single polynomial multiplication during decryption—we recover it through an auxiliary NTT prime, preserving the power-of-two protocol substrate while keeping decryption $O(N \log N)$. On ImageNet-scale ResNet-18, ResNet-50, and MobileNetV2 with AVX disabled, Jaguar achieves 2.07–3.72$\times$ lower end-to-end latency than Cheetah and 2.16–3.36$\times$ lower than Rhombus, with 1.16–1.76$\times$ lower communication than Cheetah. Our implementation is publicly available at `https://github.com/yellowon/Jaguar`.

1 Introduction

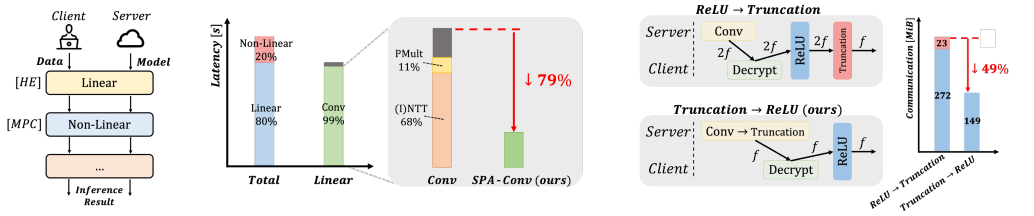
Machine learning as a service (MLaaS) is widely deployed for vision [1], medical [2, 3], mobile [4], and embedded [5, 6] inference. Remote neural network inference creates a two-sided privacy problem—the client wants to protect sensitive inputs, the server wants to protect proprietary model parameters—solved by *private inference* without revealing either side's secrets [7, 8, 9, 10, 11, 12].

Two paradigms have emerged. *Fully homomorphic encryption(HE)-based* inference evaluates the entire network under encryption but pays heavy compute cost and typically replaces nonlinearities with polynomial approximations, degrading accuracy [13, 14, 15, 16, 17, 18]. *Hybrid HE/2PC* inference evaluates linear layers under HE and nonlinear layers under secure two-party computation(2PC) [9, 10, 11, 12] (Figure 1(a)). This avoids activation approximation and is the practical design point for exact private neural network inference.

Within this hybrid framework, HE evaluates static-weight linear operators and returns their outputs as additive shares for subsequent secure computation [12]. This linear-to-share interface appears across neural architectures, including convolutions in CNNs and token-wise projections in attention models [19]. This paper targets the homomorphic arithmetic underlying this interface. We use CNNs as the running example in the following discussion and as the basis for our end-to-end implementation and evaluation. This setting allows us to establish the mathematical correctness, noise behavior, and system-level performance of our new arithmetic design within a complete inference pipeline, while

†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Hybrid HE/MPC Framework (b) Latency breakdown of convolution layers (c) Communication cost breakdown of ReLU and Truncation layers

Figure 1: Motivation for Jaguar's design choices. (a) Hybrid HE/2PC private inference framework. (b) Latency breakdown of Cheetah's ResNet-50 convolutions under LAN. (c) Communication of the ReLU+truncation stage on ResNet-50.

leaving the choice of private attention protocols separate. Within this CNN setting, our profiling identifies *two bottlenecks* in the hybrid inference pipeline.

The first is HE convolution. On ResNet-50 under LAN, our profiling of Cheetah [12, 20] shows convolution alone takes 96.71 s out of 120.82 s, over 99 % of linear-layer latency. Within HE convolution, NTT-domain polynomial operations dominate: forward and inverse NTTs together account for 68 % of convolution time, and modular prime reductions—appearing both inside and outside NTTs—account for 34.6 % (Figure 1(b)). The bottleneck is not implementation inefficiency—it is the NTT-prime *arithmetic regime* that the protocol was built on.

A second, less-discussed bottleneck is precision flow. A linear layer in fixed-point inference produces output at *doubled* fractional precision ($2f$ bits). Conventional hybrid systems pass this directly into ReLU and then invoke a *separate* truncation protocol [11, 12, 21, 22]. This is doubly wasteful: ReLU pays for the wider bitwidth, and an interactive 2PC truncation is added on top.

Why power-of-two. Both bottlenecks share a root cause: the *prime-modulus* convention inherited from FHE library design. NTT-friendly primes are excellent for generic polynomial multiplication, so libraries default to them. But a hybrid HE/2PC system performs neither generic polynomial multiplication nor deep homomorphic circuits—it performs a single linear layer at a time, immediately converts to additive shares over $\mathbb{Z}_{2^\ell}$, and reuses the share domain for 2PC. In this regime the prime modulus *blocks* two capabilities the protocol very much wants:

1. **Reduction-free coefficient arithmetic**. Convolution can be evaluated as a sum of shifted scalar–polynomial products without ever multiplying two polynomials, *if* coefficient-wise multiplication and addition are cheap. Power-of-two arithmetic makes them cheap by replacing modular reduction with a bit mask.
2. **Exact ciphertext-side truncation**. A right shift on each ciphertext component implements an exact plaintext truncation *iff* the BFV scale Δ has 2^f as a divisor—automatic when both q and p are powers of two, impossible when q is a product of NTT-friendly primes.

This paper proposes **Jaguar** that adopts $q = 2^Q, p = 2^P$ as the central design choice. SPA-CONV and pre-ReLU ciphertext truncation are then *consequences* of this choice, not independent optimizations. The remaining technical question is whether giving up NTT in the linear path costs us elsewhere; it does not, because NTT can be recovered as a local client-side tool exactly where it pays off.

Contributions. We implement Jaguar on ImageNet-scale ResNet-18, ResNet-50, and MobileNetV2 under both LAN and WAN. With AVX disabled, end-to-end latency is **2.07–3.72× lower than Cheetah** and **2.16–3.36× lower than Rhombus** [23]; communication is 1.16–1.76× lower than Cheetah and comparable to Rhombus (1.09× and 1.41× lower on ResNet-18 and MobileNetV2; 1.28× higher on ResNet-50—we discuss this case explicitly in Section 5). Convolution-kernel latency drops 79 % on ResNet-50 (Figure 1(b)); ReLU+truncation communication drops 1.98×, with the truncation stage entirely eliminated (Figure 1(c)). The technical contributions are:

- **A power-of-two HE arithmetic substrate** for hybrid HE/2PC private neural network inference. To our knowledge, Jaguar is the first HE/2PC system to use a power-of-two *ciphertext* ring as the central arithmetic regime for ImageNet-scale CNN inference. Prior coefficient-encoded systems used a power-of-two plaintext/share ring but kept the ciphertext backend on NTT-friendly primes.

- **SPA-CONV**, a coefficient-domain convolution kernel that evaluates pointwise, dense, and depthwise convolutions as sums of scalar–polynomial products and coefficient shifts, with no plaintext–ciphertext polynomial multiplication, NTT, or modular prime reduction.
- **Exact pre-ReLU ciphertext-side truncation,** a zero-communication operation that removes the separate post-ReLU truncation protocol used by all prior hybrid HE/2PC systems and reduces the bitwidth at which ReLU is evaluated. We prove its correctness (Theorem 1).
- **NTT-assisted decryption,** a client-local technique that recovers NTT exactly where it is genuinely useful—the single $\hat{c}_1' \cdot \mathsf{sk}$ multiplication during decryption—via an auxiliary NTT prime, without disturbing the power-of-two protocol substrate.

2 Preliminaries

Notations. For a positive integer n, $[n] = \{0, \ldots, n-1\}$. We use lower-case letters with a "hat" symbol for polynomials, bold lower-case for vectors, bold upper-case for matrices, and calligraphic for tensors. Let $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ be the negacyclic polynomial ring of degree N; for $\hat{x}(X) \in R_q$, $\hat{x}[i]$ is its i-th coefficient, and $\rho_\eta(\hat{x}) = X^\eta \hat{x} \bmod (X^N + 1)$ is a coefficient shift by offset η. A convolution layer has input shape (H, W, C_{in}), kernel shape $(K, K, C_{\text{in}}, C_{\text{out}})$, stride s_{stride}, and padding p_{pad}. Values are fixed-point integers with f fractional bits; a linear product has precision $2f$, and $x \gg \tau$ denotes arithmetic right shift by τ bits. We write $\langle x \rangle = (\langle x \rangle_C, \langle x \rangle_S)$ for additive sharing over $\mathbb{Z}_{2^\ell}$, with $\langle x \rangle_C + \langle x \rangle_S \equiv x \pmod{2^\ell}$. A complete notation table is provided in Appendix A.

2.1 Cryptographic Primitives

Hybrid HE/2PC inference. Jaguar follows the standard hybrid interface: HE evaluates linear layers, 2PC evaluates nonlinearities [9, 10, 11, 12]. We instantiate HE with BFV [24]. At the HE→2PC boundary, the server masks an encrypted output $\mathsf{Enc}(\hat{y})$ with $\hat{r}$, sends $\mathsf{Enc}(\hat{y} - \hat{r})$ to the client, and keeps $\hat{r}$ as its share; the client decrypts to obtain the complementary share. Each HE block is converted to shares before the next nonlinear layer, so bootstrapping is unnecessary.

Encoding and arithmetic regimes. *SIMD encoding* [9, 19, 25] packs values into transform-domain slots defined by the number-theoretic transform (NTT), the finite-field analogue of the FFT [26]. This batching amortizes homomorphic operations but requires a *prime* plaintext modulus, making the share domain prime-field rather than $\mathbb{Z}_{2^\ell}$ and creating friction with ring-based 2PC. *Coefficient encoding* [12, 23] maps values directly to polynomial coefficients; with careful arrangement, polynomial multiplication realizes convolutions, and Cheetah-style systems [12] can avoid costly rotations and use a power-of-two plaintext/share domain. **An important subtlety**: even systems using a power-of-two *plaintext* ring still use a prime-modulus *ciphertext* backend with RNS prime limbs, so plaintext–ciphertext products incur NTT-domain arithmetic and prime-modulus reductions. Jaguar keeps coefficient encoding but changes the ciphertext substrate itself.

Threat model. Standard two-party private inference: the client holds a private input, the server a proprietary model. The client learns only the final prediction; the server learns nothing beyond what the protocol transcript and output policy reveal. As in prior art [9, 12, 19, 27], we assume an honest-but-curious adversary: parties follow the protocol but may infer additional information from messages. Network architecture, tensor shapes, and protocol parameters are public. Security follows by composing the security of BFV, additive secret sharing, and the underlying 2PC protocols.

2.2 Related Work

Prior work has improved either the mapping of linear layers to secure computation or the nonlinear protocols (Table 1). Gazelle [9] uses SIMD with a prime plaintext modulus; Cheetah [12] introduces coefficient-encoded, rotation-free linear protocols with lean OT-based nonlinearities; CrypTFlow2 [11] focuses on faithful 2PC primitives; Rhombus [23] reduces output communication for matrix-vector multiplication via input-output packing; Falcon [28] specializes packing for depthwise convolutions. Orthogonally, PrivCirNet [29] modifies the model to reduce HE linear cost.

Why hasn't power-of-two been tried before? Two reasons. First, the standard FHE library backend (e.g., Microsoft SEAL [30]) is built around NTT-friendly RNS primes; switching to power-of-two requires re-deriving the convolution kernel because generic plaintext–ciphertext polynomial

multiplication via NTT becomes unavailable. Second, the perceived cost of giving up NTT looks prohibitive *if one assumes generic polynomial multiplication is needed*. Our SPA-CONV shows that for CNN convolution, scalar–polynomial products and coefficient shifts suffice, and even the residual NTT need (decryption) can be handled locally by the client. Once both observations land, the prime-modulus rationale weakens and the power-of-two regime becomes the natural choice.

Table 1: Comparison with representative hybrid HE/2PC private CNN inference systems.

Method	Linear side		Truncation	Nonlinear side	
	Target	Approach	Approach	Target	Approach
Gazelle [9]	Conv/MatMul	Packed SIMD	–	ReLU/Pool	Exact GC
CrypTFlow2 [11]	Conv/MatMul	Exact OT/HE backend	Faithful 2PC	Cmp./Div.	Exact 2PC
Cheetah [12]	Conv/MatMul	Coefficient-encoding protocol	Lean OT	ReLU	Lean OT
Rhombus [23]	PW-Conv/MatMul	Efficient output repacking	–	–	–
Falcon [28]	DW-Conv	Dense packing	–	–	–
Jaguar (ours)	**Conv**	**Power-of-two arithmetic regime**	**HE-side, zero-cost truncation**	**ReLU**	**Reduced-bitwidth evaluation**

3 Proposed Techniques

This section develops Jaguar's contributions in the order they motivate each other. Section 3.1 starts from the precision-flow mismatch and proves an exact ciphertext-side truncation. Section 3.2 shows that the divisibility condition for this truncation, together with $\mathbb{Z}_{2^\ell}$ share-domain alignment, makes a power-of-two ciphertext modulus natural. Section 3.3 presents SPA-CONV, the convolution kernel for this regime. Section 3.4 closes the loop by handling decryption.

3.1 Zero-Cost Precise Truncation

Motivation. A linear layer multiplies fixed-point inputs and weights with f fractional bits each, producing output at $2f$-bit precision. Conventional hybrid systems hand this to the ReLU protocol and then run a separate 2PC truncation [11, 12]. Both steps cost interaction. Our goal: truncate *before* ReLU, with no 2PC interaction.

Construction. Let $(\hat{c}_0, \hat{c}_1) \in R_q^2$ be a BFV ciphertext encrypting a plaintext $\hat{m}_0$, with $[\hat{c}_0 + \hat{c}_1\hat{s}]_q = \Delta\hat{m}_0 + \hat{\epsilon}$ for noise $\hat{\epsilon}$ and scale $\Delta = \lfloor q/p \rfloor$. Algorithm 1 truncates each ciphertext component by an arithmetic right shift, implemented as logical shifts on unsigned residues (Lemma 2). Algorithm 2 decrypts the truncated ciphertext using a *two-step* divide-and-round.

Algorithm 1: Proposed truncation

Input: Ciphertext $[\![\hat{m}_0]\!] = (\hat{c}_0, \hat{c}_1) \in R_q^2$, truncation parameter f
Output: Ciphertext $[\![\hat{m}_1]\!] = (\hat{c}'_0, \hat{c}'_1)$ where $\hat{m}_1[i] = (\hat{m}_0[i] \gg f)$
for $i \leftarrow 0$ to $N-1$ **do**
  $\hat{c}'_0[i] = (\hat{c}_0[i] \gg f)$
  $\hat{c}'_1[i] = (\hat{c}_1[i] \gg f)$

Algorithm 2: Decryption algorithm for f-bit truncated ciphertext

Input: Secret key $\mathsf{sk} = \hat{s} \in R_q$, ciphertext $[\![\hat{m}_1]\!] = (\hat{c}'_0, \hat{c}'_1) \in R^2_{q \gg f}$ obtained from Algorithm 1
Output: Plaintext $\hat{m}_1 \in R_{p \gg f}$
$\hat{m}_1 \leftarrow \lfloor \frac{1}{2^f} \lfloor \frac{2^f}{\Delta} [\hat{c}'_0 + \hat{c}'_1 \hat{s}]_{q \gg f} \rceil \rfloor \pmod{p \gg f}$

Theorem 1. *Algorithm 1 implements exact f-bit truncation, decryptable by Algorithm 2, provided that the ciphertext noise satisfies $||\hat{\epsilon}||_\infty < \Delta/2 - (N+1)2^f$. (Proof in Appendix B.)*

Why two-step divide-and-round. Conventional BFV decryption [24] does $\lfloor \frac{1}{\Delta}[\cdot] \rceil$ in one step. Algorithm 2 first divides by $\Delta/2^f$, rounds, then truncates by f bits. This separation is what makes the result *exact*: because Δ is divisible by 2^f when both q and p are powers of two, the inner

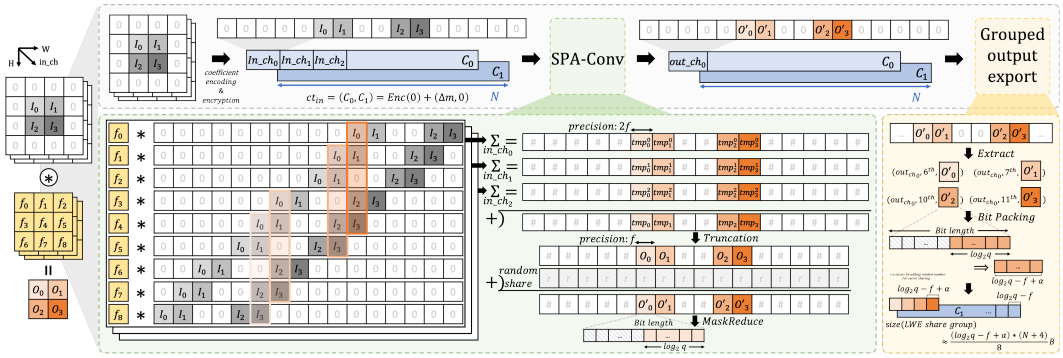


Figure 2: Computation flow of SPA-CONV.

divide-by-$\Delta/2^f$ then the rounding cleanly removes the noise polynomial as shown in the proof. A single-step decryption applied to a shifted ciphertext would conflate noise with truncation rounding.

