FHE-AGENT: AUTOMATING CKKS CONFIGURATION FOR PRACTICAL ENCRYPTED INFERENCE VIA AN LLM-GUIDED AGENTIC FRAMEWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Fully homomorphic encryption (FHE), particularly CKKS, enables privacy-preserving machine learning services, but configuring encrypted inference requires balancing security, numerical quality, and latency. A deployment must jointly choose cryptographic parameters and packing settings whose effects extend across the network and its compiler-derived bootstrap schedule. We present FHE-Agent, a large language model (LLM)-guided agentic framework that automates CKKS configuration search through coordinated multi-agent LLM roles and a deterministic FHE tool suite. Proposal agents develop initial configurations and global or layer-local revisions; selection and assessment agents interpret tool evidence and measured outcomes through shared history. A three-phase workflow progresses from initial proposals to a verified starting configuration and then feedback-guided refinement. Implemented on Orion and Lattigo, it uses static and offline analyses to guide candidate selection, while encrypted execution verifies outcomes and deterministic checks preserve the best result. Recorded searches produce verified configurations on LeNet, AlexNet, and ResNet20, including a 604.79 s ResNet20 configuration. Operation-level profiles attribute most first-to-best time reductions to bootstrapping in AlexNet and ResNet20 and to linear transforms in LeNet; paired changes show that removing a bootstrap auxiliary prime slowed inference by 14–18%, while fixed-global BSGS overrides reduced the targeted layers' time. In same-LLM LeNet pilots with three LLMs, compared with one-shot and repeated independent generation, direct measured feedback, and a general-purpose agent, FHE-Agent is the only method whose attempts all pass under every LLM. These results concern configuration quality in recorded runs, not repeated-deployment speedups or search cost.

1 INTRODUCTION

Machine learning as a service (MLaaS) (Hesamifard et al., 2018) enables deep neural network deployment on third-party infrastructure, but sending raw inputs risks exposing sensitive attributes. Fully homomorphic encryption (FHE) (Hesamifard et al., 2018; Cheon et al., 2017; Frimpong et al., 2024) allows a server to compute on encrypted data and return encrypted predictions without accessing plaintext. Approximate schemes such as CKKS (Cheon et al., 2017) support real-valued arithmetic and single-instruction, multiple-data (SIMD) packing for encrypted inference.

Encrypted inference nevertheless incurs substantial overhead (Ran et al., 2023; Krastev et al., 2024; Brynds et al., 2026) relative to cleartext evaluation. CPU implementations of ResNet-20 on CIFAR-10 (Lee et al., 2022) require careful cost management, while GPU accelerators such as Cheddar (Choi et al., 2026) reduce execution cost. Compilers including CHET (Dathathri et al., 2019), Orion (Ebel et al., 2025), and Fhelipe (Krastev et al., 2024) already automate substantial parts of parameter selection, layout, and scheduling. Orion, for example, optimizes packing, scale management, and bootstrap placement. Poor choices of modulus chain, packing layout, or bootstrap parameters can still increase runtime or cause numerical failure.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Deploying a new model can still require repeated evaluation (Krastev et al., 2024; Cabrero Holgueras, 2023), diagnosis, and revision. Practitioners jointly choose CKKS parameters, encoding scale, packing and exposed bootstrap-related settings, together with supported layer-specific layouts under security and numerical-quality constraints. These choices interact across layers in model-specific ways: packing trades ciphertext count against rotation work; larger scales can preserve precision but require sufficient modulus capacity; and insufficient depth requires another configuration or bootstrapping. Establishing their combined effect on latency and numerical quality requires costly encrypted execution, so the search must decide which candidates warrant that expense. Backends (Ebel et al., 2025; Brynds et al., 2026; Ran et al., 2023) already expose parameter-security checks, structural analyses, and profiling signals. The remaining search decisions connect this evidence across trials: which coupled settings to revise, whether to pursue a global or local change, and which candidate warrants encrypted validation. Runtime estimates, numerical diagnostics, and failures provide different and sometimes incomplete evidence for these decisions.

Large language model (LLM) agents offer a way to relate configuration changes to these different forms of evidence and revise proposals using trial history. This motivates explicit proposal, selection, and assessment responsibilities: proposal generates alternatives, selection compares their analyzed consequences, and assessment interprets measured outcomes for the next round. AutoGen (Wu et al., 2024) and Reflexion (Shinn et al., 2023) support planning and iterative refinement. In FHE, FHE-Coder (Kumar et al., 2026) covers secure program generation for torus-based TFHE and approximate-arithmetic CKKS; FHECrafter (Yu et al., 2026) combines tensor-program generation with parameter tuning; and AlphaEvolve (Gorantala et al., 2026) optimizes FHE kernels using correctness and latency feedback. We study configuration search for fixed Orion networks, connecting global settings, compiler-derived bootstrap schedules, and supported local refinements.

In this paper, we introduce **FHE-Agent**, a multi-agent framework for CKKS configuration search, implemented on Orion and Lattigo. Initialization agents propose and select a starting configuration; global and layerwise proposal roles develop revisions; refinement selection and assessment roles connect analyzed candidates to measured feedback. The roles share an underlying LLM and structured history, exchanging complete configurations, candidate evidence, and execution outcomes. Their task is to decide what to test next using the available evidence. Orion derives schedules, while deterministic checks and encrypted measurements govern feasibility and the verified best.

Concretely, we make the following contributions:

(1) A coordinated multi-agent search framework. FHE-AGENT assigns initial proposal and selection, global and layerwise revision, and refinement assessment to explicit agent roles. Candidate configurations, tool analyses, and measured feedback connect these roles through shared state in a three-phase workflow under an encrypted-evaluation budget.

(2) Evidence interfaces for agent decisions. Complete configurations connect global CKKS settings and layer-local overrides to static checks, numerical diagnostics, compiler-derived bootstrap effects, and runtime estimates. These interfaces expose candidate differences, missing evidence, and measured outcomes to the agents; deterministic checks govern admission and verified-best updates.

(3) Search- and decision-level evaluation. Recorded LeNet, AlexNet, and ResNet20 searches characterize configuration quality and best preservation, and operation-level profiles locate where measured time changes. A same-LLM LeNet comparison evaluates FHE-Agent, one-shot and repeated independent generation, direct measured feedback, and a general-purpose agent as complete systems. Decision traces and paired global/local changes relate agent choices to measured outcomes; component studies assess the numerical and runtime evidence available to those choices.

2 BACKGROUND AND RELATED WORK

2.1 FHE ENCRYPTED INFERENCE USING CKKS

CryptoNets (Gilad-Bachrach et al., 2016) demonstrated inference on homomorphically encrypted inputs. We use plaintext server weights and CKKS (Cheon et al., 2017), which supports approximate arithmetic on values packed into ciphertext slots. The ring dimension N, ciphertext modulus Q, and encoding scale Δ (reported as $\log_2 \Delta$) constrain capacity, precision, and computation. A larger scale can preserve more precision but requires more modulus capacity. Increasing Q can

change the minimum N for a security target. In residue number system form, Q comprises a chain of primes; the ciphertext level identifies the active prefix. Key switching uses an auxiliary prime chain P, and bootstrapping can use its own auxiliary chain. Multiplication changes the scale, while rescaling divides by a modulus factor and consumes a level. Multiplicative depth counts sequential dependencies, not total multiplications. Independent operations can share a depth, and different operand types need not follow the same rescaling path. Bootstrapping (Bossuat et al., 2021; Chen et al., 2019) refreshes ciphertexts at a runtime and approximation cost; its placement (Ebel et al., 2025) and restored levels affect latency. A longer chain can reduce bootstrap frequency but increases per-operation costs. Linear and convolution layers combine rotations, plaintext products, and accumulation. Packing maps tensor values to ciphertext slots and determines ciphertext and rotation counts; Orion (Ebel et al., 2025) uses diagonal encoding and baby-step/giant-step (BSGS) evaluation to reuse rotations across groups. Non-polynomial activations use polynomial approximations whose intervals and degrees affect error and level consumption. Bootstrap count alone cannot predict runtime. Appendix J covers other packing techniques, protocols, libraries, and optimizers.

2.2 FHE Libraries and Compilers

SEAL (Chen et al., 2017), OpenFHE (Al Badawi et al., 2022), and Lattigo (Lattigo) implement homomorphic primitives, whose costs vary with parameters, implementation, and hardware; compilers coordinate their use. CHET (Dathathri et al., 2019) automates parameter selection and tensor layouts; Fhelipe (Krastev et al., 2024) automates tensor packing; Rotom (Chen et al., 2026) explores layout assignments. Orion (Ebel et al., 2025) combines packing, scale management, and bootstrap placement. Dedicated optimizers address numerical management and scheduling. ELASM (Lee et al., 2023) searches scale-management plans using output-error and latency estimates. DaCapo (Cheon et al., 2024) places bootstraps using ciphertext liveness and latency estimates. ReSBM (Liu et al., 2025) uses region decomposition and min-cut optimization for scale and bootstrap management; Orbit (Zhou et al., 2026) casts their joint placement as an integer linear program. Orion's shortest-path optimization (Ebel et al., 2025) over level choices, including residual subgraphs, jointly optimizes linear-operation and bootstrap costs. FHE-Agent retains this scheduler; its outer loop varies exposed settings and compares resulting execution plans.

Parameter search also spans adaptation scopes. LOHEN (Nam et al., 2025) optimizes layerwise ciphertext configurations and switching costs. AutoFHE (Ao & Boddeti, 2024) searches mixed-degree polynomial activations and bootstrapping architectures while fine-tuning weights. We keep learned weights fixed and select supported global parameters with layer-local embedding methods and BSGS ratios; these overrides change packing rather than select per-layer cryptographic contexts. For cost guidance, CryptOracle (Brynds et al., 2026) profiles OpenFHE primitives and hardware to estimate runtime and energy. FHE-Agent combines backend analyses with compatible prior measurements, then tests selected configurations through encrypted execution. Cost estimates guide selection; parameter checks and measured encrypted results determine feasibility.

2.3 LLM-Based Agentic Workflows

LLM agents can combine tool calls with feedback and history. AutoGen (Wu et al., 2024) provides agent–tool conversations, Reflexion (Shinn et al., 2023) retains textual feedback, and Voyager (Wang et al., 2024) combines execution feedback with reusable skills. These precedents motivate explicit tool interfaces and trial history. Agentic FHE systems already address program generation and parameter tuning. FHE-Coder (Kumar et al., 2026) covers TFHE and CKKS through prompt formalization, retrieval, and iterative functional and security verification. FHECrafter (Yu et al., 2026) generates encrypted tensor programs and tunes CKKS depth, modulus, and ring dimension through execution checks and timing; its evaluated operators use leveled CKKS without bootstrapping. AlphaEvolve (Gorantala et al., 2026) evolves TFHE and CKKS kernels on tensor processing units using correctness tests and hardware latency. FHE-Agent coordinates proposal, selection, and assessment agents for end-to-end configuration search on fixed networks, covering global parameters, induced bootstrap schedules, and layer-local packing. The agents exchange backend evidence and measured outcomes, retaining the existing compiler and its schedulers. Evaluation must account for analysis, compilation, and encrypted trials alongside selected-configuration inference latency.