Noise budget. Under our parameters $(N, Q, P) = (2048, 54, 30)$ and $f = 9$, Theorem 1 requires accumulated noise below $2^{23} - 2049 \cdot 2^9 \approx 7.34 \times 10^6$. Each Jaguar linear block has multiplicative depth one before share conversion: noise in a fresh ciphertext ($\sigma_{\text{in}} = 3.2$) is scaled by the plaintext filter coefficients and accumulated across SPA-CONV terms. Across all three evaluated networks, the worst maximum-accumulation layer is a 960-term pointwise layer in MobileNetV2, which retains a 25.9σ margin to Theorem 1's bound; the corresponding ResNet-18 and ResNet-50 margins are much larger (details in Appendix C).

3.2 Power-of-Two Arithmetic Regime

Modulus choice. Security in BFV rests on the concrete hardness of the underlying RLWE instance—ring dimension, modulus, secret distribution, error distribution—and *does not require q to be prime* [24, 31, 32]. The prime/RNS convention in HE libraries is implementation-driven, not BFV-driven [33]. Jaguar adopts $q = 2^Q$, $p = 2^P$, so the BFV scale $\Delta = q/p = 2^{Q-P}$ is also a power of two. This single choice does three things: **1) Satisfies Theorem 1 by construction;** A f-bit ciphertext truncation is exact iff $2^f | \Delta$. With $\Delta = 2^{Q-P}$, every $f \le Q - P$ qualifies. **2) Aligns HE outputs with the $\mathbb{Z}_{2^\ell}$ share domain;** Subsequent 2PC operates over $\mathbb{Z}_{2^\ell}$. Using $p = 2^P$ avoids the share-domain mismatch SIMD-encoded systems must bridge. **3) Replaces modular reduction with bit masking;** Reduction modulo 2^Q is a single bitwise AND with $2^Q - 1$.

Correctness and security. The arithmetic backend changes; the privacy goal does not. Correctness follows the standard BFV requirement that accumulated noise stays below $\Delta/2$ minus the truncation slack of Theorem 1; depth-one linear blocks make this modest. Security depends on the concrete RLWE hardness of (N, Q, χ_e), not on whether Q is the bit-length of a prime. We choose $(N, Q, P) = (2048, 54, 30)$ to satisfy both correctness and 128-bit classical security (Appendix D).

Algorithmic consequence. Jaguar cannot use the standard NTT-friendly prime backend for convolution. The convolution kernel must be expressed in terms the power-of-two ring supports natively—scalar–polynomial products, coefficient shifts, additions, and bit operations—which Sec. 3.3 (SPA-CONV) provides. The single $\hat{c}'_1 \cdot \mathsf{sk}$ multiplication needed for decryption is handled by Sec. 3.4.

3.3 SPA-CONV: Scalar–Polynomial Accumulation Convolution

Intuition. A convolution is a sum of products of *scalar* filter weights and *spatial* input neighborhoods. If input channels are coefficient-encoded into RLWE ciphertexts, then for each filter coefficient $w_{o,c,u,v}$, the contribution to output channel o at spatial offset (u, v) is *one scalar* times *one shifted ciphertext*. Aggregating across input channels and kernel offsets yields the full output. No polynomial-by-polynomial multiplication is needed in the linear path; therefore **no server-side NTT is needed**.

Setup. Let $X = \{\mathrm{CT}_t\}_{t=0}^{M-1}$ be encrypted input blocks. A public packing layout $\Lambda(c, j) = (\iota, \sigma)$ returns the source ciphertext index ι and in-ciphertext coefficient offset σ for input channel c and output block j, covering both *packed* (multiple channels per ciphertext) and *split* (one channel across multiple ciphertexts) layouts. Let B be the number of output blocks per output channel. For an

Table 2: Kernel-level complexity of secure convolution in *estimated 64-bit multiplications*. Baselines are optimistic lower-envelope estimates under native packing assumptions, using $\Gamma_{\mathrm{mul}} = 6L(\log_2 N + 2)$ and $\Gamma_{\mathrm{rot}} = 3\big((\frac{\log_2 N}{2} + 2)L^2 + (\log_2 N + 2)L\big)$; details in Appendix I. L is the number of RNS limbs; $S_{\mathrm{out}} = H'W'$ is the output size; C_w is Falcon's per-polynomial channel packing.

Category	Method	HE-PMult	HE-Rot	Scalar-Poly
Dense	Gazelle	$HWC_{\mathrm{in}}C_{\mathrm{out}}\Gamma_{\mathrm{mul}}$	$(HW(C_{\mathrm{in}} + C_{\mathrm{out}}) + C_{\mathrm{out}}N)\Gamma_{\mathrm{rot}}$	–
	Cheetah	$HWC_{\mathrm{in}}C_{\mathrm{out}}\Gamma_{\mathrm{mul}}$	0	–
	Jaguar	**0**	**0**	$2NBC_{\mathrm{in}}C_{\mathrm{out}}K^2$
PW	Rhombus-V2	$S_{\mathrm{out}}C_{\mathrm{in}}C_{\mathrm{out}}\Gamma_{\mathrm{mul}}$	$\sqrt{S_{\mathrm{out}}C_{\mathrm{in}}C_{\mathrm{out}}N}\Gamma_{\mathrm{rot}}$	–
	Jaguar	**0**	**0**	$2NBC_{\mathrm{in}}C_{\mathrm{out}}$
DW	Falcon†	$\frac{C_{\mathrm{out}}}{C_w}N\Gamma_{\mathrm{mul}}$	0	–
	Jaguar	**0**	**0**	$2NBC_{\mathrm{out}}K^2$

$K \times K$ kernel with $K = 2r + 1$, define the spatial-offset set $\Omega_K = \{-r, \ldots, r\}^2$; with padded row width W_{pp}, the coefficient offset for spatial displacement (u, v) is $\delta(u, v) = uW_{\mathrm{pp}} + v$. For output channel o and block j, let $\mathcal{C}(o)$ denote the contributing input channels: $\mathcal{C}(o) = [C_{\mathrm{in}}]$ for dense and pointwise convolutions, and $\mathcal{C}(o) = \{o\}$ for depthwise convolution.

Kernel. SPA-CONV forms the raw encrypted output block

$$A_{o,j} = \sum_{c \in \mathcal{C}(o)} \sum_{(u,v) \in \Omega_K} w_{o,c,u,v} \cdot \rho_{\sigma(c,j)+\delta(u,v)}\big(\mathrm{CT}_{\iota(c,j)}\big), \qquad \Lambda(c, j) = (\iota(c, j), \sigma(c, j)),$$

where $\rho_\alpha(\cdot)$ is the negacyclic coefficient shift by offset α. Each term is a scalar–polynomial product (one plaintext scalar applied to both RLWE components), *not* a plaintext–ciphertext polynomial multiplication. Jaguar then applies ciphertext-side truncation and power-of-two reduction:

$$Y_{o,j} = \mathrm{MaskReduce}(\mathrm{Trunc}_f(A_{o,j})).$$

The output crosses the HE→2PC boundary already at target precision (Section 4). The kernel covers all CNN convolution types: pointwise sets $\Omega_K = \{(0, 0)\}$ (channel-wise scalar accumulation with no spatial shifts); dense 3×3 accumulates nine shifted neighborhoods per input channel; depthwise uses the same shifted accumulation pattern but with $\mathcal{C}(o) = \{o\}$. Detailed algorithms are in Appendix J.1.

Cost. Let $T_o = |\mathcal{C}(o)|K^2$. Each output block costs $2NT_o$ scalar multiplications (two RLWE components$\times N$ coefficients$\times$ T_o shifted scalar terms), plus linear-time shifts, additions, and masks. Summed over B output blocks per channel, SPA-CONV performs $2NBK^2 \sum_{o=0}^{C_{\mathrm{out}}-1} |\mathcal{C}(o)|$ coefficient-wise scalar multiplications, scaling as $\Theta(NBC_{\mathrm{out}}C_{\mathrm{in}}K^2)$ for dense, $\Theta(NBC_{\mathrm{out}}C_{\mathrm{in}})$ for pointwise, and $\Theta(NBC_{\mathrm{out}}K^2)$ for depthwise (Table 2). The HE-PMult and HE-Rot columns are zero for Jaguar across all convolution types. Cost shifts entirely to scalar–polynomial products in a power-of-two ring, where per-operation cost is dominated by integer multiply-and-mask rather than by NTT and modular prime reduction.

3.4 NTT-Assisted Decryption

A power-of-two q removes NTT from the linear path, but the client must still compute one polynomial product $\hat{c}_1' \cdot \mathsf{sk}$ during decryption (Algorithm 2)—part of BFV decryption itself. Naively doing this in the power-of-two ring without NTT costs $O(N^2)$, which would dominate at $N = 2048$.

Construction. Jaguar performs $\hat{c}_1' \cdot \mathsf{sk}$ via NTT *over an auxiliary NTT-friendly* prime q_{ntt} chosen large enough that the integer-valued coefficients of the product do not wrap modulo q_{ntt}. The relevant coefficients are then mapped back to residues modulo 2^Q. The auxiliary prime never appears in any ciphertext, plaintext, share, or protocol message—it is purely a client-side computational shortcut.

Choosing q_{ntt}. With ternary secret keys of Hamming weight $h = \|\mathsf{sk}\|_0 \le N$, after f-bit ciphertext truncation the coefficients of $\hat{c}_1'$ have magnitude below 2^{Q-f}, so each coefficient of $\hat{c}_1' \cdot \mathsf{sk}$ is bounded by $h \cdot 2^{Q-f}$. The sufficient condition is $q_{\mathrm{ntt}}/2 > h \cdot 2^{Q-f}$. For $N = 2048, Q = 54, f = 9$, we have $h \le 2^{11}$, giving a bound of 2^{56}; a 60-bit NTT-friendly prime suffices, with margin.

This preserves Jaguar's design: the ciphertext modulus, share domain, server-side arithmetic, and all messages on the wire remain power-of-two, so Theorem 1 still applies, $\mathbb{Z}_{2^\ell}$ alignment holds, and the

Inputs/outputs. ($\mathcal{T}$: input tensor, $\mathcal{W}$: weight tensor.) Client holds $(\langle\mathcal{T}\rangle_C, \mathsf{sk})$; server holds $(\langle\mathcal{T}\rangle_S, \mathcal{W})$, with $\langle\mathcal{T}\rangle_C + \langle\mathcal{T}\rangle_S = \mathcal{T}$ over $\mathbb{Z}_{2^\ell}$. Output: $(\langle\mathcal{T}'\rangle_C, \langle\mathcal{T}'\rangle_S)$ for $\mathcal{T}' = \mathrm{Conv2D}(\mathcal{T}, \mathcal{W}; s_{\mathrm{stride}}) \gg f$.
Public. $pp = (N, Q, P, K, s_{\mathrm{stride}}, p_{\mathrm{pad}})$ and maps Λ and Φ, where Λ defines input packing and $\Phi(o, \xi) = (\gamma, \kappa_{o,\xi})$ identifies the source ciphertext and coefficient.

1: **Client encryption.** Encode $\langle\mathcal{T}\rangle_C$ by Λ, encrypt the input blocks, and send $\{\mathsf{CT}_i^C\}$ to the server.
2: **Server-side SPA-CONV.** Evaluate SPA-CONV on $\{\mathsf{CT}_i^C\}$ with plaintext filters. Inside SPA-CONV, the server adds the server input-share contribution from $\langle\mathcal{T}\rangle_S$ and $\mathcal{W}$ at pre-truncation precision, then applies ciphertext-side truncation and power-of-two reduction. This produces target-precision ciphertexts $\{\mathsf{CT}_j^{\mathrm{tr}}\}$ encrypting $\mathrm{Conv2D}(\mathcal{T}, \mathcal{W}; s_{\mathrm{stride}}) \gg f$. For the first layer, $\langle\mathcal{T}\rangle_S = 0$.
3: **Random-share masking and grouped export.** For each logical output (o, ξ), sample a output random mask $r_{o,\xi} \in \mathbb{Z}_{2^{P-f}}$ and homomorphically mask the corresponding exported constant term so that the client decrypts $\mathcal{T}'[o, \xi] - r_{o,\xi}$. Group all valid output coefficients by source ciphertext. Each group contains one shared linear component $\hat{a}_\gamma$, masked constants $\{\widetilde{b}_{o,\xi}\}$, coefficient indices $\{\kappa_{o,\xi}\}$, and output metadata. Send the groups to the client; the server keeps $r_{o,\xi}$ as its output share.
4: **Client reconstruction.** For each group, compute $\hat{d}_\gamma(X) = \hat{a}_\gamma(X)\mathsf{sk}(X)$ once. For each stored $\kappa_{o,\xi}$, combine $\hat{d}_\gamma[\kappa_{o,\xi}]$ with $\widetilde{b}_{o,\xi}$ and decode by power-of-two rounding/shifting to obtain $\langle\mathcal{T}'\rangle_C[o, \xi] = \mathcal{T}'[o, \xi] - r_{o,\xi}$. The server share is $\langle\mathcal{T}'\rangle_S[o, \xi] = r_{o,\xi}$.

Figure 4: Jaguar convolution protocol.

server still pays no NTT cost. We simply use the right tool for the right job—power-of-two arithmetic where it matters for the protocol; NTT where it matters for asymptotic efficiency of a local product. Decryption stays $O(N \log N)$.

4 Jaguar Protocol

4.1 Protocol Overview

Across layers, activations are maintained as additive shares over $\mathbb{Z}_{2^\ell}$. To enter a linear layer, the client encrypts its share; the server runs SPA-CONV on the encrypted share, applies ciphertext-side truncation, masks the target-precision ciphertexts with fresh random output shares, and exports additive shares to the 2PC nonlinear protocol. After the nonlinearity, the client re-encrypts for the next linear layer (Figure 3).

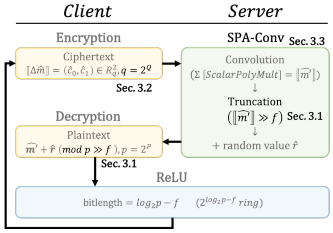


Figure 3: Overview of Jaguar.

4.2 Convolution Protocol

Figure 4 gives the secure convolution protocol: the local SPA-CONV kernel wrapped by client encryption before the linear layer and grouped HE-to-share export afterward.

Jaguar adopts Cheetah's [12] LWE export compression: LWE ciphertexts extracted from the same RLWE ciphertext share their component, transmitted only once. The crucial difference from Cheetah is that exported values are *already at target precision*, so the subsequent nonlinear protocol does not invoke a separate truncation.

Fully-connected and attention compatibility. Convolution accounts for >98 % of linear-layer latency in our evaluated CNNs, so our end-to-end implementation focuses on this path and handles the final classifier through a one-time conversion to Cheetah's FC backend [20, 12]. Beyond the CNN workloads evaluated end to end, Jaguar's static-weight linear-to-share arithmetic also applies to attention-based architectures. In particular, token-wise projections apply the same learned linear transformation independently to each token and are algebraically equivalent to a 1×1 convolution [34]. Thus, QKV and output projections can use SPA-CONV and export additive shares to a compatible attention protocol. In contrast, QK^T and AV have two input-dependent operands, and Softmax is nonlinear; these operations require separate protocols. Section 5.4 evaluates the static-weight kernels, with mapping and composition details in Appendix G.