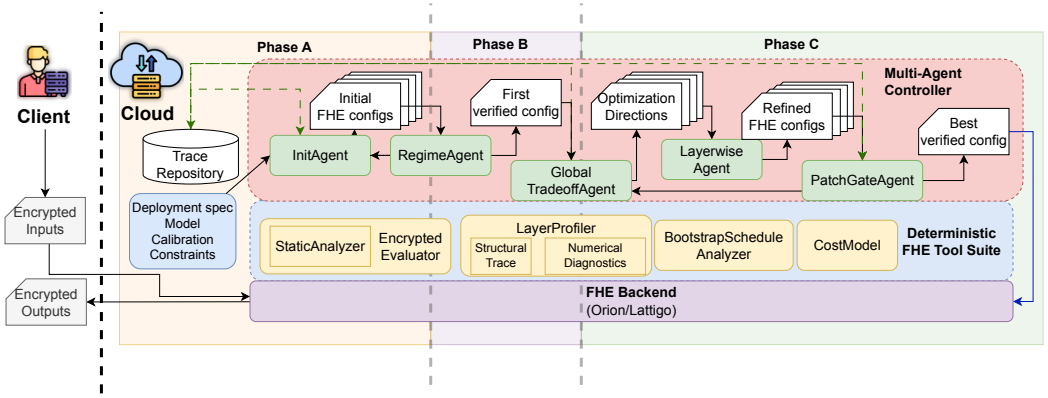

Figure 1: FHE-AGENT architecture. Agent roles organize initial proposals (A), first verified feasibility (B), and global/local refinement (C). Roles share an LLM; some proposal responsibilities share a call. All phases share the tools and backend; Orion derives bootstrap schedules. Selected candidates undergo encrypted validation, and all outcomes enter history. Parameter checks and encrypted measurements govern feasibility; only feasible measured improvements update the best. Offline calibration is separate from private client inference. Arrows show information flow.

3 FHE-AGENT: AGENTIC CKKS CONFIGURATION SEARCH

FHE-AGENT combines cooperating LLM agent roles, a deterministic FHE tool suite, and the Orion/Lattigo evaluation backend (Figure 1). The agents propose, compare, and revise configurations through shared candidate evidence and execution history. Their decisions allocate encrypted evaluations across global and layer-local alternatives; tools and measured outcomes determine whether a result can replace the verified best.

3.1 CONFIGURATION SEARCH PROBLEM

We consider offline configuration search for CKKS inference with a plaintext server model and encrypted client inputs under a semi-honest server. Numerical diagnostics use public or authorized calibration inputs; deployed client plaintext is not exposed to the controller. Model confidentiality, malicious-server verification, and side channels are out of scope (Appendix H).

For fixed network weights, architecture, and evaluation inputs, the controller searches a configuration C (Eq. (1)) of global CKKS parameters and packing options with layer-local embedding methods and BSGS ratios for the layers I with supported overrides (Table 15). Orion recomputes levels, operation counts, and bootstrap schedules for each effective configuration; these are derived facts, not controller-specified schedules. A configuration is feasible only if parameter checks pass the security target of at least 128 bits, encrypted execution completes, and measured mean absolute error (MAE) and precision satisfy the protocol's thresholds; predictions do not establish feasibility. With $\mathcal{F}$ the configurations verified during search that satisfy these conditions, the objective is

$$C = \left(C_{\text{global}}, \{C^{(i)}_{\text{local}}\}_{i \in I}\right), \qquad (1) \qquad\qquad C^* \in \arg\min_{C \in \mathcal{F}} T_{\text{FHE}}(C), \qquad (2)$$

where T_{FHE} is measured encrypted forward time under a fixed protocol. C^* is the best configuration found, without a global-optimality claim; if $\mathcal{F}$ is empty, the search returns no feasible result.

3.2 FHE TOOL SUITE AND BACKEND CAPABILITIES

The tools expose different kinds of evidence: structural plans describe the computation a configuration induces and offline numerical diagnostics identify risks, but only parameter checks and encrypted measurements establish feasibility and measured runtime. The controller weighs pre-execution evidence and compatible prior measurements before committing encrypted validation; a favorable estimate is a reason to test a proposal, not to accept it as the new best.

StaticAnalyzer. Interface and parameter checks identify unsupported fields and inconsistent settings. Candidate analysis obtains the effective residual and bootstrap contexts, so security checks use the relevant moduli and secret distribution. Rejections return violations for correction.

LayerProfiler. Offline analysis prepares packing, calibrates on authorized inputs, and constructs a symbolic execution plan. The profile records layer shapes, levels and scales, rotations, ciphertext counts, and bootstrap consequences, alongside activation ranges and approximation diagnostics. These observations connect a proposed local change to network-wide effects. Its diagnostics identify risks without measuring final ciphertext error; missing analysis is reported.

BootstrapScheduleAnalyzer. Orion derives bootstrap placement from the candidate parameters, packing, level budget, and network graph. The tool inspects this backend-derived schedule and exposes sites, slots, ciphertext counts, and surrounding levels. A longer chain may reduce bootstrap frequency while increasing arithmetic cost; auxiliary-prime changes can affect runtime even when placement is unchanged. The controller compares these consequences; Orion retains scheduling.

CostModel. A runtime proxy combines multiplication-related, rotation, and bootstrap features, calibrated to seconds from compatible prior profiles when available, otherwise relative. Optional operation-level evidence reuses compatible earlier timings and reports coverage; unmatched work is not free, so matched timings supplement the aggregate proxy. Estimates inform selection; encrypted measurements determine performance; Appendix B defines calibration and matching.

EncryptedEvaluator. Orion compiles the selected configuration and Lattigo executes encrypted inference. The evaluator records effective settings, output error, forward time, and enabled profiles. Together with parameter-security checks, these measurements establish feasibility and determine the best under Eq. (2). Compilation, analysis, and API calls are separate search costs.

3.3 Agent Roles and Evidence Exchange

The multi-agent controller separates proposal, selection, and outcome assessment, with global and layerwise proposal responsibilities reflecting the scope of configuration changes. The roles share an underlying LLM and structured search state; they do not require separate trained models or a separate call for every role. Appendix A maps these roles to the evaluated implementation. Section 3.4 describes how the roles interact across the three phases.

Initialization: InitAgent and RegimeAgent. InitAgent receives the model, supported fields, constraints, and reference examples, and proposes complete configurations with rationales. After tool analysis, RegimeAgent compares each candidate's effective settings, numerical diagnostics, runtime estimates, and available prior measurements, and selects an admitted candidate for encrypted evaluation. Admission enforces interface and preliminary parameter checks; measured feasibility is established only after execution. Missing evidence and analysis failures remain visible, and references guide generation rather than replace it with a stored answer.

Refinement: GlobalTradeoffAgent and LayerwiseAgent. GlobalTradeoffAgent considers network-wide parameter and packing revisions using the current result, verified best, and history. LayerwiseAgent considers supported embedding and BSGS overrides for eligible layers. These responsibilities can jointly produce global and local alternatives in one call; selecting a local change does not freeze the global search space. For example, rotation-heavy layers can motivate a BSGS revision whose execution plan affects other layers. Each proposal is a complete configuration, so tools can check these interactions and materialize effective settings.

Selection and feedback: PatchGateAgent. PatchGateAgent compares refinement candidates using their changes, tool evidence, and prior outcomes, then selects a candidate for evaluation. After execution, an assessment call interprets measured quality, runtime, and failures and recommends continuing or stopping. This assessment enters shared history for the next proposal; it does not itself generate the next configuration. Despite the role's name, admission, measured feasibility, and best-result updates are deterministic checks outside the LLM's authority.

The agents reason over configuration alternatives, evidence gaps, and unsuccessful attempts, while the backend retains numerical computation and scheduling. A runtime proxy alone cannot supply this interpretation: a candidate can change precision and cost, and its estimate may cover only part of the work. The LLM roles weigh these qualifications when proposing and selecting, without guaranteeing optimal decisions. Shared history retains effective configurations, outcomes, and failure reasons, including failed or slower trials that leave the best unchanged, updating the agents' information, not their weights. Appendix H details effective-setting checks and action spaces.

3.4 THREE-PHASE WORKFLOW

The three phases are coupled modes of one outer loop, not a pipeline of independent optimizers. Every iteration runs the same propose, analyze, admit, select, evaluate, and assess steps over shared tools, history, and runtime profiles; the phase sets the comparison baseline and the leading roles.

Phase A: Propose and analyze. Initial and refinement proposals both pass through this analysis. Static checks, structural plans, numerical diagnostics, and runtime estimates are computed per candidate; parameter or security violations return to the proposing role for correction, so rejected candidates consume no encrypted evaluation.

Phase B: Establish a verified best. Until a feasible result exists, selected candidates are evaluated under encryption and failures return to Phase A as feedback for new proposals. The first result meeting the security and numerical requirements becomes the verified best; this first feasible output is itself a framework result, not an external baseline.

Phase C: Refine with feedback. Refinement proposals revise the verified best, and global and local alternatives are analyzed and compared together: a local packing change can alter levels or the bootstrap schedule and prompt a global revision, while a global change is re-profiled per layer before the next local proposal. Each completed measurement enters history, refreshes the runtime profiles used by later analysis, and informs the continue-or-stop assessment. Only feasible measured improvements replace the best; the configured policy and budget end the loop.

Algorithm 1 Agent-guided CKKS configuration search

Require: Model M, constraints G, profiles D_0, stopping limits $\mathcal{L}$
Ensure: Best verified configuration C^*, or $\perp$

$\mathcal{H} \leftarrow [\,],\ C^* \leftarrow \perp,\ b \leftarrow 0,\ D \leftarrow D_0,\ \sigma \leftarrow \text{continue}$
while $\mathrm{Continue}_{\mathcal{L}}(\mathcal{H}, b, \sigma)$ **do**
  $\mathcal{P} \leftarrow \mathrm{LLMPropose}(M, G, \mathcal{H}, C^*)$
  $E \leftarrow \mathrm{Analyze}(\mathcal{P}; M, G, D), \quad \mathcal{A} \leftarrow \mathrm{Admit}(\mathcal{P}, E, G)$
  $\mathcal{H} \leftarrow \mathcal{H} \| [(\mathcal{P}, E, \mathcal{A})]$
  if $\mathcal{A} = \varnothing$ **then continue**
  $C \leftarrow \mathrm{LLMSelect}(\mathcal{A}, E, \mathcal{H})$
  if $C = \perp$ **then continue**
  $(C, R) \leftarrow \mathrm{FHEEvaluate}(C), \quad b \leftarrow b + n_{\mathrm{FHE}}(R)$
  $\mathcal{H} \leftarrow \mathcal{H} \| [(C, R)]$
  $D \leftarrow \mathrm{UpdateProfiles}(D, R)$
  $(\sigma, \mathcal{H}) \leftarrow \mathrm{LLMAssess}(\mathcal{H}, C^*; G)$
  if $\mathrm{Improves}(C, R, C^*; G)$ **then** $C^* \leftarrow C$
end while
return C^*

Algorithm 1 makes this coupling explicit: Phases A–B repeat until a feasible result exists, and the same loop then continues as Phase C. Its proposal, selection, and assessment operators group the agent responsibilities of Section 3.3, not one call per role. $\mathcal{P}$ is the proposal set, E its analysis records, and $\mathcal{A}$ the admitted subset. $\mathrm{Analyze}$ checks interfaces and parameters, then obtains candidate evidence and checks the resulting cryptographic contexts; Admit applies the required checks and keeps missing optional evidence explicit. Evaluation returns the effective configuration C and outcome R; invalid selection returns $\perp$. Initial profiles D_0 may be empty; analysis reads the current profiles D, updated only from complete encrypted measurements with required metadata. History $\mathcal{H}$ appends ($\|$) every attempt and outcome, b counts actual FHE calls $n_{\mathrm{FHE}}(R)$, and $\mathrm{LLMAssess}$ adds its interpretation to the latest record and returns a recommendation σ. $\mathrm{Improves}$ requires verified feasibility and either no incumbent or a better measured objective under Eq. (2), with tie-breaking as in Appendix H. $\mathrm{Continue}_{\mathcal{L}}$ enforces finite limits and the policy for σ (Appendix A).

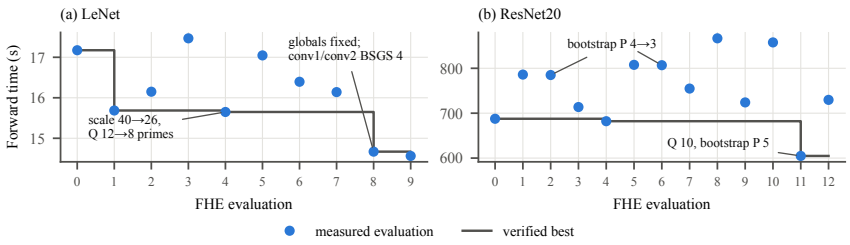

Figure 2: Measured forward time per evaluation (points) and verified best (step line) in the searches behind the LeNet and ResNet20 results of Table 1. Annotations mark the LeNet global step and fixed-global BSGS step, and the two rejected ResNet20 bootstrap-P 4→3 proposals; Section 4.3 analyzes them.

4 Evaluation

We evaluate the CKKS configurations that automated exploration produces, where their measured time changes, and how tool evidence and measured feedback drive agent decisions.

4.1 Experimental Protocol and Automation Scope

Users supply model/input assets, constraints, supported fields, references, and an evaluation budget; FHE-Agent automates proposal, analysis, selection, encrypted execution, feedback, and verified-best tracking. The fixed Orion workloads are LeNet, AlexNet, and ResNet20, using quadratic, polynomial sigmoid linear unit (SiLU), and polynomial rectified linear unit (ReLU) activations, respectively; inputs are $1\times1\times28\times28$ for LeNet and $1\times3\times32\times32$ otherwise. Initialization does not load a historical best configuration. Searches use Gemma with Orion, the `baahl-nyu/lattigo/v6` v6.2.0 fork, and Go 1.23.12 on dual 48-core AMD EPYC 9454 CPUs; the LeNet method comparison holds GPT-6 Astra (high reasoning effort), GPT-6 Luna (xhigh), and Gemma fixed. FHE-Agent receives four reference-only templates and candidate analyses; the general-purpose agent receives one LeNet template and invokes backend tools itself. LeNet framework runs start without saved calibration; Gemma searches use archived profiles. These are complete-method comparisons, not identical-information ablations (Appendices A and E).

MAE compares plaintext and decrypted outputs; $p = -\log_2(\mathrm{MAE})$ measures numerical agreement, not classification accuracy. Acceptance requires backend parameter-security support of at least 128 bits, completed encrypted execution, $p \geq 6$, and MAE $\leq 10^{-3}$; three searches used a 10^{-2} MAE gate during search, and every configuration reported here also meets 10^{-3}. Times are single diagnostic encrypted forward measurements on shared CPUs, excluding compilation, analysis, and LLM calls; budgets count actual FHE evaluations (Appendices C and E).

4.2 End-to-End Automated Configuration Exploration

Table 1: Best configuration found by FHE-Agent for each workload; all meet the security, precision, and MAE requirements. The last column is that search's first feasible and best time.

Workload	Best (s)	Bits	MAE	First→best (s)
LeNet	14.56	10.4	7.52×10^{-4}	17.17→14.56
AlexNet	267.66	21.7	2.94×10^{-7}	267.66→267.66
ResNet20	604.79	13.9	6.51×10^{-5}	687.59→604.79

Table 1 reports the best configuration FHE-Agent found for each workload. In the searches that found them, refinement lowers LeNet and ResNet20 forward time by 15.2% and 12.0% from the first feasible configuration. The fastest AlexNet configuration is already the first feasible output of its search; in another search, refinement lowers AlexNet from 333.98 to 288.36 s (13.7%). Searches differ in code revision and conditions and are not controlled comparisons (Table 7). In Figure 2,

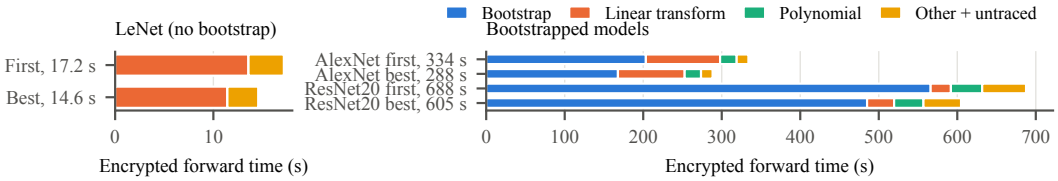

Figure 3: First-feasible versus best forward time by operation (single traced runs; values in Table 9).

Table 3: Configuration comparisons (s); all pass the gates. (a) Only bootstrap P changes (+14.1%, +14.2%, +18.3%). (b) ResNet20 initial candidates. (c) LeNet trials 4→8: only conv1/conv2 BSGS changes, 2→4 (best trial 9 also sets fc1/fc2).

(a) Bootstrap auxiliary primes

Pair, trials	Four	Three
AlexNet, 0→1	267.66	305.49
ResNet20, 0→2	687.59	785.01
ResNet20, 4→6	682.15	806.87

(b) ResNet20 initial candidates

$\lvert Q\rvert$	$\lvert P_{\text{boot}}\rvert$	Forecast	Measured
11	4	334.05	729.69
10	5	351.33	604.79
10	4	351.33	857.65

(c) LeNet local refinement

Layer	Before	After
conv1	4.574	4.022
conv2	6.411	5.992
fc1 (unchanged)	1.520	1.510
fc2 (unchanged)	0.658	0.650
Forward	15.649	14.670

slower or failed proposals never replace the verified best (Section 4.3); another AlexNet search finds no feasible result, and a ResNet20 search keeps its initialization as best (Appendices D and G).

Comparison with the same LLM. Table 2 compares four methods on the same LeNet assets: three independent or three feedback-driven direct generations (one template, no tools), a general-purpose agent with backend commands, and FHE-Agent. FHE-Agent is the only method whose attempts all pass under every LLM (9/9). Its configurations compile in 163–173 versus 239–255 s and use 38–41 versus 44–50 GB peak memory; five-run mean inference is fastest for Astra and Gemma and within 1% for Luna. Faster rejected attempts have unknown security support, not established insecurity. Framework wall times are 27.4/30.6 versus 17.8/18.4 minutes for the general agent, so equal FHE counts do not equalize cost (Appendix I). Astra's framework MAE is 1.02×10^{-7} versus 1.98×10^{-4} for the general agent; Appendix F gives the earlier prototype comparison with best-of-ten one-shot runs.

Table 2: LeNet same-LLM pilots: inference time (s) / compile time (s) / peak memory (GB) of each method's best accepted configuration (five-run mean inference; accepted/attempted); —: none accepted. All meet the gates (Table 12).

Method	Astra	Luna	Gemma
Independent ×3	18.1/242/50 (1/3)	16.7/243/47 (2/3)	20.6/239/49 (2/3)
Direct feedback ×3	— (0/3)	17.4/255/49 (2/3)	17.6/240/44 (3/3)
General agent	15.7/244/46 (2/3)	16.7/243/47 (2/3)	16.7/243/47 (3/3)
FHE-Agent	**15.6/163/38** (3/3)	16.9/**171/40** (3/3)	**16.6/173/41** (3/3)

4.3 Configuration Effects and Measured Feedback

Where measured time changes. Figure 3 decomposes first-to-best differences. Linear transforms account for 82% of the LeNet gap; bootstrap time accounts for 78% of the AlexNet gap and 98% of the ResNet20 gap. These shares describe observed differences, not causal contributions. The LeNet gain has a global and a local step: lowering scale 40→26 and Q from twelve to eight primes gives 17.17→15.65 s, and BSGS overrides with fixed global parameters give 15.65→14.56 s. Global settings also trade components: one more main P prime, inherited by bootstrapping, saves AlexNet 20.89 s of bootstrap time but adds 12.63 s of linear-transform time; square embedding removes ResNet20 rotations but adds 44.34 s of linear-transform time (Appendix E). The LeNet search trades precision for speed: MAE rises from 1.02×10^{-7} to 7.52×10^{-4}, within its gate (Appendix C.2).

Bootstrap proposals and selection. Table 3(a) isolates four-to-three bootstrap auxiliary-prime changes. In the ResNet20 search, GlobalTradeoffAgent identifies bootstrap overhead (logged as

82% of runtime) as the bottleneck and predicts that fewer auxiliary primes will reduce cost; the selected candidate instead takes 785.01 s, and a later proposal repeats the hypothesis despite this result, measuring 806.87 s. Shared feedback thus does not ensure that agents correct a mistaken hypothesis; the deterministic update rule preserves the verified best in both cases. The first batch also proposes, but does not select, a shorter-Q/five-prime direction, so proposal and PatchGateAgent selection diverge here (Appendix E.1). The final configuration (Table 3(b)) combines ten Q and five bootstrap P primes. Between two initial candidates with ten Q primes, changing only bootstrap P from four to five primes gives 857.65→604.79 s with the same 38 bootstrap calls; bootstrap time accounts for 80.0% of this pair's gap. The slower run has long bootstrap and polynomial events, so this pair does not establish a stable 29.5% speedup; the first-to-final comparison changes Q and P.