4.3 Nonlinear Protocols

Jaguar uses the same ring-based 2PC primitives as prior hybrid systems for nonlinearities [11, 12]; the difference is the *representation* of their inputs. Because the HE linear path applies exact truncation,

Table 3: Communication costs of (a) truncation and (b) ReLU. ℓ: bitwidth of the pre-truncation shared value. f: truncation amount. b_r: Millionaire radix block size. λ: security parameter.

(a) Truncation

Protocol	Small err.	Big err.	Comm. bits
CrypTFlow2	×	×	$\lambda(\ell+f+2)+19\ell+14f$
Cheetah-F	×	×	$16\ell+11f$
Cheetah	○	×	13ℓ
Jaguar	×	×	**0**

Cheetah-F denotes faithful truncation.

(b) ReLU

Scheme	Input	Cmp. width	ReLU comm. bits
Cheetah	ℓ	$b_r\lfloor(\ell-\frac{3f}{2}-1)/b_r\rfloor$	$<11b_r\lfloor(\ell-\frac{3f}{2}-1)/b_r\rfloor$ $+2\ell+1$
Jaguar	$\ell-f$	$b_r\lfloor(\ell-2f-1)/b_r\rfloor$	$<11b_r\lfloor(\ell-2f-1)/b_r\rfloor$ $+2(\ell-f)+1$

the nonlinear protocol receives target-precision shares. For ReLU, $\mathrm{ReLU}(x) \gg f = \mathrm{ReLU}(x \gg f)$, so Jaguar invokes the existing ReLU primitive directly on shares of $x \gg f$. Table 3 quantifies the saving: (a) the post-ReLU truncation protocol—13ℓ to $\lambda(\ell+f+2)+19\ell+14f$ communication bits in prior systems—is *eliminated*; (b) ReLU communication itself shrinks because the input bitwidth changes from ℓ to $\ell-f$, reducing both comparison width and share payload. For MobileNetV2 [4], ReLU6 uses the same target-precision interface with the clipping threshold at the target scale.

5 Experiments

5.1 Experiment Setup

HE parameters. The CNN experiments use BFV [24] with $N = 2048$, $q = 2^{54}$, $p = 2^{30}$, $\lambda = 128$, $f = 9$. Parameter selection for preserving fixed-point accuracy is detailed in Appendix E. The operator benchmarks in Section 5.4 use the configurations specified in Appendix G.

Models and dataset. ImageNet [35] with pretrained torchvision [36] ResNet-18 [1], ResNet-50 [1], and MobileNetV2 [4].

Implementation. Jaguar is built on SEAL [30]. Experiments run on dual Intel Xeon Gold 6250 CPUs at 3.90 GHz and 128 GB RAM. Linux `tc` emulates LAN (384 MB/s, 0.3 ms RTT) and WAN (62.5 MB/s, 20 ms RTT). We compare against OpenCheetah [20] and Rhombus [37] using their public implementations with default HE parameters and matched fixed-point scale and 2PC share bitwidth. Additional comparisons with Panther [38] and SecONNds [39], including integration with SecONNds' nonlinear protocols, are provided in Appendix H.

5.2 Microbenchmarks

Table 4: Compute-only convolution microbenchmark with AVX disabled. Speedup is Cheetah over Jaguar. (GM: geometric mean) Best/worst cases are shapes $(W, C_{\mathrm{in}}, C_{\mathrm{out}}, K)$, where W is the square spatial width and K is the kernel size; for depthwise, $C_{\mathrm{out}} = C_{\mathrm{in}}$.

Full per-shape results in Appendix F.1.

Conv. type	Layout	#Cases	GM speedup	Range	Best case	Worst case
Dense 3×3	Packed	10	2.47×	0.29–9.64×	$(32,128,128,3)$	$(4,512,512,3)$
	Split	4	17.81×	12.81–37.75×	$(64,64,64,3)$	$(224,64,64,3)$
Pointwise 1×1	Packed	9	11.82×	3.04–46.73×	$(28,128,512,1)$	$(7,320,1280,1)$
	Split	4	10.76×	9.91–12.99×	$(56,64,256,1)$	$(112,32,16,1)$
Depthwise 3×3	Packed	6	11.65×	9.64–13.04×	$(14,96,96,3)$	$(14,576,576,3)$
	Split	3	5.45×	4.29–6.65×	$(56,24,24,3)$	$(112,32,32,3)$

Convolution kernel. We benchmark representative convolution shapes from the evaluated models, measuring server-side SPA-CONV computation only (single thread, AVX disabled).

The benefit depends on convolution type and packing amortization (Table 4). **Split dense** (17.81× GM) gains the most: the 3×3 channel-mixing repeats across many spatial blocks, so the baseline pays NTT/PMult/reduction overhead each time, while SPA-CONV applies block-local shifted scalar accumulation. **Pointwise** (11.82×/10.76× GM packed/split) is consistently favorable because $K = 1$ removes spatial shifts, reducing SPA-CONV to channel-wise scalar accumulation. **Packed depthwise**

(11.65× GM) benefits because each output channel uses only $K^2 = 9$ shifted terms instead of $C_{\text{in}}K^2$, while the baseline still invokes a generic PMult-heavy path; **split depthwise** (5.45× GM) is smaller because more spatial blocks must be processed without cross-channel mixing to amortize the repeated block-local work. **Packed dense** (2.47× GM) shows the widest range: when the spatial footprint is large, only a few channels fit per ciphertext and Jaguar is much faster; when the footprint is small, the baseline strongly amortizes polynomial multiplication across many packed channels, sometimes reversing the advantage. These below-1× cases do not dominate full-network convolution time.

Nonlinear stage. Table 5 measures the ReLU and truncation layers from ResNet-50. ReLU latency drops by 2.24× under LAN and 1.61× under WAN, with 1.83× lower communication, because the ReLU comparison runs at a smaller input bitwidth. The truncation gain is larger: replacing the interactive 2PC truncation with local HE-side truncation gives 5.58× lower latency under LAN and 16.22× under WAN (the WAN gain is dominated by the eliminated round trip), while reducing truncation communication to zero. The combined ReLU+truncation stage improves by 2.32×(LAN) / 1.80×(WAN) in latency and 1.98× in communication.

Table 5: Network-level nonlinear-stage breakdown for ResNet-50.

Layer	Method	Latency (s)		Comm. (MiB)
		LAN	WAN	
ReLU	Cheetah	17.70	24.20	271.69
	Jaguar	**7.89**	**15.02**	**148.61**
		2.24×	1.61×	1.83×
Truncation	Cheetah	1.08	3.21	22.91
	Jaguar	**0.19**	**0.20**	**0.00**
		5.58×	16.22×	Eliminated

5.3 End-to-End Inference

Table 6: End-to-end latency and communication with AVX disabled. Speedup and communication reduction are computed relative to Cheetah. (AVX-enabled results are in Appendix F.2.)

Model	Protocol	LAN		WAN		Communication	
		Latency (s)	Speedup	Latency (s)	Speedup	Comm. (MiB)	Reduction
ResNet18	Cheetah	30.30	1.00×	36.94	1.00×	366.94	1.00×
	Rhombus	29.04	1.04×	36.24	1.02×	344.01	1.07×
	Jaguar	**8.64**	**3.51**×	**15.94**	**2.32**×	**315.02**	**1.16**×
ResNet50	Cheetah	120.82	1.00×	142.72	1.00×	1335.63	1.00×
	Rhombus	96.77	1.25×	111.44	1.28×	**750.22**	**1.78**×
	Jaguar	**32.44**	**3.72**×	**51.50**	**2.77**×	962.00	1.39×
MobileNetV2	Cheetah	46.99	1.00×	67.25	1.00×	1042.62	1.00×
	Rhombus	58.98	0.80×	77.19	0.87×	836.81	1.25×
	Jaguar	**18.41**	**2.55**×	**32.42**	**2.07**×	**592.50**	**1.76**×

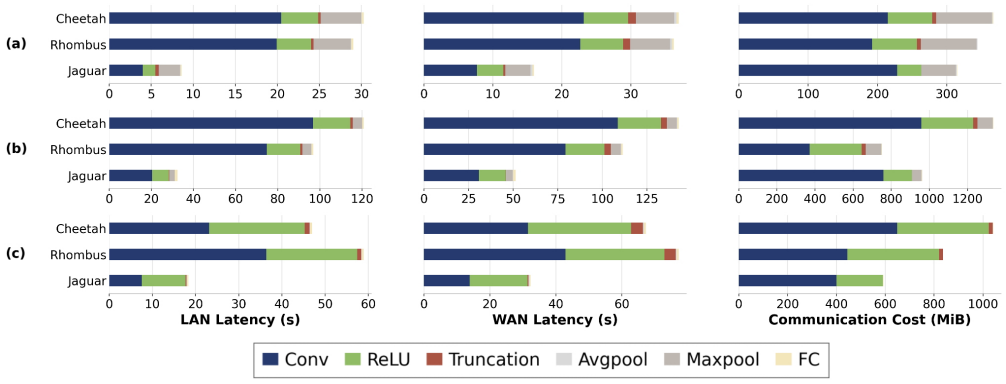


Figure 5: End-to-end latency and communication breakdown across ImageNet-scale CNNs: (a) ResNet-18, (b) ResNet-50, and (c) MobileNetV2.

We evaluate end-to-end secure inference using four threads (Table 6). Latency includes local protocol execution and network transfer; communication counts all messages exchanged in one inference.

Jaguar achieves the lowest end-to-end latency on every model: 3.51×/3.72×/2.55× over Cheetah on LAN; 2.32×/2.77×/2.07× on WAN; communication 1.16–1.76× lower than Cheetah. The improvement holds across CNNs with different mixtures of dense, pointwise, and depthwise convolutions.

ResNet-50 communication vs. Rhombus. Rhombus reports 750 MiB on ResNet-50, lower than Jaguar's 962 MiB. Rhombus is *specifically optimized* for the pointwise-convolution and MVM components that dominate ResNet-50's communication; its integration replaces these in Cheetah but leaves dense 3×3 convolution and the nonlinear stage untouched. This is why Rhombus's communication advantage on ResNet-50 is largest in absolute terms but **does not translate proportionally into latency**: Rhombus is $1.25\times$ faster than Cheetah on LAN; Jaguar is $3.72\times$. Jaguar trades a portion of communication on ResNet-50 for substantially lower compute across the *entire* convolution path and a fundamentally cheaper nonlinear stage. On more diverse layer mixes (ResNet-18, MobileNetV2), Jaguar achieves both lower latency *and* lower communication than Rhombus.

End-to-end breakdown. Across all models, the dominant latency reduction comes from the linear convolution path, with the relative contribution differing by architecture (Figure 5). The nonlinear side adds reduced ReLU input bitwidth and eliminated post-ReLU truncation, improving both compute-dominated LAN and communication-sensitive WAN latency.

AVX-512. Jaguar's power-of-two arithmetic is well-suited to AVX-512 [40, 41], since modular reduction and ciphertext-side truncation map to vectorized bit masking and arithmetic shifts. With AVX-512, Jaguar's end-to-end LAN latencies reach 7.90 s/25.55 s/16.83 s on ResNet-18/ResNet-50/MobileNetV2, respectively (Appendix F.2).

5.4 Extensions to Attention Architectures

To assess this mapping beyond CNN workloads, we benchmark the static-weight projections surrounding the input-dependent attention core. Fused QKV projections produce queries, keys, and values before attention, while output projections transform the aggregated values afterward; both apply fixed channel-mixing weights independently to each token.

We evaluate eight projection shapes covering 44 fused QKV/output-projection occurrences in MaxViT-T [42]. With matched 37-bit plaintext arithmetic, nine fractional bits, and server-side HE linear timing, Jaguar achieves a $2.24\times$ occurrence-weighted speedup over Cheetah; individual ratios range from $0.84\times$ to $11.03\times$, with six of eight shapes faster. Sparse BERT-base projections show 2.15–$2.94\times$ speedup at 1% weight density. These are kernel benchmarks, not end-to-end attention measurements; QK^{T}, Softmax, and AV remain outside SPA-CONV. Appendix G reports the setup, per-shape results, and share-level composition.

6 Conclusion

We presented Jaguar, a hybrid HE/2PC private neural network inference system built on a power-of-two ciphertext ring. The single design choice $q = 2^Q, p = 2^P$ unlocks three contributions that prior work could not realize on the conventional NTT-prime backend: SPA-CONV, which evaluates convolution as scalar–polynomial accumulation without polynomial multiplication, NTT, or modular prime reduction in the linear path; exact pre-ReLU ciphertext-side truncation, which removes the post-ReLU truncation protocol and reduces ReLU input bitwidth; and NTT-assisted decryption, which recovers $O(N \log N)$ client reconstruction via an auxiliary NTT prime without disturbing the protocol substrate. Experiments on ResNet-18, ResNet-50, and MobileNetV2 show 2.07–$3.72\times$ lower end-to-end latency than Cheetah and 2.16–$3.36\times$ lower than Rhombus under both LAN and WAN settings. Jaguar shows that *changing the HE arithmetic regime itself*—rather than only optimizing packing or communication on top of the conventional backend—can unlock substantial gains for practical private neural network inference.

Acknowledgments

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. RS-2025-02217656, Privacy Preserving AI Integrated System for K-Cloud based on Homomorphic Encryption), by K-CHIPS (Korea Collaborative & High-tech Initiative for Prospective Semiconductor Research) (2410018409, RS-2026-25524122, 26086-45FC) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea), and by IITP under the Graduate School of Artificial Intelligence Semiconductor (IITP-2026-RS-2023-00256081) grant funded by the Korea government (MSIT).

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[2] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017. doi: 10.1038/nature21056.

[3] Georgios A. Kaissis, Alexander Ziller, Jonathan Passerat-Palmbach, Théo Ryffel, Dmitrii Usynin, Andrew Trask, Ionésio Lima, Jason Mancuso, Friederike Jungmann, Marc-Matthias Steinborn, Rickmer Braren, Marcus Makowski, Daniel Rueckert, et al. End-to-end privacy preserving deep learning on multi-institutional medical imaging. *Nature Machine Intelligence*, 3(6):473–484, 2021. doi: 10.1038/s42256-021-00337-8.

[4] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. doi: 10.1109/CVPR.2018.00474.

[5] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, and Song Han. MCUNet: Tiny deep learning on IoT devices. In *Advances in Neural Information Processing Systems*, volume 33, pages 11711–11722, 2020.

[6] Pete Warden and Daniel Situnayake. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, 2019. ISBN 9781492052043.

[7] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38. IEEE, 2017. doi: 10.1109/SP.2017.12.

[8] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 619–631, 2017. doi: 10.1145/3133956.3134056.

[9] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security symposium (USENIX Security 18)*, pages 1651–1669, 2018.

[10] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2505–2522, 2020.

[11] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow2: Practical 2-party secure inference. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 325–342, 2020.

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

[13] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 201–210. PMLR, 2016.

[14] Alon Brutzkus, Oren Elisha, and Ran Gilad-Bachrach. Low latency privacy preserving inference. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 812–821. PMLR, 2019.

[15] Eunsang Lee, Joon-Woo Lee, Junghyun Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and Woosuk Choi. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 12403–12422. PMLR, 2022.

[16] Jae Hyung Ju, Jaiyoung Park, Jongmin Kim, Minsik Kang, Donghwan Kim, Jung Hee Cheon, and Jung Ho Ahn. Neujeans: Private neural network inference with joint optimization of convolution and fhe bootstrapping. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, pages 4361–4375, 2024. doi: 10.1145/3658644.3690375.

[17] Alexandru Stoian, Jordan Fréry, Roman Bredehoft, Luis Montero, Celia Kherfallah, and Benoît Chevallier-Mames. Deep neural networks for encrypted inference with tfhe. Cryptology ePrint Archive, Paper 2023/257, 2023. URL `https://eprint.iacr.org/2023/257`.

[18] Hyeri Roh, Jinsu Yeo, Yeongil Ko, Gu-Yeon Wei, David Brooks, and Woo-Seok Choi. Flash: A hybrid private inference protocol for deep CNNs with high accuracy and low latency on cpu. *arXiv preprint arXiv:2401.16732*, 2024.

[19] Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. BOLT: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 4753–4771, 2024. doi: 10.1109/SP54263.2024.00130.

[20] Alibaba Gemini Lab. OpenCheetah: Proof-of-concept implementation for Cheetah. `https://github.com/Alibaba-Gemini-Lab/OpenCheetah`, 2022. Accessed: 2026-04-26.

[21] Woo-Seok Choi, Brandon Reagen, Gu-Yeon Wei, and David Brooks. Impala: Low-latency, communication-efficient private deep learning inference. *arXiv preprint arXiv:2205.06437*, 2022.

[22] Kanav Gupta, Deepak Kumaraswamy, Nishanth Chandran, and Divya Gupta. LLAMA: A low latency math library for secure inference. *Proceedings on Privacy Enhancing Technologies*, 2022(4):274–294, 2022.

[23] Jiaxing He, Kang Yang, Guofeng Tang, Zhangjie Huang, Li Lin, Changzheng Wei, Ying Yan, and Wei Wang. Rhombus: Fast homomorphic matrix-vector multiplication for secure two-party inference. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 2490—2504, 2024.

[24] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012. URL `https://eprint.iacr.org/2012/144`.

[25] Hyeri Roh and Woo-Seok Choi. Hyena: Optimizing homomorphically encrypted convolution for private cnn inference. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2024.

[26] J. M. Pollard. The fast fourier transform in a finite field. *Mathematics of Computation*, 25(114): 365–374, 1971.

[27] Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. In *Advances in Neural Information Processing Systems 35*, pages 15718–15731, 2022.

[28] Tianshi Xu, Meng Li, Runsheng Wang, and Ru Huang. Falcon: Accelerating homomorphically encrypted convolutions for efficient private mobile network inference. In *2023 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–9, 2023. doi: 10.1109/ICCAD57390.2023.10323672.

[29] Tianshi Xu, Lemeng Wu, Runsheng Wang, and Meng Li. Privcirnet: Efficient private inference via block circulant transformation. In *Advances in Neural Information Processing Systems 37*, 2024.

[30] SEAL. Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL, January 2023. Microsoft Research, Redmond, WA.

[31] HomomorphicEncryption.org Standardization Consortium. Homomorphic encryption standard. Version 1.1, 2024. URL https://homomorphicencryption.org/wp-content/uploads/2024/08/Homomorphic-Encryption-Standard-v1.1.pdf.

[32] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2013. doi: 10.1007/978-3-642-38348-9_3.

[33] Kim Laine. *Simple Encrypted Arithmetic Library 2.3.1*. Microsoft Research, 2017. URL https://www.microsoft.com/en-us/research/wp-content/uploads/2017/11/sealmanual-2-3-1.pdf.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[36] PyTorch Contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2024. Accessed: 2026-04-30.

[37] Jiaxing He. RhombusEnd2End: Public implementation of rhombus. https://github.com/2646jx/RhombusEnd2End, 2025. MIT License. Accessed: 2026-04-30.

[38] Jun Feng, Yefan Wu, Hong Sun, Shunli Zhang, and Debin Liu. Panther: Practical secure two-party neural network inference. *IEEE Transactions on Information Forensics and Security*, 20:1149–1162, 2025. doi: 10.1109/TIFS.2025.3526063.

[39] Shashank Balla. SecONNds: Secure outsourced neural network inference on ImageNet. *arXiv preprint arXiv:2506.11586*, 2025. doi: 10.48550/arXiv.2506.11586.

[40] Intel Corporation. Intel advanced vector extensions 512 (Intel AVX-512) overview. https://www.intel.com/content/www/us/en/architecture-and-technology/avx-512-overview.html, 2024. Accessed: 2026-04-27.

[41] Intel Corporation. Intel intrinsics guide. https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html, 2024. Version 3.6.9, accessed: 2026-04-27.

[42] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. MaxViT: Multi-axis vision transformer. In *Computer Vision – ECCV 2022*, volume 13684 of *Lecture Notes in Computer Science*, pages 459–479. Springer, 2022. doi: 10.1007/978-3-031-20053-3_27.

[43] Shashank Balla. SecONNds: Open-source implementation. https://github.com/shashankballa/SecONNds.

A Notations

Conventions. Ring elements are italic polynomials, e.g., $\hat{x}(X) \in R_q$ or short for $\hat{x} \in R_q$, and $\hat{x}[i]$ denotes the i-th coefficient. Lower-case letters with "hat" symbols denote polynomials, bold lower-case symbols denote vectors/lists, bold upper-case symbols denote matrices, and calligraphic symbols denote tensors.

Symbol	Meaning	Symbol	Meaning
General and cryptographic parameters			
$[n]$	$\{0, \ldots, n-1\}$ for an integer n.	N	Ring degree / number of polynomial coefficients.
R_q	$\mathbb{Z}_q[X]/(X^N+1)$.	$q = 2^Q$	Ciphertext modulus and bitwidth.
$p = 2^P$	Plaintext modulus and bitwidth.	Δ	BFV scale, $\Delta = q/p = 2^{Q-P}$.
q_{ntt}	Auxiliary NTT-friendly prime for client-side reconstruction.	λ	Security parameter.
ℓ	Bitwidth of the additive-share ring for the pre-truncation value; i.e. shares are over $\mathbb{Z}_{2^\ell}$.	f	Fixed-point fractional bits; main truncation amount.
b_r	Millionaire comparison radix block size.	L	Number of RNS limbs in prime-modulus baselines.
Γ_{mul}	Estimated 64-bit multiplication cost for HE-PMult.	Γ_{rot}	Estimated 64-bit multiplication cost for HE-Rot.
h	Hamming weight of the ternary secret key.		
Ring elements, ciphertexts, and shares			
$\hat{x}(X) \in R_q$	Ring element / polynomial.	$\hat{x}[i]$	i-th coefficient of $x(X)$.
$\rho_\eta(\hat{x})$	Negacyclic shift $X^\eta \hat{x} \bmod (X^N+1)$.	$\mathsf{sk} = \hat{s}(X)$	BFV secret-key polynomial.
$\hat{\epsilon}$	BFV noise polynomial.	χ_e	BFV error distribution.
$\hat{m}$	Plaintext polynomial.	$[\![\hat{m}]\!]$	BFV encryption of $\hat{m}$.
$(\hat{c}_0, \hat{c}_1)$	Fresh BFV ciphertext components.	$(\hat{c}'_0, \hat{c}'_1)$	Ciphertext components after truncation.
$\langle x \rangle$	Additive sharing of x over $\mathbb{Z}_{2^\ell}$.	$\langle x \rangle_C, \langle x \rangle_S$	Client and server shares.
$r_{o,\xi}$	Server random mask/share for output (o, ξ).	$\mathbf{r}$	Vector of random masks.
Convolution and SPA-CONV			
$\mathcal{T}$	Input activation tensor.	$\mathcal{W}$	Convolution weight tensor.
H, W	Input height and width.	H', W'	Output height and width.
K	Kernel size; kernel is $K \times K$.	$C_{\mathrm{in}}, C_{\mathrm{out}}$	Input/output channel counts.
Ω_K	Spatial offsets $\{-r, \ldots, r\}^2$.	(u, v)	Spatial offset in Ω_K.
s_{stride}	Convolution stride.	p_{pad}	Padding size.
W_{pp}	Padded row width.	$\delta(u, v)$	Spatial coefficient offset $uW_{\mathrm{pad}} + v$.
Λ	Input packing map.	$\Lambda(c, j) = (\iota, \sigma)$	Source ciphertext index and coefficient offset.
CT_t	t-th encrypted input block.	M	Number of encrypted input blocks.
B	Output ciphertext blocks per output channel.	j	Output block index.
ν	Split factor.	o	Output channel index.
$\mathcal{C}(o)$	Contributing input-channel set for output o.	T_o	Number of shifted scalar terms, $\lvert\mathcal{C}(o)\rvert K^2$.
$A_{o,j}$	Raw encrypted output block.	$Y_{o,j}$	Truncated/reduced encrypted output block.
$\mathrm{Trunc}_\tau(\cdot)$	Ciphertext-side τ-bit truncation.	$\mathrm{MaskReduce}(\cdot)$	Power-of-two modular reduction by masking.
Grouped export and reconstruction			
Φ	Output extraction map.	$\Phi(o, \xi) = (\gamma, \kappa)$	Source ciphertext and coefficient index.

ξ	Logical output spatial position.	κ	Coefficient index inside an RLWE ciphertext.
γ	Source RLWE ciphertext group index.	$\mathcal{P}_\gamma$	Logical outputs extracted from source γ.
G_γ	Grouped export record for source γ.	$\mathcal{G}$	Set/list of grouped export records.
G	Number of non-empty export groups.	m_g	Number of exported coefficients in group g.
β_g	Transport bitwidth for group g.	E	Total exported values, $\sum_g m_g = C_{\text{out}}H'W'$.
$\hat{a}_\gamma(X)$	Shared RLWE linear component for group γ.	$\hat{b}_\gamma(X)$	RLWE constant component for group γ.
$\mathbf{b}_\gamma$	Vector of exported constant coefficients.	$\boldsymbol{\kappa}_\gamma$	Vector of coefficient indices.
$\mathbf{ch}_\gamma$	Vector of output-channel metadata.	$\mathbf{off}_\gamma$	Vector of output-position metadata.
$\hat{d}_\gamma(X)$	Client-side product $a_\gamma(X)\mathsf{sk}(X)$.	u_i	Reconstructed masked coefficient before decoding.
z_i	Decoded client share for one exported value.	$\mathbf{z}$	Vector of decoded client shares.

B Proof of Theorem 1

Theorem 1. *Algorithm 1 implements exact f-bit truncation, decryptable by Algorithm 2, provided that the ciphertext noise satisfies $\|\hat{\epsilon}\|_\infty < \Delta/2 - (N+1)2^f$. (Proof in Appendix B.)*

Lemma 2 (Unsigned shifts over power-of-two residues). Let $q = 2^Q$ and $q_f = 2^{Q-f}$ with $0 < f \le Q$. For any integer $\tilde{c} \in [-2^{Q-1}, 2^{Q-1})$, let $c = \tilde{c} \bmod q \in [0, q)$ be its canonical unsigned residue. Then

$$\left\lfloor \frac{c}{2^f} \right\rfloor \equiv \left\lfloor \frac{\tilde{c}}{2^f} \right\rfloor \pmod{q_f}.$$

Thus, a logical right shift of the canonical unsigned residue modulo 2^Q represents the arithmetic right shift of the signed centered representative modulo 2^{Q-f}.

Proof of Lemma 2. If $\tilde{c} \ge 0$, then $c = \tilde{c}$ and the claim is immediate. If $\tilde{c} < 0$, then $c = 2^Q + \tilde{c}$. Since 2^f divides 2^Q,

$$\left\lfloor \frac{c}{2^f} \right\rfloor = \left\lfloor \frac{2^Q + \tilde{c}}{2^f} \right\rfloor = 2^{Q-f} + \left\lfloor \frac{\tilde{c}}{2^f} \right\rfloor.$$

Therefore,

$$\left\lfloor \frac{c}{2^f} \right\rfloor \equiv \left\lfloor \frac{\tilde{c}}{2^f} \right\rfloor \pmod{2^{Q-f}},$$

which proves the claim. □

Proof of Theorem 1. By Lemma 2, the logical right shifts used in Algorithm 1 on canonical unsigned residues are equivalent, modulo $q_f = 2^{Q-f}$, to arithmetic right shifts of the corresponding signed centered representatives. Hence, below we analyze $\hat{c}'_0$ and $\hat{c}'_1$ as coefficient-wise arithmetic shifts of $\hat{c}_0$ and $\hat{c}_1$ in the reduced ring R_{q_f}.

Since $(\hat{c}_0, \hat{c}_1) \in R_q^2$ encrypts $\hat{m}_0 \in R_p$, we have $[\hat{c}_0 + \hat{c}_1\hat{s}]_q = \Delta\hat{m}_0 + \hat{\epsilon}$ for some noise polynomial $\hat{\epsilon}$. Hence, for some integer-coefficient polynomial $\hat{r}$,

$$\frac{\hat{c}_0 + \hat{c}_1\hat{s}}{2^f} = \frac{\Delta\hat{m}_0}{2^f} + \frac{\hat{\epsilon}}{2^f} + \frac{q}{2^f}\hat{r}$$

Using $\lfloor x \rfloor \le x < \lfloor x \rfloor + 1$ for $\forall x \in \mathbb{R}$, and $\hat{s}[i] \in \{-1, 0, 1\}$ for $\forall i \in [N]$,

$$(\frac{\hat{c}_0 + \hat{c}_1\hat{s}}{2^f})[i] - N - 1 < (\hat{c}'_0 + \hat{c}'_1\hat{s})[i] < (\frac{\hat{c}_0 + \hat{c}_1\hat{s}}{2^f})[i] + N, \quad \forall i \in [N]$$

$$\Rightarrow\ (\frac{\Delta\hat{m}_0}{2^f} + \frac{\hat{\epsilon}}{2^f} + \frac{q}{2^f}\hat{r})[i] - N - 1 < (\hat{c}'_0 + \hat{c}'_1\hat{s})[i] < (\frac{\Delta\hat{m}_0}{2^f} + \frac{\hat{\epsilon}}{2^f} + \frac{q}{2^f}\hat{r})[i] + N$$

$$\Rightarrow\ (\hat{m}_0+\frac{\hat{\epsilon}}{\Delta})[i]-\frac{(N+1)2^f}{\Delta}<(\frac{2^f}{\Delta}[\hat{c}'_0+\hat{c}'_1\hat{s}]_{q_f})[i]<(\hat{m}_0+\frac{\hat{\epsilon}}{\Delta})[i]+\frac{2^f}{\Delta}N$$

As long as $||\hat{\epsilon}||_\infty < \Delta/2-(N+1)2^f$, we get $\lfloor\frac{2^f}{\Delta}[\hat{c}'_0+\hat{c}'_1\hat{s}]_{q_f}\rceil=\hat{m}_0$. Therefore, Algorithm 2 returns the plaintext $\hat{m}_1$, where $\hat{m}_1[i]=(\hat{m}_0[i]\gg f)$. □

C Noise-Budget Analysis for Maximum-Accumulation Layers

To assess whether Jaguar's homomorphic convolution remains within the ciphertext-noise budget, we analyze the layers with the largest scalar–ciphertext accumulation in each evaluated network. For a fixed output ciphertext coefficient, these layers maximize the number of accumulated weighted ciphertext terms and therefore represent the most adverse candidates for noise growth in the SPA-CONV path.

The maximum accumulation counts are 4608 for ResNet-18 (layers 16, 18, and 19), 4608 for ResNet-50 (layers 44, 48, and 51), and 960 for MobileNetV2 (layers 44, 47, and 50). For each such layer, we reconstruct the BN-fused convolution weights exactly as in the Jaguar implementation. Let $w_{o,j}^{\text{centered}}$ denote the centered convolution weight and BNMul_o the BN scaling factor for output channel o. The fused integer weight is

$$w_{o,j}^{\text{fused}}=\mathrm{Round}\left(\frac{w_{o,j}^{\text{centered}}\cdot\mathrm{BNMul}_o}{2^{s_{\mathrm{BN}}}}\right),$$

where s_{BN} is the BN-fusion scale parameter used by the runtime.

Gaussian noise propagation. The noise polynomial coefficients of a fresh symmetrically-encrypted BFV ciphertext are independent, zero-mean Gaussian random variables,

$$n_{\text{in}}[j]\sim\mathcal{N}(0,\sigma_{\text{in}}^2),\qquad \sigma_{\text{in}}=3.2.$$

For output channel o, the accumulated pre-truncation output noise is

$$n_{\text{out}}[o]=\sum_j w_{o,j}^{\text{fused}}\,n_{\text{in}}[j].$$

Under the independence assumption,

$$\mathrm{Var}[n_{\text{out}}[o]]=\sigma_{\text{in}}^2\sum_j\left(w_{o,j}^{\text{fused}}\right)^2,\qquad \sigma_{\text{out}}[o]=\sigma_{\text{in}}\sqrt{\sum_j\left(w_{o,j}^{\text{fused}}\right)^2}.$$

Thus, the noise amplification of each output channel is governed by the ℓ_2 norm of its BN-fused weight vector, not by the accumulation count alone.