Layer-local proposals and measured effects. In the LeNet search, LayerProfiler evidence marks conv2 (48% of runtime) and conv1 (34%) as actionable, and after a slower ratio-8 attempt LayerwiseAgent suggests BSGS ratio 4 on both. Against the same global configuration without these overrides (Table 3(c)), the two modified layers account for about 99% of the 0.979 s reduction, while unchanged layers vary by at most 0.010 s. The rationale anticipates fewer rotations, yet the standalone `rotate` count stays at 11: the saving occurs inside linear transforms, so profiling localizes the effect independently of the agent's hypothesis. An independent three-pair study on another LeNet configuration also records lower conv1 times with ratio 4 (medians 4.617 versus 4.017 s), but whole-network ranges overlap, so neither establishes a general end-to-end speedup. Other local trials fail to improve: Luna keeps its initialization, and a later AlexNet trial is faster overall though its modified layer is not, so neither counts as a layer-local gain (Appendix E.1).

4.4 Evidence for Automated Decisions

Numerical evidence. A retrospective diagnostic of the first composite ReLU tracks local approximation error: changing range margin from 2 to 3 raises offline MAE from 3.9171×10^{-5} to 1.0059×10^{-4}, matching decoded FHE endpoint errors 3.9197×10^{-5} and 1.0062×10^{-4}. This is local evidence, not final-network error prediction. AlexNet estimates miss in both directions: one predicts 17.2 bits but measures 4.2 bits (MAE 5.37×10^{-2}), another predicts MAE 0.001741 but measures 4.77×10^{-6}; encrypted verification remains necessary (Figure 7).

Runtime evidence and selection. Compatible prior measurements describe some matched components accurately: for a square-embedding ResNet20 candidate, bootstrap and polynomial aggregates have 1.0% and 2.4% errors, and matched work covers 80.0% of network time (not the bootstrap share of that pair's gap). A local-BSGS candidate matches 516/578 calls but only 7.4% of time, so unmatched work cannot be treated as free. Whole-candidate ranking is weaker: in Table 3(b), forecasts are 42–59% below the measured times, equal low-confidence forecasts conceal different measured times, and the forecast order misses the fastest candidate, and for three feasible AlexNet initial candidates (295.16–300.21 s) the predictor abstains (Appendix E).

5 Conclusion

FHE-Agent coordinates LLM agent roles for proposal, global/local revision, selection, and assessment through a three-phase workflow over shared tool evidence and history, while deterministic checks preserve the verified best. Its searches find feasible configurations for all three workloads, including 604.79 s for ResNet20. Operation-level profiles place most measured gains in bootstrapping for the bootstrapped models and in linear transforms for LeNet; paired changes isolate the bootstrap auxiliary-prime effect and localize BSGS savings to the modified layers. Traces also expose limits of LLM judgment: a refuted bootstrap hypothesis was proposed again, and offline forecasts misordered candidates, so encrypted validation and deterministic best updates remain essential. Every accepted configuration is thus backed by an encrypted measurement that passed the security and numerical checks, which makes each search auditable. In same-LLM LeNet pilots, FHE-Agent is the only method whose attempts all pass under three LLMs. Single-run timings and fixed inputs limit generalization; repeated deployment measurements, matched external baselines, same-information ablations, and more models and backends remain future work.

AI USE STATEMENT

The framework itself uses an LLM for configuration proposals, candidate selection, and feedback interpretation (Appendix A). The authors also used AI coding and writing assistants to help refine and implement the framework, design and analyze experiments, prepare figures, and draft and edit the manuscript. The authors reviewed these outputs and take full responsibility for the content; all reported measurements come from Orion/Lattigo runs, not LLM output.

REFERENCES

Ehud Aharoni, Allon Adir, Moran Baruch, Nir Drucker, Gilad Ezov, Ariel Farkash, Lev Greenberg, Ramy Masalha, Guy Moshkowich, Dov Murik, Hayim Shaul, and Omri Soceanu. HeLayers: A Tile Tensors Framework for Large Neural Networks on Encrypted Data. *Proceedings on Privacy Enhancing Technologies*, 2023(1):325–342, January 2023. ISSN 2299-0984. doi: 10.56553/popets-2023-0020. URL https://petsymposium.org/popets/2023/popets-2023-0020.php.

Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. OpenFHE: Open-source fully homomorphic encryption library. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pp. 53–63. ACM, 2022. doi: 10.1145/3560827.3563379. URL https://doi.org/10.1145/3560827.3563379.

Wei Ao and Vishnu Naresh Boddeti. AutoFHE: Automated adaption of CNNs for efficient evaluation over FHE. In *33rd USENIX Security Symposium*, pp. 2173–2190. USENIX Association, 2024. URL https://www.usenix.org/conference/usenixsecurity24/presentation/ao.

Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In *Advances in Cryptology – EUROCRYPT 2021*, pp. 587–617. Springer, 2021. doi: 10.1007/978-3-030-77870-5_21. URL https://doi.org/10.1007/978-3-030-77870-5_21.

Cory Brynds, Parker McLeod, Lauren Caccamise, Asmita Pal, Dewan Saiham, Sazadur Rahman, Joshua San Miguel, and Di Wu. CryptOracle: A modular framework to characterize FHE. In *IEEE International Symposium on Performance Analysis of Systems and Software*, 2026. URL https://www.ispass.org/ispass2026/program.php.

José Cabrero Holgueras. *Towards Improved Homomorphic Encryption for Privacy-Preserving Deep Learning*. PhD thesis, Carlos III University of Madrid, March 2023. URL https://hdl.handle.net/10016/38266.

José Cabrero-Holgueras and Sergio Pastrana. Towards automated homomorphic encryption parameter selection with fuzzy logic and linear programming. *Expert Systems with Applications*, 229: 120460, 2023. doi: 10.1016/j.eswa.2023.120460. URL https://doi.org/10.1016/j.eswa.2023.120460.

Edward Chen, Fraser Brown, and Wenting Zheng. Bridging usability and performance: A tensor compiler for autovectorizing homomorphic encryption. In *35th USENIX Security Symposium*, pp. 141–160. USENIX Association, 2026. URL https://www.usenix.org/conference/usenixsecurity26/presentation/chen-edward.

Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library – SEAL v2.1. In *Financial Cryptography and Data Security: FC 2017 International Workshops*, pp. 3–18. Springer, 2017. doi: 10.1007/978-3-319-70278-0_1. URL https://doi.org/10.1007/978-3-319-70278-0_1.

Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2019*, pp. 34–54. Springer, 2019. doi: 10.1007/978-3-030-17656-3_2. URL https://doi.org/10.1007/978-3-030-17656-3_2.

Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. THE-X: Privacy-preserving transformer inference with homomorphic encryption. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 3510–3520, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.277. URL https://aclanthology.org/2022.findings-acl.277/.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, pp. 409–437. Springer, 2017. doi: 10.1007/978-3-319-70694-8_15. URL https://doi.org/10.1007/978-3-319-70694-8_15.

Seonyoung Cheon, Yongwoo Lee, Dongkwan Kim, Ju Min Lee, Sunchul Jung, Taekyung Kim, Dongyoon Lee, and Hanjun Kim. DaCapo: Automatic bootstrapping management for efficient fully homomorphic encryption. In *33rd USENIX Security Symposium*, pp. 6993–7010. USENIX Association, 2024. URL https://www.usenix.org/conference/usenixsecurity24/presentation/cheon.

Seonyoung Cheon, Yongwoo Lee, Hoyun Youm, Dongkwan Kim, Sungwoo Yun, Kunmo Jeong, Dongyoon Lee, and Hanjun Kim. HALO: Loop-aware Bootstrapping Management for Fully Homomorphic Encryption. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 572–585, Rotterdam Netherlands, March 2025. ACM. doi: 10.1145/3669940.3707275. URL https://dl.acm.org/doi/10.1145/3669940.3707275.

Wonseok Choi, Jongmin Kim, and Jung Ho Ahn. Cheddar: A swift fully homomorphic encryption library designed for GPU architectures. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 35–49, 2026. doi: 10.1145/3760250.3762223. URL https://doi.org/10.1145/3760250.3762223.

Meghan Cowan, Deeksha Dangwal, Armin Alaghi, Caroline Trippel, Vincent T. Lee, and Brandon Reagen. Porcupine: A synthesizing compiler for vectorized homomorphic encryption. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, pp. 375–389, New York, NY, USA, June 2021. Association for Computing Machinery. ISBN 978-1-4503-8391-2. doi: 10.1145/3453483.3454050. URL https://dl.acm.org/doi/10.1145/3453483.3454050.

Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin Lauter, Saeed Maleki, Madan Musuvathi, and Todd Mytkowicz. CHET: An optimizing compiler for fully-homomorphic neural-network inferencing. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pp. 142–156, 2019. doi: 10.1145/3314221.3314628. URL https://doi.org/10.1145/3314221.3314628.

Roshan Dathathri, Blagovesta Kostova, Olli Saarikivi, Wei Dai, Kim Laine, and Madan Musuvathi. EVA: An Encrypted Vector Arithmetic Language and Compiler for Efficient Homomorphic Computation. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 546–561, June 2020. doi: 10.1145/3385412.3386023. URL https://doi.org/10.1145/3385412.3386023.

Austin Ebel, Karthik Garimella, and Brandon Reagen. Orion: A fully homomorphic encryption framework for deep learning. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 734–749. ACM, 2025. doi: 10.1145/3676641.3716008. URL https://doi.org/10.1145/3676641.3716008.

Eugene Frimpong, Khoa Nguyen, Mindaugas Budzys, Tanveer Khan, and Antonis Michalas. GuardML: Efficient privacy-preserving machine learning services through hybrid homomorphic encryption. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, SAC '24, pp. 953–962, Avila, Spain, 2024. Association for Computing Machinery. doi: 10.1145/3605098.3635983. URL https://doi.org/10.1145/3605098.3635983.

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, pp. 201–210. PMLR, 2016. URL https://proceedings.mlr.press/v48/gilad-bachrach16.html.

Shruthi Gorantala, Jianming Tong, Asra Ali, Baiyu Li, Jonathan Katz, Jeremy Kun, Thomas Steinke, Abhradeep Thakurta, Julian Walker, and Amir Yazdanbakhsh. Adapting AlphaEvolve to optimize fully homomorphic encryption on TPUs, 2026. URL https://arxiv.org/abs/2605.14718.

Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N. Wright. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies*, 2018(3): 123–142, 2018. doi: 10.1515/popets-2018-0024. URL https://petsymposium.org/popets/2018/popets-2018-0024.php.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1651–1669, Baltimore, MD, USA, 2018. USENIX Association. URL https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar.

Aleksandar Krastev, Nikola Samardzic, Simon Langowski, Srinivas Devadas, and Daniel Sanchez. A tensor compiler with automatic data packing for simple and efficient fully homomorphic encryption. *Proceedings of the ACM on Programming Languages*, 8(PLDI), 2024. doi: 10.1145/3656382. URL https://doi.org/10.1145/3656382.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc. URL https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Mayank Kumar, Jiaqi Xue, Mengxin Zheng, and Qian Lou. FHE-Coder: Benchmarking secure agentic code generation for fully homomorphic encryption. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4F1py5vQXm.