Consistent noise scale and threshold. The comparison must be performed at a consistent scale. Jaguar uses $(Q,P,f)=(54,30,9)$, so

$$\Delta=2^{Q-P}=2^{24}.$$

The pre-truncation BFV rounding margin is

$$B_{\text{dec}}=\Delta/2=2^{Q-P-1}=2^{23}.$$

The sufficient condition in Theorem 1 is slightly stronger because it also accounts for the component-wise shift error:

$$B_{\text{trunc}}=\Delta/2-(N+1)2^f=2^{23}-2049\cdot 2^9=7{,}339{,}520.$$

Equivalently, after the f-bit shift, the corresponding theorem-level threshold is

$$\frac{B_{\text{trunc}}}{2^f}=2^{Q-P-f-1}-(N+1)=14{,}335.$$

Since the noise standard deviation is reduced by the same factor 2^f after truncation, the effective margin is unchanged as long as both quantities are compared at the same scale. In this appendix, we compare pre-truncation noise estimates against the pre-truncation theorem bound B_{trunc}.

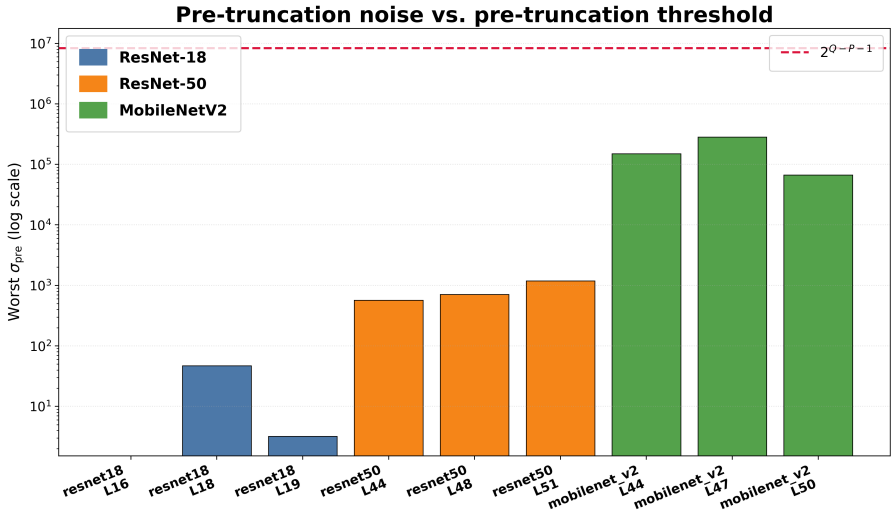
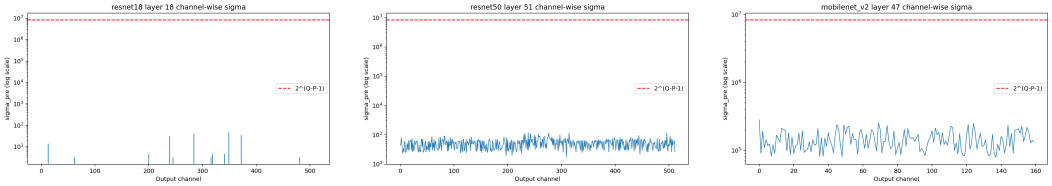


Figure 6: Worst-case pre-truncation output-noise standard deviation at the maximum scalar–ciphertext accumulation layers of Jaguar. Each bar corresponds to the output channel with the largest estimated σ_{out} in that layer. The dashed line marks the pre-truncation BFV rounding margin $B_{\text{dec}} = 2^{Q-P-1} = 2^{23}$; the theorem-level bound is $B_{\text{trunc}} = B_{\text{dec}} - (N+1)2^f$. All estimated worst-channel noise levels remain far below both boundaries.



(a) ResNet-18, layer 18 (b) ResNet-50, layer 51 (c) MobileNetV2, layer 47

Figure 7: Channel-wise pre-truncation output-noise standard deviation for representative maximum-accumulation layers. The variation across output channels reflects the ℓ_2 norms of the corresponding BN-fused weight vectors.

Results. The largest estimated pre-truncation standard deviations among the analyzed maximum-accumulation layers are 47.14 for ResNet-18, 1187.05 for ResNet-50, and 2.84×10^5 for MobileNetV2. Relative to $B_{\text{trunc}} = 7{,}339{,}520$, these correspond to margins of approximately $1.56 \times 10^5\sigma$, $6.18 \times 10^3\sigma$, and 25.9σ, respectively. The corresponding Gaussian tail probability

$$\Pr\left(|\mathcal{N}(0, \sigma_{\max}^2)| > B_{\text{trunc}}\right) = \operatorname{erfc}\left(\frac{B_{\text{trunc}}}{\sqrt{2}\sigma_{\max}}\right)$$

is numerically negligible; for the worst MobileNetV2 layer, it is approximately 9.3×10^{-148}.

MobileNetV2 exhibits the largest estimated noise growth, especially at layer 47. This is not caused by missing BN fusion. Rather, it results from a 960-term pointwise (1×1) accumulation combined with relatively large BN scaling coefficients, which increase the ℓ_2 norms of the BN-fused weights. In contrast, the ResNet layers accumulate more terms ($512 \times 3 \times 3 = 4608$), but their BN-fused weight norms are much smaller, so their propagated noise remains substantially lower.

This analysis provides a conservative sanity check for the most accumulation-heavy homomorphic convolution layers used in Jaguar: under the Gaussian propagation model, the BN-fused weight norms of the analyzed worst-case layers remain far from the theorem-level truncation boundary.

D Security and Parameter Selection

Jaguar instantiates BFV over the negacyclic ring $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ with a power-of-two ciphertext modulus $q = 2^Q$ and a power-of-two plaintext modulus $p = 2^P$. The parameter set used in our main evaluation is

$$(N, Q, P) = (2048, 54, 30), \qquad q = 2^{54}, \quad p = 2^{30}.$$

This appendix explains how these parameters are selected.

Security target. The security of BFV is determined by the concrete hardness of the underlying RLWE instance, which depends on the ring dimension, ciphertext modulus, secret distribution, and error distribution, rather than on whether the ciphertext modulus is prime. We follow the Homomorphic Encryption Standard and evaluate the parameter set using the standard attacks against RLWE [31]. For the small-secret distribution used in Jaguar, the HE standard table with classical BKZ.sieve cost model gives the following estimates for $N = 2048$ and $\log_2 q = 54$:

$$\lambda_{\mathrm{uSVP}} = 129.7, \qquad \lambda_{\mathrm{dec}} = 144.4, \qquad \lambda_{\mathrm{dual}} = 134.2.$$

We take the minimum over these attacks, yielding a classical security estimate of 129.7 bits. Thus, $(N, Q) = (2048, 54)$ satisfies the classical 128-bit security target. This choice also matches the 128-bit coefficient-modulus budget commonly used by BFV implementations for $N = 2048$ [33].

Why $N = 2048$. The ring dimension N directly affects ciphertext size and the cost of coefficient-wise arithmetic. Since Jaguar's convolution kernel is built from scalar–polynomial products, coefficient shifts, additions, and masks, its linear-layer cost scales linearly with N for each ciphertext block. We therefore choose the smallest ring dimension that supports the required plaintext space and satisfies the classical 128-bit security target. Under the HE standard table, $N = 2048$ permits a 54-bit ciphertext modulus at 128-bit classical security, making it the most efficient choice for our target parameter range.

Why $P = 30$. The plaintext modulus determines the integer range used for fixed-point inference and the ring used by the subsequent 2PC protocols. We use $p = 2^{30}$ so that HE outputs are naturally represented over the same power-of-two ring as the additive shares used by the non-linear protocols. The choice $P = 30$ also provides sufficient dynamic range for the fixed-point configuration used in our ImageNet models. In Appendix E, we validate this choice by comparing fixed-point inference against the FP32 torchvision baseline.

Why $Q = 54$. Given $N = 2048$, increasing $Q = \log_2 q$ increases the available noise budget but reduces the concrete RLWE security level. We choose $Q = 54$ because it is the largest ciphertext-modulus bit-length allowed by the classical 128-bit security table for the selected ring dimension. This maximizes the noise budget while remaining within the target security level.

Compatibility with ciphertext-side truncation. For the selected parameters, the BFV scale is

$$\Delta = \frac{q}{p} = 2^{Q-P} = 2^{24}.$$

Jaguar performs ciphertext-side truncation by f bits, where $f = 9$ is the fixed-point fractional precision used in our evaluation. Since

$$2^f = 2^9 \mid \Delta = 2^{24},$$

the divisibility condition required by Section 3.1 is satisfied. After truncation, the effective scale remains $\Delta/2^f = 2^{15}$, which is sufficient for decoding target-precision fixed-point values.

The correctness condition from Section 3.1 requires the accumulated BFV noise before truncation to remain below

$$\frac{\Delta}{2} - (N + 1)2^f.$$

With $N = 2048$, $\Delta = 2^{24}$, and $f = 9$, this bound becomes

$$2^{23} - 2049 \cdot 2^9 = 7{,}339{,}520.$$

Jaguar's HE computation has multiplicative depth one per linear block before the result is converted back to additive shares. Therefore, the noise growth is limited to one plaintext–ciphertext linear evaluation, and we empirically validate correctness by checking that the decrypted, truncated HE outputs match the corresponding fixed-point reference outputs for all evaluated layers.

Classical versus post-quantum security. The parameter set above is selected for classical 128-bit security. Under the post-quantum BKZ.qsieve table in the HE standard, a smaller modulus budget is required for $N = 2048$ to claim 128-bit post-quantum security. If post-quantum 128-bit security is required, Jaguar can either reduce Q accordingly or increase the ring dimension, e.g., to $N = 4096$, while keeping the same power-of-two arithmetic design. The algorithms in Sections 3 and 4 are unchanged by this parameter adjustment, although the per-ciphertext arithmetic cost grows with N.

E Parameter Selection for Fixed-Point Inference

Table 8: Agreement with the FP32 torchvision baseline from actual OpenCheetah executions on a sampled validation subset, using the original setting and the tested (f, P) candidates.

Model	OpenCheetah fixed-point setting (f, P)				
	Default $(12, 37)$	$(9, 25)$	$(9, 27)$	$(9, 29)$	$(9, 30)$
ResNet18	97.0%	0.0%	77.0%	85.0%	93.5%
ResNet50	97.5%	0.0%	6.0%	8.0%	96.0%
MobileNetV2	87.0%	0.0%	87.5%	92.0%	94.5%

We re-selected (f, P) using a three-stage procedure after extending the OpenCheetah [20] evaluation beyond the originally provided models to additional architectures, including ResNet18 and MobileNetV2.

In the first stage, we used plaintext fixed-point simulation to screen candidate settings by measuring their agreement with the FP32 torchvision baseline under ideal arithmetic. This stage captures deterministic fixed-point effects such as quantization error and range overflow, but it is not sufficient for the final parameter choice because the actual OpenCheetah pipeline introduces approximation in the nonlinear stage.

In the second stage, we used a protocol-aware plaintext simulator to account for the numerical behavior of OpenCheetah's nonlinear primitives without executing the cryptographic protocols. On full 50,000 validation images, the protocol-aware simulation yields FP32 agreement of 85.918%/93.112% for ResNet-18 and 9.632%/96.008% for ResNet-50 at $(f, P) = (9, 29)/(9, 30)$, respectively.

In the third stage, given the high cost of end-to-end secure execution, we validated the shortlisted candidates by running OpenCheetah on a sampled subset of the validation set and comparing their outputs against the FP32 baseline.

As shown in the table above, the original OpenCheetah reference setting $(f, P) = (12, 37)$ already agrees well with the FP32 baseline on ResNet-18 and ResNet-50 (97.0% and 97.5%, respectively), but drops to 87.0% on MobileNetV2. Under $f = 9$, the setting $(f, P) = (9, 30)$ achieves FP32 agreement of 93.5%, 96.0%, and 94.5% on ResNet-18, ResNet-50, and MobileNetV2, respectively. For both ResNet-18 and ResNet-50, the sampled secure-execution agreement rates follow the same trends as the full-validation simulation in the second stage. These results provide implementation-level confirmation of the simulation-based screening.

As an additional label-level sanity check, the selected setting $(f, P) = (9, 30)$ achieves sampled top-1 accuracies of 74%, 84.5%, and 77.0% on ResNet18, ResNet50, and MobileNetV2, respectively, which are comparable to the original OpenCheetah setting (75.5%, 82.0%, and 70.0%). Based on these results, we use $(f, P) = (9, 30)$ in all subsequent experiments.

Table 9: Case-level compute-only convolution microbenchmark with AVX disabled and a single thread. Latency is averaged over 10 runs. Speedup is computed as Cheetah over Jaguar.

Conv Type	Layout	Dimension $(W, C_{\text{in}}, C_{\text{out}}, K)$	Cheetah (ms)	Jaguar (ms)	Speedup
Dense 3×3	packed	$(32, 128, 128, 3)$	2631.60	272.95	9.64×
		$(32, 16, 16, 3)$	45.21	4.82	9.38×
		$(28, 128, 128, 3)$	1975.56	266.77	7.41×
		$(28, 512, 512, 3)$	30956.48	4303.07	7.19×
		$(16, 64, 64, 3)$	180.03	69.03	2.61×
		$(16, 256, 256, 3)$	2699.46	1099.17	2.46×
		$(14, 256, 256, 3)$	2007.58	1132.27	1.77×
		$(8, 128, 128, 3)$	240.01	273.78	0.88×
		$(7, 512, 512, 3)$	2717.50	4376.67	0.62×
		$(4, 512, 512, 3)$	1241.24	4347.74	0.29×
	split	$(64, 64, 64, 3)$	7715.16	204.37	37.75×
		$(56, 64, 64, 3)$	1958.06	134.66	14.54×
		$(56, 256, 256, 3)$	30964.59	2165.62	14.30×
		$(224, 64, 64, 3)$	30962.18	2417.93	12.81×
Pointwise 1×1	packed	$(28, 128, 512, 1)$	6030.81	129.04	46.73×
		$(28, 512, 128, 1)$	5896.45	126.95	46.45×
		$(28, 32, 192, 1)$	614.23	14.55	42.20×
		$(14, 1024, 256, 1)$	6060.75	501.12	12.09×
		$(14, 256, 1024, 1)$	6088.65	522.41	11.65×
		$(14, 96, 576, 1)$	1344.09	121.90	11.03×
		$(7, 512, 2048, 1)$	6899.98	2025.97	3.41×
		$(7, 2048, 512, 1)$	6083.87	1983.25	3.07×
		$(7, 320, 1280, 1)$	2494.59	821.44	3.04×
	split	$(56, 64, 256, 1)$	937.30	72.18	12.99×
		$(56, 256, 64, 1)$	939.37	91.27	10.29×
		$(56, 24, 144, 1)$	206.50	20.41	10.12×
		$(112, 32, 16, 1)$	102.24	10.31	9.91×
Depthwise 3×3	packed	$(14, 96, 96, 3)$	50.29	3.86	13.04×
		$(28, 32, 32, 3)$	17.90	1.42	12.57×
		$(28, 192, 192, 3)$	98.95	8.09	12.23×
		$(7, 320, 320, 3)$	164.85	13.54	12.18×
		$(7, 160, 160, 3)$	87.60	8.23	10.65×
		$(14, 576, 576, 3)$	294.73	30.57	9.64×
	split	$(56, 24, 24, 3)$	14.47	2.18	6.65×
		$(56, 144, 144, 3)$	78.72	13.89	5.67×
		$(112, 32, 32, 3)$	70.89	16.52	4.29×

F Additional Experimental Results

F.1 Full Convolution Microbenchmark Results

Table 9 provides the full per-case results behind the grouped summary in Table 4. We use the same compute-only setup as in the main microbenchmark: each experiment measures only the server-side convolution computation, excluding key generation, encryption, decryption, communication, and non-linear protocols. All runs are executed with a single thread and AVX disabled. For each shape, we repeat the measurement 10 times and report the average latency. Speedup is computed from the paired average latency of Cheetah over Jaguar.

We group cases by convolution type and ciphertext layout. Since all spatial dimensions are square, each shape is written as $(W, C_{\text{in}}, C_{\text{out}}, K)$, where W is the spatial width, C_{in} and C_{out} are the input and output channel counts, and K is the kernel size. For depthwise convolutions, $C_{\text{out}} = C_{\text{in}}$.

F.2 AVX-512 Implementation

Jaguar's power-of-two modulus is well-suited to AVX-512; reduction modulo 2^Q is a vector bitwise AND and ciphertext-side truncation is a vector arithmetic right shift, so the dominant 64-bit integer kernels in SPA-CONV (scalar–polynomial multiplication, shifted accumulation, server-share addition, truncation, reduction) map directly to Intel vector instructions [40, 41].

Table 10: End-to-end latency of Jaguar with and without AVX-512 under LAN and WAN settings. Numbers in parentheses are latency reduction over AVX OFF.

Model	LAN Latency (s)		WAN Latency (s)	
	AVX ON	AVX OFF	AVX ON	AVX OFF
ResNet18	**7.90** (8.6%)	8.64	**15.10** (5.2%)	15.94
ResNet50	**25.55** (21.3%)	32.44	**44.91** (12.8%)	51.50
MobileNetV2	**16.83** (8.6%)	18.41	**31.81** (1.9%)	32.42

AVX consistently lowers end-to-end latency without changing communication (Table 10); largest gain (21.3 % on ResNet-50 LAN) appears where large convolution layers expose the most coefficient-wise work. MobileNetV2's smaller gain reflects more small layers and non-vectorized protocol overheads.

Table 11: End-to-end latency and communication with AVX enabled. Speedup and communication reduction are computed relative to Cheetah.

Model	Protocol	LAN		WAN		Communication	
		Latency (s)	Speedup	Latency (s)	Speedup	Comm. (MiB)	Reduction
ResNet18	Cheetah	22.22	1.00×	29.27	1.00×	366.94	1.00×
	Rhombus	21.96	1.01×	29.24	1.00×	344.01	1.07×
	Jaguar	**7.90**	**2.81**×	**15.10**	**1.94**×	**315.02**	**1.16**×
ResNet50	Cheetah	77.89	1.00×	103.72	1.00×	1335.64	1.00×
	Rhombus	66.27	1.18×	80.83	1.28×	**750.22**	**1.78**×
	Jaguar	**25.55**	**3.05**×	**44.91**	**2.31**×	962.0	1.39×
MobileNetV2	Cheetah	41.02	1.00×	61.20	1.00×	1042.63	1.00×
	Rhombus	46.75	0.88×	68.29	0.90×	836.81	1.25×
	Jaguar	**16.83**	**2.44**×	**31.81**	**1.92**×	**592.5**	**1.76**×

Table 11 reports the end-to-end comparison under the AVX-enabled configuration. This table complements Table 10: while Table 10 isolates the effect of AVX-512 within Jaguar, Table 11 shows the performance of the optimized Jaguar implementation against prior protocols. With AVX enabled, Jaguar achieves 2.44–3.05× lower LAN latency and 1.92–2.31× lower WAN latency than Cheetah, while also reducing communication by 1.16–1.76×. The latency benefit is largest on ResNet-50, where large convolution layers expose substantial coefficient-wise work for SPA-CONV and AVX-512 vectorization. The WAN speedups are smaller than the LAN speedups because communication and round-trip latency, which are unaffected by AVX, account for a larger fraction of end-to-end time.

G Limitation and Future Work

FC/MatMul extension. Jaguar targets the convolution path, which accounts for > 98 % of linear-layer latency in our evaluated CNNs. The current CNN implementation retains Cheetah's FC backend for the final classifier. Static-weight MatMul can also use Jaguar's arithmetic, but its efficiency depends on matrix shape, packing, and sparsity. Figure 8 shows a complementary opportunity: as weight density decreases, Jaguar-style accumulation cost drops linearly with the number of nonzero weights, while the Cheetah-style baseline remains flat, with a crossover around density 0.0295. Sparse and pruned MatMul are therefore a natural setting for extending Jaguar's regime to FC layers.

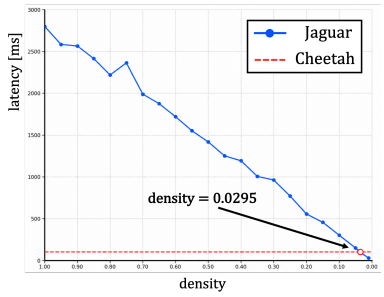


Figure 8: FC/MatMul latency trend under different weight densities.

Table 12(b) evaluates sparse BERT-base output-projection and FFN-expansion shapes using Jaguar's $(N, Q, P, f) = (2048, 54, 30, 9)$ configuration. Jaguar's timing includes sparse-metadata traversal,

Table 12: Static-weight operator microbenchmarks in attention architectures. All latencies are in milliseconds, and speedup is Cheetah/Jaguar. (a) Dense MaxViT-T projections: medians of nine runs with matched precision and HE-linear timing boundaries; n is the graph occurrence count. (b) Sparse BERT-base MatMul: Jaguar latency at each weight density, compared with a density-independent Cheetah HE-linear baseline; Jaguar reports the best of five runs and Cheetah one run per shape. Neither part measures end-to-end inference.

(a) Dense MaxViT-T projections

T	$C_{\text{in}} \to C_{\text{out}}$	Projection	n	Jaguar	Cheetah	Speedup
3136	$64 \to 64$	Output	4	11.05	71.64	$6.48\times$
3136	$64 \to 192$	QKV	4	34.80	234.16	$6.73\times$
784	$128 \to 128$	Output	4	26.88	292.73	$10.89\times$
784	$128 \to 384$	QKV	4	81.78	902.20	$11.03\times$
196	$256 \to 256$	Output	10	93.82	294.16	$3.14\times$
196	$256 \to 768$	QKV	10	285.62	929.73	$3.26\times$
49	$512 \to 512$	Output	4	379.54	320.05	$0.84\times$
49	$512 \to 1536$	QKV	4	1144.94	1004.97	$0.88\times$
Occurrence-weighted sum			44	10510.35	23541.91	$2.24\times$

(b) Sparse BERT-base static-weight MatMul

		Jaguar at weight density				
Weight shape	Sparsity	5%	3%	1%	Cheetah	Speedup at 1%
768×768	Unstructured	20.52	12.65	5.11	12.41	$2.43\times$
768×768	16×16 block	19.10	11.78	4.22	12.41	$2.94\times$
768×3072	Unstructured	87.63	54.92	23.77	51.19	$2.15\times$
768×3072	16×16 block	81.18	49.54	20.33	51.19	$2.52\times$

accumulation, and output construction but excludes metadata preprocessing; Cheetah measures its server-side HE linear loop. Across unstructured and 16×16 block sparsity, speedups are 0.93–$1.05\times$ at 3% density and 2.15–$2.94\times$ at 1%.

Mapping static projections to SPA-CONV. For token features $X \in \mathbb{Z}_{2^P}^{T \times d}$ and server-held static weights $W \in \mathbb{Z}_{2^P}^{d \times d'}$, a projection computes

$$Y_{t,o} = \sum_{c=1}^{d} X_{t,c} W_{c,o}.$$

Treating tokens as spatial positions makes this precisely a $K = 1$ convolution: the same channel-mixing weights are reused at every position. Fused QKV projection uses $W = [W_Q \; W_K \; W_V]$; the attention output projection and MLP linear layers have the same static-weight form. They therefore admit SPA-CONV's scalar–polynomial accumulation.

Share-level composition. Jaguar's projection outputs can feed an attention protocol as additive shares, provided that fixed-point scaling is compatible and a common power-of-two share ring accommodates intermediate ranges. This avoids an additional ring conversion without requiring identical ciphertext moduli. Public token permutations satisfy $\Pi(XW) = (\Pi X)W$; we verified this identity for the token permutations induced by MaxViT-T's block and grid partitioning [42], so these permutations can be applied locally to shares. The measurements below evaluate static-weight kernels; integration with protocols for the input-dependent attention core and end-to-end secure attention evaluation remain future work.

Dense attention projections. Both backends use $N = 4096$, $p = 2^{37}$, and nine fractional bits, with identical synthetic integer inputs in $[-255, 255]$ and nonzero weights in $[-31, 31]$. Jaguar uses $q = 2^{60}$; Cheetah uses a 60-bit NTT-prime data modulus and a 109-bit key-level modulus. Each retains its native packing. On an Intel Xeon Gold 6250 with AVX enabled and one worker pinned to the same core, we report medians of nine runs after two warm-ups, alternating backend order.

Timing includes HE linear evaluation, reductions, and output-buffer construction, excluding encoding, encryption/decryption, truncation, masking, share export, and communication.

Results. Table 12(a) shows speedups of 0.84–$11.03\times$, with Jaguar faster on six of eight shapes. The two 49-token shapes favor Cheetah's channel packing. Weighting median latencies by graph occurrence counts gives $\sum_s n_s t_s^{\mathrm{Cheetah}} / \sum_s n_s t_s^{\mathrm{Jaguar}} = 2.24$. All 1,505,280 logical output coefficients match the integer reference exactly modulo 2^{37} after decryption. These results measure projection kernels, not end-to-end MaxViT inference.

H Comparison with Recent Hybrid HE/2PC Systems

Recent systems such as Panther [38] and SecONNds [39] improve hybrid HE/2PC inference through optimizations to linear computation and nonlinear protocols. We compare their designs with Jaguar (Table 13) and examine how their nonlinear improvements complement Jaguar's exact HE-side truncation and reduced-bitwidth evaluation.

Table 13: Comparison of representative hybrid HE/2PC inference systems. CrypTFlow2 [11] refers to its HE-based linear backend, and SecONNds denotes the base variant rather than SecONNds-P.

System	Linear	ReLU / comparison	Truncation
CrypTFlow2	SIMD + rotations	Chosen-OT/LUT Millionaire	Faithful; bit-exact
Cheetah	Coefficient encoding; no rotations	Same Millionaire + silent OT	One-bit approximate
Panther	Shared ciphertext component; fewer polynomial multiplications	New OT Millionaire; N-way merging	Cheetah-style approximate
SecONNds	NTT preprocessing; optional GPU	Boolean GMW/FMILL; no chosen-OT/LUT comparison	Cheetah-style approximate
Jaguar	**Power-of-two SPA-CONV; no server NTT for convolution**	**Cheetah ReLU at lower bitwidth**	**Exact HE-side; no 2PC**

Technical relationship and orthogonality. Panther, SecONNds, and Jaguar all target the polynomial multiplication and NTT costs in Cheetah [12], but at different levels. SecONNds retains Cheetah's linear protocol and preprocesses weights into the NTT domain, keeping MACs in-domain until the final inverse transforms. Panther also retains an NTT-based backend, but customizes HE so that multiple ciphertexts share one polynomial component, reducing polynomial multiplications, and reuses NTT representations across repeated inner products. Jaguar instead changes the arithmetic substrate: SPA-CONV uses scalar–polynomial products, coefficient shifts, additions, and masks over a power-of-two ciphertext ring, eliminating server-side plaintext–ciphertext polynomial multiplication and NTT for convolution. Their concrete linear kernels are therefore alternatives rather than direct add-ons, although their broader packing and reuse principles remain relevant.

Panther's shared-component construction reduces the number of polynomial multiplications by up to $2\times$ in the optimal case. Jaguar achieves a measured 2.00–$5.00\times$ speedup over Cheetah for the complete Conv+FC path across the evaluated models and LAN/WAN settings.

The nonlinear contributions are orthogonal. Jaguar does not propose a new ReLU or max-pooling protocol: it exports additive shares over $\mathbb{Z}_{2^\ell}$, the interface consumed by Panther and SecONNds. Their nonlinear protocols can therefore replace Jaguar's Cheetah-based implementations. Jaguar's contribution is to move exact truncation before nonlinear evaluation, reducing the input bitwidth rather than changing the nonlinear primitive itself. This benefit is retained when the primitive is replaced; we directly verify this compatibility for SecONNds below.

CrypTFlow2 [11] faithfully corrects truncation errors through interactive 2PC, whereas Cheetah permits a possible one-LSB error. Panther and base SecONNds retain this approximate truncation while accelerating the underlying comparison and AND computations. Jaguar instead performs exact local ciphertext-side shifts before HE-to-2PC conversion, using $2^f \mid \Delta$ and the correctness conditions established in our truncation analysis. This removes post-ReLU 2PC truncation and allows the nonlinear protocol to operate on lower-bitwidth shares.

Table 14: Matched online end-to-end evaluation with AVX disabled. Speedup and communication reduction are relative to Cheetah. Bold denotes the best measured result for each model and metric. A dash denotes an unmeasured configuration.

		LAN		WAN		Communication	
Model	Protocol	Latency (s)	Speedup	Latency (s)	Speedup	MiB	Reduction
ResNet-18	Cheetah	30.30	1.00×	36.94	1.00×	366.94	1.00×
	SecONNds	14.56	2.08×	39.08	0.95×	359.62	1.02×
	Jaguar	8.64	3.51×	**15.94**	**2.32×**	315.02	1.16×
	Jaguar+SecONNds	**5.74**	**5.28×**	—	—	**308.73**	**1.19×**
ResNet-50	Cheetah	120.82	1.00×	142.72	1.00×	1335.63	1.00×
	SecONNds	67.46	1.79×	143.87	0.99×	1377.76	0.97×
	Jaguar	32.44	3.72×	**51.50**	**2.77×**	**962.00**	**1.38×**
	Jaguar+SecONNds	**24.31**	**4.97×**	—	—	998.84	1.34×
MobileNetV2	Cheetah	46.99	1.00×	67.25	1.00×	1042.62	1.00×
	SecONNds	26.13	1.80×	84.10	0.80×	1100.53	0.95×
	Jaguar	18.41	2.55×	**32.42**	**2.07×**	**592.51**	**1.76×**
	Jaguar+SecONNds	**10.80**	**4.35×**	—	—	625.42	1.67×

Matched evaluation and direct integration. We evaluate the public CPU implementation of SecONNds [43] using the same hardware, four threads, fixed-point parameters, and emulated LAN/WAN settings as Jaguar. Results use AVX disabled and measure online end-to-end inference; offline triple generation is excluded. We evaluate both serial and low-round (LR) variants and report the faster one for each model and network setting. Jaguar+SecONNds replaces only Jaguar's ReLU and max-pooling implementations with SecONNds, retaining SPA-CONV and exact HE-side truncation. Speedup and communication reduction in Table 14 are relative to Cheetah.

Under LAN, Jaguar is 1.42–2.08× faster than standalone SecONNds. The integrated system further improves Jaguar by 1.51×, 1.33×, and 1.70× on ResNet-18, ResNet-50, and MobileNetV2, respectively, confirming that SecONNds' nonlinear protocols compound with Jaguar's linear and truncation gains. SecONNds is more sensitive to network latency because its nonlinear protocol requires more communication rounds than Cheetah. Consequently, it is faster than Cheetah under our low-RTT LAN setting but slower under our WAN emulation. Since standalone SecONNds is slower than Cheetah under our WAN setting, we evaluate Jaguar+SecONNds only under LAN.

Overall, Panther and SecONNds optimize the conventional linear backend and/or nonlinear protocol, whereas Jaguar provides a different power-of-two linear backend, exact zero-communication truncation, and narrower nonlinear inputs. The direct SecONNds integration shows that recent nonlinear advances do not subsume Jaguar; they add further gains when the network environment favors them.

I Derivation of the Kernel-Level Complexity Table

Table 2 compares the dominant online server-side arithmetic of secure convolution kernels after translating different HE backends into estimated 64-bit multiplication counts. The comparison is not intended to predict full end-to-end latency. It excludes key generation, encryption, decryption, HE-to-share export, communication, additions, coefficient shifts, masks, memory traffic, and other protocol-boundary costs unless explicitly shown. For the baseline systems, we use optimistic lower-envelope estimates under their native packing assumptions, omitting ceiling factors, tiling multipliers, preprocessing costs, and lower-order terms. This convention avoids overcharging baselines for costs that are not included in Jaguar's Scalar-Poly count.