Lattigo. Lattigo v5.0.2. Online: https://github.com/tuneinsight/lattigo/releases/tag/v5.0.2, November 2023. EPFL-LDS, Tune Insight SA.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL https://doi.org/10.1109/5.726791.

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, pp. 12403–12422. PMLR, 2022. URL https://proceedings.mlr.press/v162/lee22e.html.

Yongwoo Lee, Seonyoung Cheon, Dongkwan Kim, Dongyoon Lee, and Hanjun Kim. ELASM: Error-Latency-Aware scale management for fully homomorphic encryption. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 4697–4714. USENIX Association, 2023. URL https://www.usenix.org/conference/usenixsecurity23/presentation/lee-yongwoo.

Yan Liu, Jianxin Lai, Long Li, Tianxiang Sui, Linjie Xiao, Peng Yuan, Xiaojing Zhang, Qing Zhu, Wenguang Chen, and Jingling Xue. ReSBM: Region-based scale and minimal-level bootstrapping management for FHE via min-cut. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 924–939. ACM, 2025. doi: 10.1145/3669940.3707276. URL https://doi.org/10.1145/3669940.3707276.

Qian Lou, Song Bian, and Lei Jiang. AutoPrivacy: Automated layer-wise parameter selection for secure neural network inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/6244b2ba957c48bc64582cf2bcec3d04-Abstract.html.

Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. BumbleBee: Secure two-party inference framework for large transformers. In *Proceedings 2025 Network and Distributed System Security Symposium*, San Diego, CA, USA, 2025. Internet Society. doi: 10.14722/ndss.2025.230057. URL https://doi.org/10.14722/ndss.2025.230057.

Benjamin Mikek, Danylo Vashchilenko, Bryan Lu, and Panpan Xu. Agentic code optimization via compiler-LLM cooperation, 2026. URL https://arxiv.org/abs/2604.04238.

Kevin Nam, Youyeon Joo, Dongju Lee, Seungjin Ha, Hyunyoung Oh, Hyungon Moon, and Yunheung Paek. LOHEN: Layer-wise optimizations for neural network inferences over encrypted data with high performance or accuracy. In *34th USENIX Security Symposium*, pp. 5583–5600. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/nam-lohen.

Dongjin Park, Eunsang Lee, and Joon-Woo Lee. Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11090–11111, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.543. URL https://aclanthology.org/2025.acl-long.543/.

Yun-Soo Park, Hyunmin Choi, Hyoungshick Kim, and Mun-Kyu Lee. HE-DAP: Homomorphic encryption-based dynamic adaptive parameter optimization for statistical computation. In *Proceedings of the 41st ACM/SIGAPP Symposium on Applied Computing*, pp. 1848–1856. ACM, 2026. doi: 10.1145/3748522.3779981. URL https://doi.org/10.1145/3748522.3779981.

Ran Ran, Xinwei Luo, Wei Wang, Tao Liu, Gang Quan, Xiaolin Xu, Caiwen Ding, and Wujie Wen. SpENCNN: Orchestrating encoding and sparsity for fast homomorphically encrypted neural network inference. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 28718–28728. PMLR, 2023. URL https://proceedings.mlr.press/v202/ran23b.html.

Dmitry Redko, Albert Fazlyev, Konstantin Sozykin, Maria Ivanova, Evgeny Burnaev, and Egor Shvetsov. Prior knowledge or search? a study of LLM agents in hardware-aware code optimization, 2026. URL https://arxiv.org/abs/2605.19782.

Bilel Sefsaf, Abderraouf Dandani, Abdessamed Seddiki, Arab Mohammed, Eduardo Chielle, Michail Maniatakos, and Riyadh Baghdadi. CHEHAB RL: Learning to optimize fully homomorphic encryption computations. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 342–360. ACM, 2026. doi: 10.1145/3779212.3790138. URL https://doi.org/10.1145/3779212.3790138.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Alexander Viand, Patrick Jattke, Miro Haller, and Anwar Hithnawi. HECO: Fully homomorphic encryption compiler. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 4715–4732. USENIX Association, 2023. URL https://www.usenix.org/conference/usenixsecurity23/presentation/viand.

Junping Wan, Yucen Liao, Yinglong Liao, Zejiu Tan, Jinming Xu, Zoe L. Jiang, and Binxing Fang. CipherAPR: Accelerating RNS-CKKS encrypted inference via importance-guided and level-aware mixed-degree polynomial design. *Cryptography*, 10(4):44, 2026. doi: 10.3390/cryptography10040044. URL https://doi.org/10.3390/cryptography10040044.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Qiuchu Yu, Ruiyuan Xu, and Guangli Li. FHECrafter: A multi-agent framework for automated fully homomorphic encrypted tensor program generation. In *Proceedings of the 40th ACM International Conference on Supercomputing - Workshops*, pp. 55–59. ACM, 2026. doi: 10.1145/3774895.3812197. URL https://doi.org/10.1145/3774895.3812197.

Zikai Zhou, William Seo, Edward Chen, Alex Ozdemir, Fraser Brown, and Wenting Zheng. Orbit: Optimizing rescale and bootstrap placement with integer linear programming techniques for secure inference. In *35th USENIX Security Symposium*, pp. 61–80. USENIX Association, 2026. URL https://www.usenix.org/conference/usenixsecurity26/presentation/zhou.

A AGENT AND TOOL INTERFACES

The five agent names in Figure 1 identify architectural responsibilities, not five separately trained models or a fixed sequence of five LLM calls. InitAgent proposes initial configurations; RegimeAgent selects an initial candidate using tool evidence; GlobalTradeoffAgent proposes global revisions; LayerwiseAgent proposes supported layer-specific overrides; and PatchGateAgent handles refinement selection and outcome assessment. These roles share trial history and the verified best.

In the implementation, one proposal module serves the initial, global, and layerwise proposal roles, and a separate selection/assessment module serves selection and outcome assessment through distinct role-specific calls. Workflow graph nodes coordinate these interfaces and backend execution. Deterministic admission, feasibility checks, and best-result comparison remain separate from LLM reasoning; the outcome-assessment prompt asks for interpretation and continuation, leaving configuration proposals to the next proposal call.

For a fresh search, the initial request contains the model description, supported parameter fields, numerical and security constraints, and explicitly identified reference templates. Selection requests contain the candidate configurations and differences, structural or numerical analysis, runtime-estimate provenance, and failures of analysis where present. Refinement requests additionally contain the current outcome, previous verified best, and historical configuration/outcome summaries. Reference-template availability and preexisting timing profiles are separate experimental settings; neither is equivalent to loading a historical best as the initial answer.

Full local records retain the input and output of each LLM call and the tool evidence associated with the selected effective configuration. Current and best observations retain relevant layer detail; historical summaries preserve configurations, measured outcomes, prediction summaries, and failure reasons.

LLM settings. The Gemma experiments use the API model identifier `gemma-4-31b`, with reasoning effort requested as `high`. Configuration generation uses temperature 0.7, candidate selection 0.2, and outcome judging 0.3. Each of these calls requests at most 16,384 output tokens. These are client request settings; the provider controls the served weights and decoding implementation. The same model serves all roles, and the framework does not train or fine-tune it. Settings for the GPT-6 models are given with the same-LLM comparison.

Input and response formats. Each call sends a role-specific prompt as a user message. Generation requests ask for three to five complete configurations, each delimited by `=== CONFIG ... ===` and `=== END ===`. Required flat fields include `LogN`, `LogScale`, explicit `LogQ`/`LogP` lists, `H`, ring type, embedding method, margin, and backend, together with the proposed objective and rationale. Optional bootstrap and layer overrides have their own documented fields. A complete response is still a proposal, not evidence of feasibility.

Selection returns a JSON object with `selected_index` and `reasoning`; the index refers to the supplied candidate list. The judge returns an assessment of the measured outcome, the next phase, and a stop or continue decision. Its assessment does not replace the deterministic feasibility conditions in Section 3.1. Archived prompts preserve the actual model information, constraints, reference examples, configurations, tool evidence, and history visible at each decision, together with the corresponding replies and parser outcomes.

Termination and cost accounting. The total-trial limit counts records not marked as rejected, rather than physical FHE invocations or elapsed time. Experiments report actual calls and time independently and specify whether early controller stops are honored or iterations are forced. Proposal corrections, compilation attempts, and encrypted evaluations are counted separately. A proposal rejected before encryption is not an FHE invocation, but its analysis and API time contribute to search cost. Termination preserves the best result satisfying the feasibility conditions in Section 3.1.

Algorithm 1 abstracts these limits as $\mathcal{L}$. Its history is an ordered sequence: appending an identical rejected proposal still records a new attempt. Thus the continuation predicate can bound attempts even when no FHE call occurs. Selection corrections are bounded and return no valid choice on exhaustion. An evaluation outcome retains failures as well as successful measurements; a failure before

encrypted execution adds zero to b, while an invoked but failed encrypted evaluation counts as a call. A successful outcome stores backend-effective settings with its measurements. These distinctions prevent proposal retries from being mistaken for encrypted evaluations or requested overrides for effective configurations. Profile updates require successful, complete encrypted measurements with valid runtime and metadata, not necessarily a result meeting the numerical feasibility thresholds. The judge sees the current outcome and the best retained on entry; deterministic best comparison follows its assessment. Hard stopping boundaries and security-preflight rejections can bypass the LLM judge call; ordinary controller stops remain subject to the configured policy.

B Runtime Estimation Details

Aggregate features and calibration. The aggregate estimate uses three coarse cost features, $\phi(C) = (\phi_M, \phi_R, \phi_B)$, for multiplication-related work, rotations, and bootstraps. A fixed reference vector w_0 defines cost units $u(C) = w_0^\top \phi(C)$. The profile-scaled prediction is

$$\widehat{T}_{\mathrm{agg}}(C) = a_g\, u(C), \qquad a_g = \frac{1}{|J_g|} \sum_{j \in J_g} \frac{T_{\mathrm{FHE}}(C_j)}{u(C_j)}, \tag{3}$$

where the nonempty set J_g contains prior measurements with positive cost units in parameter group g, and a_g is seconds per cost unit. Estimates report the match type, sample count, and unmatched dimensions. The reference weights in Eq. (3) are $w_0 = (1, 0.15, 8)$; they define cost units rather than seconds. Multiplication-related cost aggregates linear-transform diagonal work and coarse activation work. Rotation cost uses the compiled baby-step/giant-step groups when available. Bootstrap cost uses ciphertext counts and slots when that metadata is available; otherwise it falls back to a per-operation estimate using level and slots. The feature weights and level-dependent cost functions are heuristic; their units are kept fixed when collecting the ratios in Eq. (3). In particular, activation depth is not reinterpreted as the number of internal polynomial multiplications.

The lookup first tries the full parameter-group key. It then tries a reduced match that retains the auxiliary-prime dimensions while allowing secret-weight and range-margin differences, followed by broader stored groups. The chosen group and dimensions omitted by a broad match are returned with the estimate. Each stored group averages its observed runtime-to-cost ratios. This grouping is distinct from operation-dependent state matching used for the partial timings in Eq. (4).