Concrete 64-bit multiplication model. For prime-modulus RNS baselines, arithmetic is performed modulo RNS prime limbs. Thus, each NTT butterfly multiplication, pointwise product, and key-switching product requires a modular reduction. We model one modular product over a 60-bit RNS

prime as

$$c_{\mathrm{red}} = 3$$

64-bit multiplications, corresponding to one raw 64×64 product and two reduction-related products in a Barrett/Montgomery-style implementation. Additions, shifts, comparisons, and conditional subtractions are not counted.

Let

$$n_N = \log_2 N.$$

For an NTT/RNS-based plaintext–ciphertext polynomial multiplication on an RLWE ciphertext with two polynomial components, we assume the plaintext polynomial can be pre-transformed whenever applicable. Hence, per ciphertext component and per RNS limb, we count one forward NTT on the ciphertext component, one pointwise product, and one inverse NTT:

$$\frac{N}{2} n_N + N + \frac{N}{2} n_N + N = N(n_N + 2)$$

modular products. Since an RLWE ciphertext has two polynomial components, one plaintext–ciphertext polynomial multiplication costs

$$\mathsf{PMult}_{64} = 2c_{\mathrm{red}} NL(n_N + 2) = N\Gamma_{\mathrm{mul}},$$

where

$$\Gamma_{\mathrm{mul}} = 6L(\log_2 N + 2).$$

For a homomorphic rotation/automorphism, we count the dominant key-switching work. The automorphism itself is an index permutation and is not counted as multiplication work. We model key switching as L^2 NTTs of decomposed digits, $2L^2$ pointwise products with the switching key, and inverse NTTs for two output components over L RNS limbs. This gives

$$\mathsf{Rot}_{64} = c_{\mathrm{red}} N \left[\left(\frac{n_N}{2} + 2 \right) L^2 + (n_N + 2)L \right] = N\Gamma_{\mathrm{rot}},$$

where

$$\Gamma_{\mathrm{rot}} = 3 \left[\left(\frac{\log_2 N}{2} + 2 \right) L^2 + (\log_2 N + 2)L \right].$$

This model is still favorable to the baselines: it does not separately charge basis extension, decomposition bookkeeping, lazy-reduction bookkeeping, additions, memory movement, or cache effects. In contrast, Jaguar uses a power-of-two ciphertext modulus, so coefficient reductions are realized by native wraparound, masking, or shifts rather than prime-modulus reduction. These non-multiplicative operations are not included in the Scalar-Poly count.

Gazelle dense convolution. Gazelle uses packed SIMD HE and ciphertext slot permutations for homomorphic convolution [9]. In Table 2, the HE-PMult column for Gazelle should be read as the cost of SIMD plaintext–ciphertext scalar multiplication translated into the same 64-bit multiplication model.

Let c_n denote the number of channels packed into one ciphertext. We use the optimistic packing assumption

$$c_n \approx \frac{N}{HW}$$

and suppress the small kernel factor R^2 as a constant for the dense convolution comparison. Under this assumption, the number of SIMD scalar multiplications is lower-bounded by

$$\frac{C_{\mathrm{in}} C_{\mathrm{out}}}{c_n} = \frac{HW C_{\mathrm{in}} C_{\mathrm{out}}}{N}.$$

Multiplying by $\mathsf{PMult}_{64} = N\Gamma_{\mathrm{mul}}$ gives

$$HW C_{\mathrm{in}} C_{\mathrm{out}} \Gamma_{\mathrm{mul}}.$$

For rotations, Gazelle has input-rotation and output-rotation variants. We use a lower-envelope estimate by taking the better native packing choice and suppressing constant kernel factors. After substituting $c_n \approx N/(HW)$, the number of rotations is estimated as

$$\frac{HW(C_{\mathrm{in}} + C_{\mathrm{out}})}{N} + C_{\mathrm{out}}.$$

Multiplying by $\mathsf{Rot}_{64} = N\Gamma_{\mathrm{rot}}$ gives

$$\big(HW(C_{\mathrm{in}} + C_{\mathrm{out}}) + C_{\mathrm{out}} N\big)\Gamma_{\mathrm{rot}}.$$

Cheetah dense convolution. Cheetah uses coefficient encoding to realize convolution through plaintext–ciphertext polynomial multiplication and avoids homomorphic rotations [12]. Under the packed full-convolution view, the number of plaintext–ciphertext polynomial multiplications is

$$\frac{HWC_{\text{in}}C_{\text{out}}}{N}.$$

Therefore, the estimated 64-bit multiplication count is

$$\frac{HWC_{\text{in}}C_{\text{out}}}{N} \cdot N\Gamma_{\text{mul}} = HWC_{\text{in}}C_{\text{out}}\Gamma_{\text{mul}}.$$

Since Cheetah's coefficient encoding removes homomorphic rotations from the convolution kernel, the HE-Rot entry is zero.

Rhombus pointwise convolution. A pointwise convolution with output spatial size $S_{\text{out}} = H'W'$ can be viewed as a matrix multiplication between

$$X' \in \mathbb{Z}^{S_{\text{out}} \times C_{\text{in}}} \quad \text{and} \quad W' \in \mathbb{Z}^{C_{\text{in}} \times C_{\text{out}}}.$$

Thus, the total matrix-multiplication volume is $S_{\text{out}}C_{\text{in}}C_{\text{out}}$. Rhombus-V2 reduces the number of plaintext–ciphertext multiplications to

$$\frac{S_{\text{out}}C_{\text{in}}C_{\text{out}}}{N}$$

and the number of rotations to

$$\sqrt{\frac{S_{\text{out}}C_{\text{in}}C_{\text{out}}}{N}}$$

using input-output packing and split-point picking [23]. Therefore, the translated PMult cost is

$$\frac{S_{\text{out}}C_{\text{in}}C_{\text{out}}}{N} \cdot N\Gamma_{\text{mul}} = S_{\text{out}}C_{\text{in}}C_{\text{out}}\Gamma_{\text{mul}},$$

and the translated rotation cost is

$$\sqrt{\frac{S_{\text{out}}C_{\text{in}}C_{\text{out}}}{N}} \cdot N\Gamma_{\text{rot}} = \sqrt{S_{\text{out}}C_{\text{in}}C_{\text{out}}N}\Gamma_{\text{rot}}.$$

Falcon depthwise convolution. Falcon optimizes depthwise convolution by packing multiple depthwise filters/channels into one weight polynomial and by applying communication-aware operator tiling [28]. Let C_w denote the number of channels packed into one weight polynomial. Ignoring ceiling factors and assuming a single spatial tile, the number of plaintext–ciphertext polynomial multiplications is

$$\frac{C_{\text{out}}}{C_w}.$$

Thus, the estimated 64-bit multiplication count is

$$\frac{C_{\text{out}}}{C_w} \cdot N\Gamma_{\text{mul}} = \frac{C_{\text{out}}}{C_w}N\Gamma_{\text{mul}}.$$

Falcon does not require homomorphic rotations in this depthwise coefficient-packing path, so the HE-Rot entry is zero. If spatial tiling is used, the above PMult term is multiplied by the number of spatial tiles. We omit this factor in Table 2 to keep the baseline estimate optimistic.

J Detailed Protocols and Algorithm

J.1 SPA-Conv: Scalar–Polynomial Accumulation Convolution

We describe Jaguar's linear convolution kernel as *Scalar–Polynomial Accumulation Convolution* (SPA-Conv). The key idea is to evaluate convolution directly in the coefficient domain by combining scalar–polynomial products with shifted accumulations in the power-of-two ciphertext ring.

Let the input tensor have shape (H, W, C_{in}) and the filter tensor have shape $(K, K, C_{\text{in}}, C_{\text{out}})$, where $K = 2r + 1$ is odd. Let

$$W_{\text{pp}} = W + p_l + p_r$$

denote the padded spatial width and

$$S = W_{\text{pp}}^2$$

the padded spatial footprint of one input channel.

SPA-Conv operates on a collection of encrypted input blocks

$$\mathcal{X} = \{\mathsf{CT}_t\}_{t=0}^{M-1},$$

where each CT_t is an RLWE ciphertext over the power-of-two ring $R_q = \mathbb{Z}_q[X]/(X^N + 1)$. Depending on the packing regime, a ciphertext may contain multiple padded input channels or a spatial strip of a single padded channel.

We define a layout map

$$\Lambda(c, j) = (\iota, \sigma),$$

which returns the source ciphertext index ι and the in-ciphertext coefficient offset σ for input channel c and output block index j. We use B to denote the number of ciphertext blocks produced per output channel. In the no-split regime, $B = 1$. In the split regime, $B = \nu$, where ν is the number of spatial strips used to represent one padded channel.

For dense and pointwise convolutions, the contributing input-channel set is

$$\mathcal{C}(o) = [C_{\text{in}}],$$

for each output channel o. For depthwise convolutions, the contributing input-channel set is

$$\mathcal{C}(o) = \{o\}.$$

We further define the kernel-offset set

$$\Omega_K = \{-r, \ldots, r\} \times \{-r, \ldots, r\},$$

and the coefficient-domain offset function

$$\delta(u, v) = uW_{\text{pp}} + v.$$

Packing layouts. A convenient abstraction is:

$$\Lambda(c, 0) = \Big(\lfloor c/\eta \rfloor,\ (c \bmod \eta)\, S\Big), \qquad \eta = \left\lfloor \frac{N}{S} \right\rfloor,$$

in the packed regime $S \leq N$, and

$$\Lambda(c, j) = (cB + j,\ 0), \qquad j \in [B],$$

in the split regime $S > N$.

For compactness, define

$$\mathcal{I}_{o,j} = \mathcal{C}(o) \times \Omega_K,$$

namely the set of all input-channel and spatial-offset pairs contributing to output channel o and block j.

Algorithm 3: SPA-Conv over a power-of-two ciphertext ring

Input: encrypted input blocks $\mathcal{X} = \{\mathsf{CT}_t\}$, filter tensor $\mathbf{W}$, layout map Λ, padded width W_{pp}, kernel size $K = 2r + 1$, truncation amount τ
Output: truncated encrypted output blocks $\{\mathsf{Y}_{o,j}\}$
for $o = 0$ to $C_{\mathrm{out}} - 1$ **do**
  for $j = 0$ to $B - 1$ **do**
    choose an anchor term $(c^\star, u^\star, v^\star)$
    $(\iota^\star, \sigma^\star) \leftarrow \Lambda(c^\star, j)$
    $\eta^\star \leftarrow \sigma^\star + \delta(u^\star, v^\star)$
    $\mathsf{Y}_{o,j} \leftarrow \mathsf{InitTerm}\big(W[u^\star, v^\star, c^\star, o], \mathsf{CT}_{\iota^\star}, s^\star\big)$
    for all $(c, u, v) \in \mathcal{I}_{o,j} \setminus \{(c^\star, u^\star, v^\star)\}$ **do**
      $(\iota, \sigma) \leftarrow \Lambda(c, j)$
      $s \leftarrow \sigma + \delta(u, v)$
      $\mathsf{Y}_{o,j} \leftarrow \mathsf{AccTerm}\big(\mathsf{Y}_{o,j}, W[u, v, c, o], \mathsf{CT}_\iota, s\big)$
    end for
    $\mathsf{Y}_{o,j} \leftarrow \mathsf{Trunc}_\tau(\mathsf{Y}_{o,j})$
    $\mathsf{Y}_{o,j} \leftarrow \mathsf{MaskReduce}(\mathsf{Y}_{o,j})$
  end for
end for
return $\{\mathsf{Y}_{o,j}\}$

Primitive operations. $\mathsf{InitTerm}(\alpha, \mathsf{CT}, s)$ initializes an accumulator with the shifted scalar term $\alpha \cdot \mathrm{Shift}_s(\mathsf{CT})$. $\mathsf{AccTerm}(\mathsf{Y}, \alpha, \mathsf{CT}, s)$ adds the shifted scalar term $\alpha \cdot \mathrm{Shift}_s(\mathsf{CT})$ into the accumulator Y. Here $\mathrm{Shift}_s(\cdot)$ denotes the negacyclic coefficient shift in R_q. Trunc_τ is Jaguar's exact ciphertext-side truncation from Section 3.1, and $\mathsf{MaskReduce}$ denotes optional local masking / reduction before the subsequent boundary conversion.

Special cases. The same algorithm specializes to several common CNN operators:

- **Pointwise** 1×1 **convolution:** set $K = 1$, so $\Omega_K = \{(0, 0)\}$.
- **Dense** 3×3 **convolution:** set $K = 3$ and $\mathcal{C}(o) = [C_{\mathrm{in}}]$.
- **Depthwise** 3×3 **convolution:** set $K = 3$ and $\mathcal{C}(o) = \{o\}$.

Thus, SPA-Conv provides a single arithmetic skeleton for pointwise, dense, and depthwise convolutions.

J.2 Client-Side Grouped Reconstruction

The grouped export protocol shares the RLWE linear component across many exported coefficients, but each coefficient still has a coefficient-specific decryption equation. This subsection makes the reconstruction rule explicit.

Let a truncated source RLWE ciphertext be

$$Y_\gamma = (\hat{b}_\gamma(X), \hat{a}_\gamma(X)) \in R_{2^Q}^2,$$

and let the secret key be $\hat{s}(X)$. The decrypted polynomial before fixed-point decoding is

$$\hat{u}_\gamma(X) = \hat{b}_\gamma(X) + \hat{a}_\gamma(X)\hat{s}(X) \pmod{X^N + 1, 2^Q}.$$

For a logical output value (o, ξ), the extraction map gives

$$\Phi(o, \xi) = (\gamma, \kappa),$$

so the client must reconstruct the coefficient

$$\hat{u}_{\gamma,\kappa} = \hat{b}_\gamma[\kappa] + (\hat{a}_\gamma \hat{s})[\kappa] \pmod{2^Q}.$$

Therefore, the grouped export record carries the coefficient indices $\boldsymbol{\kappa}_\gamma$, and the client reconstructs the polynomial $\hat{a}_\gamma(X)\hat{s}(X)$ once per source ciphertext group. The resulting coefficient $(\hat{a}_\gamma \hat{s})[\kappa_i]$ is then reused with the corresponding exported constant term $\mathbf{b}_\gamma[i]$.

Algorithm 4: Client-side grouped reconstruction

Require: Grouped exports $\mathcal{G} = \{G_\gamma\}$, secret key $\hat{s}(X)$, parameters (Q, P, f)
Ensure: Client additive share $\langle \mathbf{T}' \rangle_C$

Initialize $\langle \mathbf{T}' \rangle_C$ with zeros
Precompute an NTT representation of $\hat{s}(X)$ over an auxiliary NTT prime q_{ntt}
for all $G_\gamma = (\hat{a}_\gamma, \mathbf{b}_\gamma, \boldsymbol{\kappa}_\gamma, \mathbf{ch}_\gamma, \mathbf{off}_\gamma, \beta_\gamma) \in \mathcal{G}$ **do**
    Compute

$$\hat{d}_\gamma(X) \leftarrow \hat{a}_\gamma(X)\hat{s}(X) \pmod{X^N + 1}$$

    using the auxiliary NTT prime and map the result back to residues modulo 2^Q
    for $i = 0$ to $|\mathbf{b}_\gamma| - 1$ **do**
        $\kappa \leftarrow \boldsymbol{\kappa}_\gamma[i]$
        $u \leftarrow \hat{d}_\gamma[\kappa] + \mathbf{b}_\gamma[i] \pmod{2^Q}$

$$z \leftarrow \left((u + 2^{Q-P-1-f}) \bmod 2^Q\right) \gg (Q - P) \pmod{2^{P-f}}$$

        Place z into channel $\mathbf{ch}_\gamma[i]$ and output position $\mathbf{off}_\gamma[i]$
    end for
end for
return $\langle \mathbf{T}' \rangle_C$

In our implementation, this multiplication is performed locally by the client using an auxiliary NTT prime q_{ntt}. This auxiliary NTT is only an implementation technique for client-side reconstruction; it does not change the Jaguar ciphertext modulus, the protocol messages, or the communication pattern. The client first maps the coefficients of a_γ and the ternary secret key to residues modulo q_{ntt}, computes

$$\hat{d}_\gamma(X) = \hat{a}_\gamma(X)\hat{s}(X) \pmod{X^N + 1}$$

via NTT multiplication, and maps the selected coefficients back to residues modulo 2^Q. For each exported coefficient i, it then computes

$$u_i = \hat{d}_\gamma[\boldsymbol{\kappa}_\gamma[i]] + \mathbf{b}_\gamma[i] \pmod{2^Q}.$$

Finally, because the ciphertext has already undergone f-bit HE-side truncation, the client applies the same two-step fixed-point decoding rule as Algorithm 2:

$$z_i = \left((u_i + 2^{Q-P-1-f}) \bmod 2^Q\right) \gg (Q - P) \pmod{2^{P-f}}.$$

The decoded value z_i is written to the output location specified by $(\mathbf{ch}_\gamma[i], \mathbf{off}_\gamma[i])$. Thus, the grouped export shares the RLWE linear component in communication, while reconstruction remains coefficient-correct through the explicit coefficient index $\boldsymbol{\kappa}_\gamma[i]$.

Auxiliary NTT modulus. The auxiliary prime must be large enough so that the integer coefficients of $\hat{a}_\gamma(X)\hat{s}(X)$ do not wrap modulo q_{ntt}. Let $h = \|\hat{s}\|_0$ be the number of nonzero secret-key coefficients. After f-bit ciphertext-side truncation, the exported linear component has effective magnitude below 2^{Q-f}. A conservative sufficient condition is

$$q_{\mathrm{ntt}}/2 > h \cdot 2^{Q-f}.$$

For our main setting $N = 2048$, $Q = 54$, and $f = 9$, we have $h \le N = 2^{11}$, so the product bound is at most 2^{56}. A 60-bit NTT-friendly prime has centered range about 2^{59}, which is sufficient for the grouped reconstruction path. We choose a 60-bit prime $q_{\mathrm{ntt}} \equiv 1 \pmod{2N}$ and lift coefficients to centered representatives before the auxiliary-prime NTT product. If the export is performed without HE-side truncation ($f = 0$), this single-prime bound is no longer sufficient; in that case one can either use a multi-prime CRT NTT or a direct power-of-two negacyclic multiplication. Jaguar's evaluated protocol performs grouped export after HE-side truncation.

K Complexity Analysis of Linear-Layer Protocol of Jaguar

K.1 Complexity Analysis of SPA-Conv

We analyze SPA-Conv at the level of coefficient-wise integer operations. Our goal is to expose the arithmetic structure of Jaguar's linear convolution kernel independently of any specific code path.

Cost model. An RLWE ciphertext in Jaguar consists of two degree-N polynomials. We count five types of low-level integer operations: *multiplications*, *additions/subtractions*, *arithmetic shifts*, *bitwise masks*, and *plaintext-mask additions*. We ignore memory writes and index arithmetic, since they do not affect the leading arithmetic cost.

For one ciphertext block, the SPA-Conv primitives have the following costs:

- $\mathsf{InitTerm}$ touches both RLWE components over all N coefficients:
$$\mathrm{Mul}(\mathsf{InitTerm}) = 2N.$$
- $\mathsf{AccTerm}$ performs one shifted scalar accumulation on both RLWE components. It requires one coefficient-wise scalar multiplication and one accumulator update per coefficient, together with sign-adjustments induced by negacyclic wrap-around. We conservatively upper-bound its arithmetic cost by
$$\mathrm{Mul}(\mathsf{AccTerm}) = 2N, \qquad \mathrm{Add/Sub}(\mathsf{AccTerm}) \le 4N.$$
- Trunc_τ performs one arithmetic right shift on each coefficient of both ciphertext components:
$$\mathrm{Shift}(\mathsf{Trunc}_\tau) = 2N.$$
- $\mathsf{MaskReduce}$ applies power-of-two reduction to both ciphertext components:
$$\mathrm{Mask}(\mathsf{MaskReduce}) = 2N.$$
- If a plaintext mask is added before export, only the first ciphertext component is updated:
$$\mathrm{PlainAdd} = N.$$

Number of convolution terms. For one output channel o and one output block b, define
$$T_o = |\mathcal{C}(o)|\, K^2.$$
This is the total number of shifted scalar terms accumulated into that output block. SPA-Conv uses one anchor term to initialize the accumulator and then accumulates the remaining $T_o - 1$ terms.

Therefore, the arithmetic cost for one output block is
$$\mathrm{Mul}_{o,j} = 2N + 2N(T_o - 1) = 2N\,T_o,$$
$$\mathrm{Add/Sub}_{o,j} \le 4N(T_o - 1),$$
and the local post-processing cost is
$$\mathrm{Shift}_{o,j} = 2N, \qquad \mathrm{Mask}_{o,j} = 2N, \qquad \mathrm{PlainAdd}_{o,j} = N.$$

General layer complexity. Let B denote the number of ciphertext blocks produced per output channel. Then the total arithmetic cost of one SPA-Conv layer is
$$\mathrm{Mul}_{\mathrm{SPA-Conv}} = 2NB \sum_{o=0}^{C_{\mathrm{out}}-1} T_o = 2NBK^2 \sum_{o=0}^{C_{\mathrm{out}}-1} |\mathcal{C}(o)|,$$
$$\mathrm{Add/Sub}_{\mathrm{SPA-Conv}} \le 4NB \sum_{o=0}^{C_{\mathrm{out}}-1} (T_o - 1) = 4NB \left(K^2 \sum_{o=0}^{C_{\mathrm{out}}-1} |\mathcal{C}(o)| - C_{\mathrm{out}} \right),$$
$$\mathrm{Shift}_{\mathrm{SPA-Conv}} = 2NBC_{\mathrm{out}}, \qquad \mathrm{Mask}_{\mathrm{SPA-Conv}} = 2NBC_{\mathrm{out}}, \qquad \mathrm{PlainAdd}_{\mathrm{SPA-Conv}} = NBC_{\mathrm{out}}.$$

These expressions separate the convolution kernel from the boundary conversion. In particular, the grouped output-export procedure used by Jaguar is linear in the number of exported coefficients and depends on the chosen export format; we therefore treat it separately from the arithmetic kernel analyzed here.

Dense and depthwise specializations. For dense and pointwise convolutions,

$$|\mathcal{C}(o)| = C_{\text{in}} \qquad \forall o,$$

and therefore

$$\text{Mul}_{\text{dense}} = 2NBC_{\text{out}}C_{\text{in}}K^2,$$
$$\text{Add/Sub}_{\text{dense}} \leq 4NBC_{\text{out}}(C_{\text{in}}K^2 - 1),$$
$$\text{Shift}_{\text{dense}} = 2NBC_{\text{out}}, \quad \text{Mask}_{\text{dense}} = 2NBC_{\text{out}}, \quad \text{PlainAdd}_{\text{dense}} = NBC_{\text{out}}.$$

For depthwise convolutions,

$$|\mathcal{C}(o)| = 1 \qquad \forall o,$$

and therefore

$$\text{Mul}_{\text{dw}} = 2NBC_{\text{out}}K^2,$$
$$\text{Add/Sub}_{\text{dw}} \leq 4NBC_{\text{out}}(K^2 - 1),$$
$$\text{Shift}_{\text{dw}} = 2NBC_{\text{out}}, \quad \text{Mask}_{\text{dw}} = 2NBC_{\text{out}}, \quad \text{PlainAdd}_{\text{dw}} = NBC_{\text{out}}.$$

Common special cases. The formulas above directly yield the following cases of practical interest.

Pointwise 1×1 convolution. Setting $K = 1$ in the dense formula gives

$$\text{Mul}_{1\times1} = 2NBC_{\text{out}}C_{\text{in}},$$
$$\text{Add/Sub}_{1\times1} \leq 4NBC_{\text{out}}(C_{\text{in}} - 1),$$
$$\text{Shift}_{1\times1} = 2NBC_{\text{out}}, \quad \text{Mask}_{1\times1} = 2NBC_{\text{out}}, \quad \text{PlainAdd}_{1\times1} = NBC_{\text{out}}.$$

Dense 3×3 convolution. Setting $K = 3$ in the dense formula gives

$$\text{Mul}_{3\times3\text{-dense}} = 18NBC_{\text{out}}C_{\text{in}},$$
$$\text{Add/Sub}_{3\times3\text{-dense}} \leq 4NBC_{\text{out}}(9C_{\text{in}} - 1),$$
$$\text{Shift}_{3\times3\text{-dense}} = 2NBC_{\text{out}}, \quad \text{Mask}_{3\times3\text{-dense}} = 2NBC_{\text{out}}, \quad \text{PlainAdd}_{3\times3\text{-dense}} = NBC_{\text{out}}.$$

Depthwise 3×3 convolution. Setting $K = 3$ in the depthwise formula gives

$$\text{Mul}_{3\times3\text{-dw}} = 18NBC_{\text{out}},$$
$$\text{Add/Sub}_{3\times3\text{-dw}} \leq 32NBC_{\text{out}},$$
$$\text{Shift}_{3\times3\text{-dw}} = 2NBC_{\text{out}}, \quad \text{Mask}_{3\times3\text{-dw}} = 2NBC_{\text{out}}, \quad \text{PlainAdd}_{3\times3\text{-dw}} = NBC_{\text{out}}.$$

Interpretation. These expressions make the arithmetic structure of Jaguar explicit. Dense $K \times K$ convolution scales as $\Theta(NBC_{\text{out}}C_{\text{in}}K^2)$, pointwise convolution scales as $\Theta(NBC_{\text{out}}C_{\text{in}})$, and depthwise convolution scales as $\Theta(NBC_{\text{out}}K^2)$. Hence, relative to dense 3×3 convolution, pointwise convolution removes the spatial K^2 factor, whereas depthwise convolution removes the multiplicative C_{in} factor. The packing layout only changes the block multiplier B: in the no-split regime, $B = 1$, while in the split regime, $B = \nu$.

K.2 Complexity Analysis of Grouped Output Export

Grouped Output Export. After SPA-Conv produces encrypted output blocks $\{\mathsf{Y}_{o,j}\}$, Jaguar converts them into additive shares by exporting only the coefficients that correspond to valid convolution outputs. For each source RLWE ciphertext, Jaguar extracts the valid output coefficients as coefficient-wise LWE ciphertexts of the form $(b_\kappa, \hat{a})$, where all extracted coefficients from the same source ciphertext share the same linear component $\hat{a}$. Instead of transmitting these LWEs independently, Jaguar packs them into a grouped export record consisting of one shared $\hat{a}$, multiple b values, and output-location metadata.

Let

$$\Phi : (o, \xi) \mapsto (\gamma, \kappa)$$

denote the extraction map induced by the packing layout, where o is the output channel index, ξ is the logical output position, γ is the source RLWE ciphertext index, and κ is the coefficient index inside that ciphertext.

For each source ciphertext $\mathsf{Y}_\gamma = (b_\gamma, \hat{a}_\gamma)$, define the extracted set

$$\mathcal{P}_\gamma = \{(o, \xi, \kappa) : \Phi(o, \xi) = (\gamma, \kappa)\}.$$

Jaguar forms one grouped export per non-empty set $\mathcal{P}_\gamma$.

Algorithm 5: Grouped output export

Require: Truncated encrypted output blocks $\{Y_\gamma\}$ and extraction map Φ
Ensure: Grouped exports $\mathcal{G} = \{G_\gamma\}$

```
Initialize 𝒢 ← ∅
for all source ciphertext indices γ such that P_γ ≠ ∅ do
    Let Y_γ = (b_γ, â_γ)
    Initialize empty vectors b_γ, κ_γ, ch_γ, off_γ
    for all (o, ξ, κ) ∈ P_γ do
        Append b_γ[κ] to b_γ
        Append κ to κ_γ
        Append o to ch_γ
        Append ξ to off_γ
    end for
    Choose transport bit-width β_γ
    G_γ ← (â_γ, b_γ, κ_γ, ch_γ, off_γ, β_γ)
    Append G_γ to 𝒢
end for
return 𝒢
```

Complexity of Grouped Output Export. We now analyze the boundary-conversion cost of grouped output export. This cost is separate from the SPA-Conv arithmetic kernel and depends on how many logical output coefficients are exported from each source ciphertext.

Notation. Let G denote the number of non-empty export groups. For group $g \in [G]$, let

$$m_g$$

be the number of exported coefficients in that group, and let

$$\beta_g$$

be the transport bit-width used to pack its values. Let

$$E = \sum_{g=1}^{G} m_g$$

denote the total number of exported output coefficients. For convolution, E equals the total number of logical output values, i.e.,

$$E = C_{\text{out}} H' W'.$$

Server-side grouping cost. For each group, the server copies one shared RLWE linear component of length N and extracts m_g constant coefficients. Hence, the arithmetic work of forming all groups is

$$\Theta(GN + E)$$

coefficient reads/writes. If bit-packing is included, the bit-level packing work is

$$\Theta\left(\sum_{g=1}^{G} \beta_g (N + m_g)\right).$$

Communication cost. Group g contains: (i) one shared RLWE linear component of length N, (ii) m_g exported constant terms, (iii) m_g coefficient indices, and (iv) metadata describing the logical output locations of those m_g values. Thus, the transmitted payload of group g is

$$\beta_g(N + m_g) + m_g \left(\lceil \log_2 N \rceil + \lceil \log_2 C_{\text{out}} \rceil + \lceil \log_2(H'W') \rceil\right)$$

bits, up to lower-order header terms. Summing over all groups,

$$\text{Comm}_{\text{grouped}} = \sum_{g=1}^{G} \left[\beta_g(N + m_g) + m_g \left(\lceil \log_2 N \rceil + \lceil \log_2 C_{\text{out}} \rceil + \lceil \log_2(H'W') \rceil\right)\right] + O(G).$$

Client-side reconstruction cost. For each group, the client reconstructs the negacyclic product $a_\gamma(X)s(X)$ once and then selects the m_g requested coefficients. With the auxiliary NTT implementation, the cost is

$$\Theta(GN \log N + E)$$

arithmetic operations, after one global precomputation of the NTT representation of the secret key. The M term accounts for coefficient selection, addition of the exported constant terms, and power-of-two decode/round steps. This client-side cost is still separate from the server-side SPA-CONV kernel and does not affect communication.

A direct power-of-two implementation could instead compute the same product by negacyclic add/sub accumulation using the ternary secret key; the protocol is unchanged, but the implementation cost becomes $O(GNh + E)$, where $h = \|\hat{s}\|_0$.

Comparison with per-output export. Without grouping, exporting each logical output independently would require one length-N linear component per output value, leading to communication proportional to EN. Grouped export reduces this to $GN + E$, where typically $G \ll E$ because many output coefficients originate from the same source RLWE ciphertext. This is the main reason Jaguar's HE-to-share boundary remains efficient even when the logical convolution output is dense.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction match the proposed techniques and experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4.2 discusses Jaguar's current FC/MatMul limitation, and Appendix G provides further discussion of native power-of-two FC/MatMul extensions.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The main correctness theorem states its assumptions and is proved in Appendix B. Security/parameter assumptions and the protocol/cost analyses are also provided in Appendices.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper reports the evaluated models, parameters, baselines, hardware, network settings. Source code and reproduction scripts are publicly available at `https://github.com/yellowon/Jaguar`.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Source code and reproduction scripts are publicly available at `https://github.com/yellowon/Jaguar`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5.1 specifies the HE parameters, models, dataset, implementation, hardware, network settings, and baselines.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report confidence intervals or error bars because the main evaluation requires expensive end-to-end private inference runs; the code is provided for reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The hardware configuration and execution settings are described in Section 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work follows the NeurIPS Code of Ethics and uses existing public datasets, pretrained models, and public baseline implementations.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper focuses on the technical efficiency of private CNN inference and does not include a separate discussion of broader societal impacts. The main expected benefit is improved privacy-preserving inference, while deployment-specific risks should be assessed in downstream applications.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models, scraped datasets, or generative systems requiring additional safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing datasets, pretrained models, libraries, and baseline implementations are cited and documented in the paper or supplementary material.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Source code and reproduction scripts are publicly available at `https://github.com/yellowon/Jaguar`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human-subject research or collection of personal data.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as an original or non-standard component of the method, implementation, or experiments.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.