When no such profile is available, a calibrated three-feature model supplies an estimate if it has been fitted in the same feature space. A stored global runtime scale, when available, takes precedence over that feature-model value; a matching empirical correction can then adjust the result. These fallback sources are identified separately. A cold, uncalibrated cost is not reported as elapsed time. Profile scaling applies only to the aggregate estimate, not to the operation contributions, which are already in seconds.

Operation evidence. Operation-duration reuse is enabled separately from aggregate calibration. Its measurements come from earlier completed trials in the same run with a matching execution-context identity. Matching uses environment, moduli, scale, operands, and relevant packing or bootstrap settings. Dependencies are operation-specific: auxiliary-prime changes affect key switching without invalidating unrelated additions or rescales. For candidate C, let $n_C(o, s)$ count planned backend calls of family o with matching state s. When operation profiling is enabled, $D_{<t}(o, s)$ contains durations available before decision t from earlier completed trials in the same run and execution context. The matched contribution is

$$\widehat{T}_o^{\mathrm{matched}}(C) = \sum_{s \in \mathcal{M}_{C,o}} n_C(o, s)\ \mathrm{median}\, D_{<t}(o, s), \tag{4}$$

where $\mathcal{M}_{C,o}$ contains states with compatible measurements. Planned, matched, and supporting sample counts describe coverage. This partial contribution excludes unmatched and uninstrumented work, so it accompanies rather than replaces the aggregate estimate. The selector compares both estimates with the configuration changes; encrypted measurements determine acceptance. A linear-transform block and a polynomial evaluation are timed backend calls. The polynomial call already includes its internal multiplications and rescales; adding estimated internal timings to that call would double-count the work. Missing states remain missing contributions rather than zero-cost operations.

The comparison between predicted and measured components uses the same operation families and state groups, and reports time coverage separately from call coverage.

C Experiment Provenance and Counts

The three earlier searches in Table 7 use three source revisions of the framework, one per workload. All 20 evaluations use the same instrumented Orion backend build, with the `baahl-nyu/lattigo/v6` v6.2.0 fork and Go 1.23.12; this is distinct from the prototype's Lattigo v5.0.2. The operation-state reconstruction and partial timing analysis in Table 11 use a later source revision with the archived ResNet20 configurations and encrypted traces. These versions are kept separate so that earlier successful search trajectories are not presented as measurements of later component changes. The activation metadata and candidate-evidence projection described in Section 3 were refined after these searches, so the recorded trajectories do not evaluate those refinements end to end. The normalized evidence record and per-trial artifacts link each row to its configuration, model and input assets, parameter-security report, measured output errors, and timing records.

C.1 Automation Counts

For the three earlier searches, proposed configurations count complete configuration occurrences in parsed generation replies, including initialization, rejected setup batches, and repeat proposals before historical-configuration deduplication. LeNet, AlexNet, and ResNet20 have 10, 6, and 6 proposal rounds, containing 38, 23, and 24 occurrences. AlexNet and ResNet20 each include an initial batch returned for correction. These counts do not represent unique effective configurations, and proposals minus executions is not a rejection count.

Encrypted evaluations number 10, 5, and 5, all feasible under the recorded primary thresholds. Verified-best updates after initialization number 4, 2, and 1, respectively, matching the feasible running minima and saved final results.

C.2 Initial and Best Configurations

Table 4 gives the saved configuration fields for the initial and best results of the three earlier searches in Table 7. All six use `LogN`=16, the standard ring, hybrid embedding, range margin 2, and the Lattigo backend. The notation $b^{\times k}$ denotes k consecutive entries of bit length b, not multiplication of the bit lengths. The lists specify prime sizes, not the numerical values of the primes.

Table 4: Configuration fields corresponding to the recorded initial and best encrypted evaluations. "Omitted" means that no explicit `boot_params.LogP` override was saved; the recorded backend versions inherit application `LogP` in this case. LeNet has no bootstrap invocations.

Model	Stage	LogScale	H	LogQ	LogP	Bootstrap LogP
LeNet	Initial	40	192	$(55, 40^{\times 10}, 60)$	$(61^{\times 3})$	Omitted
	Best	26	192	$(48, 26^{\times 6}, 48)$	$(55^{\times 3})$	Omitted
AlexNet	Initial	40	192	$(60, 40^{\times 10})$	$(61^{\times 3})$	Omitted
	Best	32	32768	$(60, 32^{\times 10})$	$(61^{\times 4})$	Omitted
ResNet20	Initial	40	192	$(55, 40^{\times 7})$	$(61^{\times 3})$	$(61^{\times 6})$
	Best	38	192	$(53, 38^{\times 7})$	$(61^{\times 3})$	$(61^{\times 6})$

Only the LeNet best has explicit layer-local overrides in these six configurations: the baby-step/giant-step ratio is set to 4 for `conv1`, `conv2`, `fc1`, and `fc2`. The other five configurations contain no local overrides. These settings show why the initial-to-best differences are joint configuration comparisons, not separate estimates of the effect of scale, secret weight, or packing. Each configuration must be interpreted with its recorded backend version and fixed model/input assets; matching the configuration fields alone does not reproduce a measured time on another workload or execution environment.

C.3 Configuration Interpretation and Remaining Validation

AlexNet's $H = 32768$ is the recorded fixed Hamming weight of the main ternary secret, including the bootstrap main contexts; the bootstrap encapsulation secret separately has weight 32. The controller introduced $H = 32768$ with four application P primes in Trial 1, citing a larger admissible modulus budget, and retained both in the best Trial 3. Ring degree remained unchanged. Omitted bootstrap P inherits the application setting, so this also changes effective bootstrap P from three to four primes. The saved best has residual $\log_2(QP) = 624.0$ and bootstrap $\log_2(QP) = 1445.0$. Holding these saved contexts fixed, both fit below the recorded checker's $H = 192$ budget of 1550 bits; the earlier Trial 1 bootstrap product (1475.0 bits) also fits. Thus the archived budget arithmetic does not establish that the larger secret weight was necessary; we did not run an $H = 192$ variant of this configuration.

Isolating AlexNet's main secret weight or the extended ResNet20 search's local BSGS overrides, with effective bootstrap settings held fixed, is left to future work. Verified feasibility applies to the fixed evaluated inputs, not a held-out input distribution. LeNet's best MAE of 7.52×10^{-4} has about $1.33\times$ margin to the 10^{-3} acceptance threshold ($13.3\times$ to the 10^{-2} gate used during that search), so multi-input numerical validation and nearby scale settings are natural follow-up studies.

D Additional Instrumented Searches

Table 5 reports two five-evaluation searches using the same later revision, fresh LLM proposals, fixed per-model assets, warm calibration profiles, and forced full encrypted evaluations with diagnostic tracing. ResNet20 inherited the calibration state after AlexNet. The recorded gates required security of at least 128 bits, precision of at least 6 bits, and MAE at most 10^{-3}, rather than the 10^{-2} search threshold used for the three earlier rows of Table 7.

Table 5: Additional instrumented search outcomes. All ten evaluations recorded 128-bit security and a passing security check; Gate reports the combined numerical and security outcome. Boots is the archived structural count of ciphertext bootstraps in Orion's schedule, not the number of insertion sites. Times include diagnostic work.

Model	Trial	Time (s)	Prec. (bits)	MAE	Boots	Gate
AlexNet	000	296.6	−25.7	5.28×10^{7}	6	Fail
	001	269.5	−35.6	5.22×10^{10}	6	Fail
	002	258.8	−40.3	1.34×10^{12}	6	Fail
	003	374.7	−34.8	3.00×10^{10}	15	Fail
	004	378.5	−40.5	1.61×10^{12}	15	Fail
ResNet20	000	900.8	13.7	7.36×10^{-5}	76	Pass
	001	1007.0	13.7	7.36×10^{-5}	76	Pass
	002	926.5	−24.4	2.20×10^{7}	76	Fail
	003	924.1	13.7	7.36×10^{-5}	76	Pass
	004	924.9	13.7	7.36×10^{-5}	76	Pass

These searches ran under full diagnostic tracing at an intermediate source revision and do not supply the best results in the main text. No AlexNet proposal passed the gate (0/5). ResNet20 passed 4/5 but did not improve on its initial 900.8 s configuration. In both cases, failed or slower proposals did not replace the verified best.

Pre-measurement candidate record. In the first ResNet20 refinement pool, predictions and the LLM selection were saved before measurement; only one candidate was feasible, and the LLM selected it, so runtime ranking cannot be assessed from this pool.

AlexNet follow-up search. Table 6 reports a completed five-evaluation run with a subsequent source revision, using the same numerical thresholds and diagnostic timing convention as the preceding additional searches. Its initialization was generated and selected by the LLM, not loaded from a historical best. Subsequent candidates were proposed and selected through the regular feedback loop.

Table 6: AlexNet follow-up search evaluations. All five recorded a passing 128-bit parameter-security check; four also passed precision ≥ 6 and MAE $\leq 10^{-3}$. Times include diagnostic work.

Trial	Time (s)	Precision (bits)	MAE	Gate
000	267.7	21.7	2.94×10^{-7}	Pass
001	305.5	21.7	2.93×10^{-7}	Pass
002	270.5	4.2	5.37×10^{-2}	Fail
003	280.4	21.7	2.89×10^{-7}	Pass
004	487.6	21.7	2.87×10^{-7}	Pass

The initial configuration remained best. The failed trial reduced bootstrap auxiliary-prime widths from 61 to 55 bits; the offline precision estimate did not identify its numerical failure. Other trials changed the auxiliary chain length, local packing, or the global packing and modulus chain. These observations demonstrate feasible execution and best preservation, but not an improvement from refinement in this run.

Local approximation diagnostic. In a separate retrospective diagnostic, two completed ResNet20 evaluations with range margins 2 and 3 are compared at the first composite ReLU. Using the fixed model and plaintext input, we evaluate the backend's degree-$(15, 15, 27)$ polynomial composition with each margin's scaling. The resulting MAEs against exact ReLU are 3.9171×10^{-5} and 1.0059×10^{-4}; the corresponding decoded FHE endpoint errors are 3.9197×10^{-5} and 1.0062×10^{-4}. The coefficients are generated without keys or encrypted computation; one coefficient set takes 3.35 s on one CPU thread and is reused for both margins. Neither measured error is used to compute the plaintext estimate. This explains the observed local worsening through approximation rather than assuming that a wider interval improves accuracy. The diagnostic is evaluated separately from online candidate prediction and does not estimate final-network ciphertext error.

E Extended Searches and Same-LLM Comparison

Sources. The extended Gemma AlexNet and ResNet20 searches, the AlexNet follow-up search, and the LeNet comparison (Table 2) each use a fixed source revision; all five LeNet methods share one revision. These are separate from the three earlier search revisions in Appendix C. Model, input, weights, executed configurations, gate summaries, and LLM request/response records are retained with each run. For the extended AlexNet search, we report the seven evaluations completed at submission.

Table 7: Gemma searches behind the reported results (two further diagnostic-traced searches are in Appendix D). Pass counts use each search's recorded MAE gate (10^{-2} for the earlier searches, 10^{-3} otherwise); every displayed best configuration meets both. First is the first feasible framework output of that search. The extended ResNet20 row includes three validations of framework-proposed initial candidates. Bits and MAE describe the best. Table 1 reports the fastest configuration per workload.

Workload	Search	FHE (pass)	First (s)	Best (s)	Bits	MAE
LeNet	Earlier	10 (10)	17.17	14.56	10.4	7.52×10^{-4}
AlexNet	Earlier	5 (5)	333.98	288.36	17.7	4.58×10^{-6}
ResNet20	Earlier	5 (5)	796.97	762.03	14.0	6.30×10^{-5}
AlexNet	Follow-up	5 (4)	267.66	267.66	21.7	2.94×10^{-7}
AlexNet	Extended, first seven	7 (7)	306.24	283.92	21.2	4.09×10^{-7}
ResNet20	Extended + initial candidates	13 (13)	687.59	604.79	13.9	6.51×10^{-5}

The ResNet20 final selection combines ten online evaluations with three encrypted measurements of the framework's initial candidates. All thirteen pass security support, precision, and MAE gates. There are twelve distinct effective configurations: one initial configuration is measured both online and as an initial candidate. The online minimum is 682.150227 s, while the final minimum is 604.785352 s. The initial candidates are generated by the framework, and their forecasts are saved

before these measurements. The winning candidate's saved configuration is the final selected configuration; a later measurement is not retroactively inserted into the online decision history.

Selected configurations. The AlexNet follow-up result uses $\log N = 16$, scale 40, $H = 192$, $Q = (55, 40^{\times 10})$, application $P = (61^{\times 3})$, and explicit bootstrap $P = (61^{\times 4})$. It uses hybrid embedding, margin 2, and no layer override. The extended AlexNet sixth and seventh evaluations use scale 36, $H = 32768$, $Q = (60, 36^{\times 10})$, application $P = (61^{\times 4})$ also inherited by bootstrapping, hybrid embedding, margin 2, and BSGS ratio 4 on `features.4.conv.0`, `classifier.0.linear.0`, `classifier.1.linear.0`, and `classifier.2`. The seventh adds BSGS ratio 4 on `features.2.conv.0`, giving 283.915412 s, 21.2231 bits, and MAE 4.08513×10^{-7}. The ResNet20 final configuration uses $\log N = 16$, scale 40, $H = 192$, $Q = (55, 40^{\times 9})$, application $P = (61^{\times 3})$, explicit bootstrap $P = (61^{\times 5})$, hybrid embedding, margin 2, and no local override.

Table 8: Configuration choices associated with the measured results. All listed configurations use $\log N = 16$. Earlier rows describe joint initial-to-best changes; later rows describe the final selected settings.

Result	Configuration changes or selected settings
Earlier LeNet	Scale $40 \to 26$; Q chain $12 \to 8$ primes; P widths $61 \to 55$ bits; four local BSGS ratios set to 4.
Earlier AlexNet	Scale/interior Q widths $40 \to 32$; $H : 192 \to 32768$; application P count $3 \to 4$, inherited by bootstrap P.
Earlier ResNet20	Scale/interior Q widths $40 \to 38$; leading Q width $55 \to 53$; H, application P, and explicit bootstrap P unchanged.
AlexNet, 267.66 s	Scale 40, $H = 192$, $Q = (55, 40^{\times 10})$, application $P = (61^{\times 3})$, explicit bootstrap $P = (61^{\times 4})$.
ResNet20, 604.79 s	Scale 40, $H = 192$, $Q = (55, 40^{\times 9})$, application $P = (61^{\times 3})$, explicit bootstrap $P = (61^{\times 5})$.

Table 9 gives the exact operation-level values behind Figure 3. Two single-setting comparisons in the earlier searches show component trade-offs: raising AlexNet's main P from three to four primes (trial 2→1, bootstrap P inherited) changes bootstrap time by -20.89 s and linear-transform time by $+12.63$ s, and global square embedding in ResNet20 (trial 0→1) removes standalone rotations but adds 44.34 s of linear-transform time.

Table 9: First-to-best measured time reductions (s), decomposed by operation. Negative values indicate increased time. Remaining is total minus the three listed components, including other operations and diagnostic overhead.

Search	Total	Bootstrap	Linear transform	Polynomial	Remaining
LeNet, earlier	2.61	0.00	2.15	0.00	0.46
AlexNet, earlier	45.62	35.58	9.53	0.30	0.22
ResNet20, earlier	34.94	28.15	2.14	0.99	3.67
ResNet20, extended	82.81	80.97	−8.65	2.49	7.99

LeNet comparison protocol. Each method executes three FHE trials for each LLM, except the Gemma general agent. Astra uses high reasoning effort; Luna uses xhigh. Framework roles are separate calls to the same model, with no tool access. The general-purpose agent instead runs one non-interactive session and invokes backend commands itself. Both use the same fixed LeNet weights and a single input tensor of shape $1 \times 1 \times 28 \times 28$.

Framework runs do not use template-store initialization or saved calibration. They still receive four reference-only examples (LeNet, LoLa, MLP, and ResNet); the general-purpose agent receives one LeNet reference and is prohibited from reading historical configurations or the other method's results. The framework supplies structured candidate analyses and history. These information differences are part of the method comparison, not controlled away as an identical-information ablation.

Both framework runners finish normally after three records. All six trials report backend security support of at least 128 bits, precision above 6 bits, MAE below 10^{-3}, and successful encrypted inference with zero bootstraps. Noise margin and peak memory are not measured in these runs.

Table 10: Complete LeNet framework trials. Each arm keeps its global settings fixed and changes only Conv2 BSGS. Initial effective ratio is 2. Rotation counts are compiler-derived for Conv2, not counted primitive events; Conv2 times are measured module timings.

LLM	Trial / ratio	Forward (s)	Conv2 (s)	Rotations	Bits	MAE
Astra	0 / 2	16.778	7.155	126	23.2246	1.0203×10^{-7}
	1 / 4	16.656	6.090	114	23.2226	1.0217×10^{-7}
	2 / 8	17.953	7.245	114	23.2234	1.0211×10^{-7}
Luna	0 / 2	16.465	6.784	126	23.2209	1.0228×10^{-7}
	1 / 4	17.606	7.190	114	23.2214	1.0225×10^{-7}
	2 / 1	16.528	7.541	126	23.2328	1.0144×10^{-7}

Table 10 supplies the module measurements behind Figure 4. Astra uses $\log N = 16$, scale 40, $H = 192$, $Q = (60, 40^{\times 7})$, $P = (60^{\times 3})$, hybrid embedding, and margin 1; Luna uses $Q = (55, 40^{\times 7}, 60)$, $P = (61^{\times 3})$, and margin 2, with the other global fields the same. Astra's selected Conv2 ratio is 4; Luna retains the no-override initialization. Astra's general-purpose-agent best instead uses scale 30, $Q = (50, 30^{\times 7})$, $P = (61^{\times 3})$, margin 2, and square embedding/BSGS ratio 4 on both convolutions. Luna's corresponding best uses scale 40, $Q = (55, 40^{\times 10}, 60)$, $P = (61^{\times 3})$, margin 2, and square embedding on both convolutions. The differing final configurations are the objects being compared.

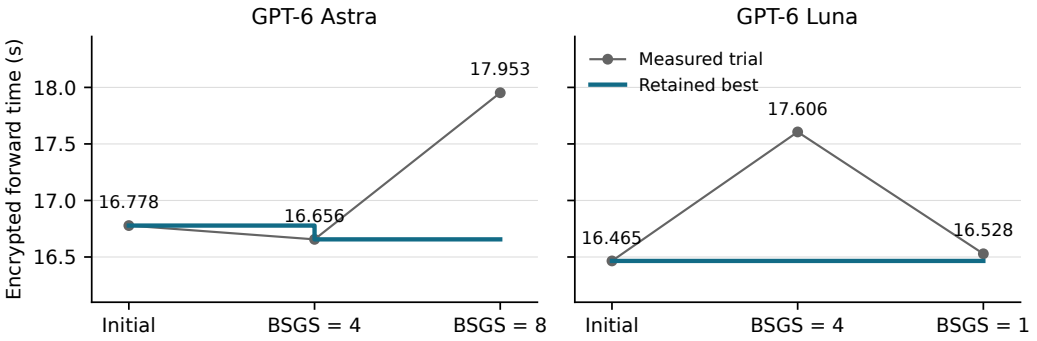

Figure 4: Layer-local LeNet exploration with fixed global parameters in each run. Points show measured encrypted forward times; the step line retains the verified best. "Initial" has no explicit local override; later trials change only Conv2 BSGS. All trials pass the measured gates.

Local and bootstrap evidence. The AlexNet Trials 1–5 share effective global parameters, seven bootstrap placements, and nine traced ciphertext-bootstrap calls. Their derived rotation totals are 1,207, 1,207, 1,400, 1,326, and 1,283; aggregate measured time of the three classifier linear modules is 31.547, 31.434, 49.543, 31.245, and 30.247 s. These module sums exclude intervening activation/bootstrap work. Within these first six evaluations, the sixth 288.578750 s trial improves on the 290.403022 s incumbent; 313.410574 s is a slower exploration point, not the comparison baseline. The seventh evaluation (283.92 s) is that search's best (Table 7).

For the two ten-prime ResNet20 initial candidates, `ckks_params`, embedding, margin, and local overrides are identical; only explicit bootstrap P changes from four to five 61-bit primes. Both captures contain 507 primitive events, including 38 bootstrap calls, with identical operation counts. Aggregate bootstrap times are 687.418098/485.115076 s, polynomial-operation times 82.838198/37.372215 s, and linear-transform times 36.882673/34.436757 s, respectively. These are individual synchronous backend calls within encrypted inference. Their MAEs are 6.511938×10^{-5} and 6.512932×10^{-5}. The bootstrap-time difference is 80.0% of the measured whole-network difference (857.65→604.79 s); it is not the sole changing time component.

The exact component values underlying Figure 7 are preserved in Table 11.

Table 11: Evidence available for configuration decisions. (a) Both errors compare with exact ReLU; the decoded endpoint also includes preceding FHE computation. (b) Planned calls and calls with compatible prior timing; time coverage is matched observed time divided by whole-network time.

(a) Local numerical diagnostics

Margin	Offline MAE	FHE MAE
2	3.9171×10^{-5}	3.9197×10^{-5}
3	1.0059×10^{-4}	1.0062×10^{-4}

(b) Runtime evidence coverage

Change	Calls	Timed	Coverage
Square	484	462	80.0%
Longer Q	577	0	0.0%
Local BSGS	578	516	7.4%
Scale/Q	578	0	0.0%

Selection scope in the local case studies. Global alternatives remain admissible in the inspected selection batches. The two LeNet runs propose thirteen post-initialization candidates, six global and seven local, but select local BSGS changes for all four subsequent FHE evaluations. In its first seven evaluations, AlexNet selects one joint global/local revision followed by five local-only revisions. These counts describe the allocation of measurements in these cases, rather than a restriction of the framework's configuration space or an assessment of global-search quality.

Prediction diagnostics. The three ResNet20 initial-candidate forecasts are 334.046, 351.326, and 351.326 s; measured times are 729.686, 604.785, and 857.654 s. The forecast order does not select the fastest candidate first. A diagnostic scores concordant pairs as 1, discordant pairs as 0, and low-confidence forecast ties as 0.5, yielding 0.5 over three pairs. This does not imply that the predictor claimed a reliable total order. Among the AlexNet initial candidates, three eligible measured times span only 1.71%; under the analysis's 5% near-tie rule there are no scorable actual directions. That rule is a diagnostic convention, not a confidence interval. A fourth AlexNet candidate runs in 266.002 s but lacks a complete security report, so it is not counted. Another candidate's predicted MAE 0.00174133 contrasts with measured MAE 4.77083×10^{-6}, the numerical false-negative example discussed in Section 4.4.

Earlier search trajectory. Figure 5 preserves the earlier five-evaluation ResNet20 case underlying the retrospective timing study. Its BSGS trial also changes effective bootstrap P, so it cannot isolate BSGS effects; the LeNet and extended AlexNet cases above provide local-only changes.

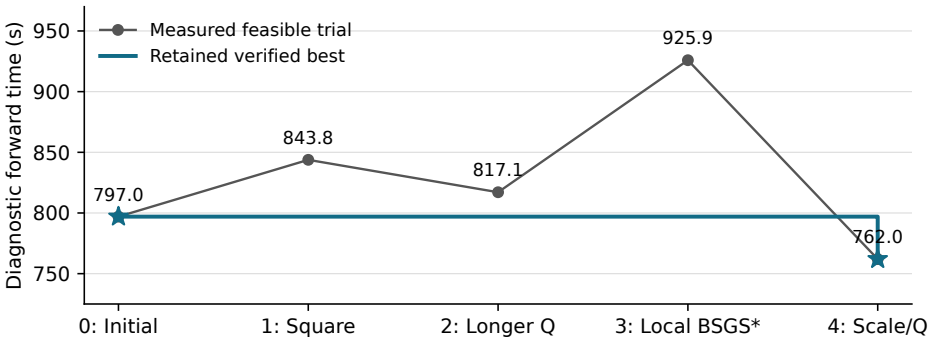

Figure 5: Earlier ResNet20 trajectory and preserved verified best. All five trials pass the recorded gates. The local-BSGS trial also changes bootstrap P from six primes to three; the last scale/Q revision improves the initial best.

Comparison cost scope. FHE-Agent uses 11 Astra and 8 Luna LLM calls, with aggregate call elapsed times 329.9 and 478.4 s. Whole-run times are 27m24s and 30m34s; the corresponding general-purpose sessions take 17m48s and 18m25s and contain 44 and 25 command-execution events. These wall times include setup, analysis, model calls, compilation, and inference. The

two iterative agents therefore have equal FHE counts but not equal total cost. Neither the LLM-call time nor compile/forward subtotals are added again to whole-run time. Measurements were collected under shared CPU conditions.

E.1 Decision records and timing diagnostics

Bootstrap-P decisions. The extended ResNet20 batches at iterations 2 and 6 both select a reduction from four to three bootstrap auxiliary primes. Their rationales expect lower bootstrap cost; the measurements instead rise from the respective 687.59/682.15 s incumbents to 785.01/806.87 s. The second batch has access to the earlier outcome but repeats the hypothesis. Feedback did not correct this hypothesis; the deterministic best-update rule retained the incumbent in both cases. Iteration 2 also contains an unselected shorter-chain/five-prime proposal. The final 604.79 s configuration comes from the validated initial candidates; similarity of this direction does not make the unselected later proposal an executed result.

Layer-local interpretation. The earlier LeNet comparison in Table 3(c) uses Trials 4 and 8, with $\log N = 16$, scale 26, $Q = (48, 26^{\times 6}, 48)$, $P = (55^{\times 3})$, $H = 192$, hybrid embedding, and margin 2. The proposal targets conv1/conv2 after a slower conv2 ratio-8 trial, anticipating reduced rotation work. The standalone rotate primitive count remains 11, while measured linear-transform time falls. This counter is distinct from compiler-derived rotation counts, including operations inside linear transforms; it does not identify their internal mechanism. The two MAEs are 7.48191×10^{-4} and 7.57043×10^{-4}.

The separate LeNet fixed-configuration study interleaves three measurements per configuration with identical backend build, model, input, and thread settings. Only conv1 BSGS changes from 2 to 4. Its times are 4.617/4.589/4.652 s versus 4.283/4.017/3.982 s; medians are 4.617/4.017 s. Whole-network times are 17.192/16.146/16.086 s versus 16.346/14.866/15.440 s. This is independent local evidence, not a repeated evaluation of the scale-26 configuration above.

In the later Astra pilot, Conv2 BSGS 4 changes the observed total from 16.778 to 16.656 s (0.73%), whereas Luna's initialization remains best. Table 10 retains all six trials. Neither the small total gap nor the differing Conv2 outcomes establishes stable network speedup. For extended AlexNet, the seventh evaluation changes one additional layer's BSGS ratio; that module takes 8.883 versus 8.827 s in the preceding trial. Its lower total time therefore cannot be attributed to a faster target layer. Figure 6 retains the six-point AlexNet record preceding the seventh evaluation and the ResNet20 initial-candidate decomposition.

Variation and additional LeNet pilots. Archived repeat diagnostics include identical recorded configurations across separate runs: AlexNet examples span 300.21–342.15 s (six measurements) and 288.36–297.58 s (two); a ResNet20 example spans 687.59–736.48 s (three). These groups span execution dates and conditions, and are not a common confidence interval or universal noise threshold. The four-prime ResNet20 initial-candidate capture also contains long bootstrap calls of approximately 89 and 77 s and a 45 s polynomial call; its aggregate gap is not solely a uniform per-operation shift.

The independent-generation and direct-feedback rows of Table 2 are three-attempt LeNet pilots without tools. Search-time single measurements below precede the re-timing in Table 12. Best accepted times are 17.51 s (Astra) and 16.36 s (Luna) for independent generation; direct feedback accepts no Astra candidate. Accepted/attempt counts are 1/3, 2/3, 0/3, and 2/3, respectively. One Astra feedback attempt fails backend initialization before FHE; other rejected attempts pass the numerical gates but have unknown security support (outside the checker's supported domain). The direct prompt uses one template and does not expose the full supported security-profile domain. Two independent Luna attempts generate the identical configuration, measured at 16.61 and 16.36 s. The best Luna direct-feedback configuration was re-timed five times in one compiled process, averaging 17.38 s, all passing the gates. The Gemma columns use `gemma-4-31b` with high reasoning effort under the same protocol: independent generation accepts 2/3 (best 20.61 s; one attempt has unknown security support), direct feedback 3/3 (best 18.13 s), and FHE-Agent 3/3 (best 17.11 s, precision 14.18 bits, MAE 5.38×10^{-5}). The Gemma general agent calls one terminal tool through the model API instead of a coding-agent session; its accepted configuration is identical to the Luna general-agent configuration.

Table 12: Best accepted LeNet configuration of each method and LLM in Table 2 (all $\log N = 16$, $H = 192$), re-timed five times in one compiled process with the same worker; compile time and peak memory are recorded once per configuration. Packing lists layer overrides (sq: square embedding; B: BSGS ratio); memory is peak resident memory. Identical configurations share a row's measurements.

LLM	Method	Scale	$\|Q\|$	Margin	Packing	Time (s)	Bits	MAE	Mem. (GB)	Compile (s)
Astra	Independent	40	10	2	c1:sq c2:sq	18.13	23.2	1.0×10^{-7}	50.2	242
Astra	General agent	30	8	2	c1:sq/B4 c2:sq/B4	15.68	12.3	2.0×10^{-4}	45.6	244
Astra	FHE-Agent	40	8	1	c2:B4	15.56	23.2	1.0×10^{-7}	38.3	163
Luna	Independent	40	12	2	c1:sq c2:sq	16.74	23.2	1.0×10^{-7}	47.4	243
Luna	Direct feedback	40	11	2	c1:sq c2:sq	17.38	23.2	1.0×10^{-7}	48.8	255
Luna	General agent	40	12	2	c1:sq c2:sq	16.74	23.2	1.0×10^{-7}	47.4	243
Luna	FHE-Agent	40	9	2	hybrid	16.88	23.2	1.0×10^{-7}	39.5	171
Gemma	Independent	40	10	2	c1:sq c2:sq	20.64	23.2	1.0×10^{-7}	49.2	239
Gemma	Direct feedback	40	9	2	c1:sq c2:sq	17.59	23.2	1.0×10^{-7}	43.8	240
Gemma	General agent	40	12	2	c1:sq c2:sq	16.74	23.2	1.0×10^{-7}	47.4	243
Gemma	FHE-Agent	32	9	3	hybrid	16.61	14.2	5.4×10^{-5}	41.0	173

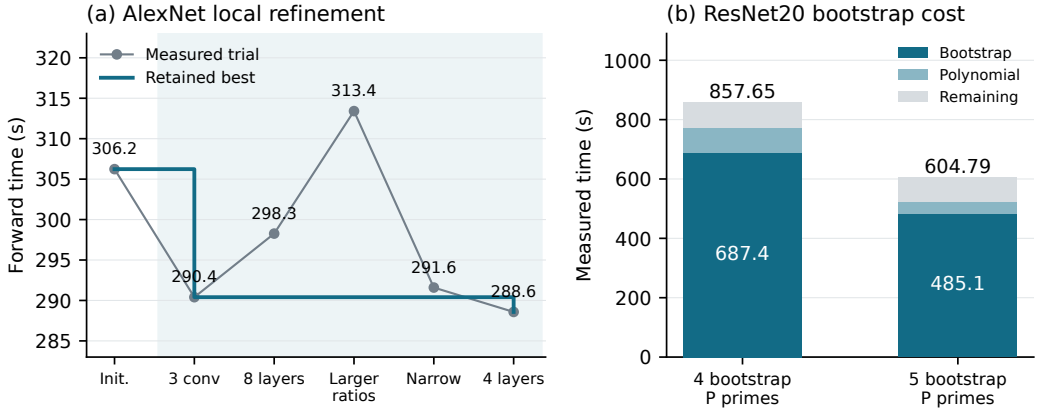

Figure 6: Configuration case studies. (a) The first six extended AlexNet evaluations and the best among those six (the seventh, 283.92 s, is in Table 7); the shaded region has fixed global parameters and different local BSGS overrides. (b) ResNet20 candidates differing only in bootstrap P. Both execute 38 bootstraps. Remaining time is whole-network time minus the two displayed operation sums, including other work and diagnostic overhead.

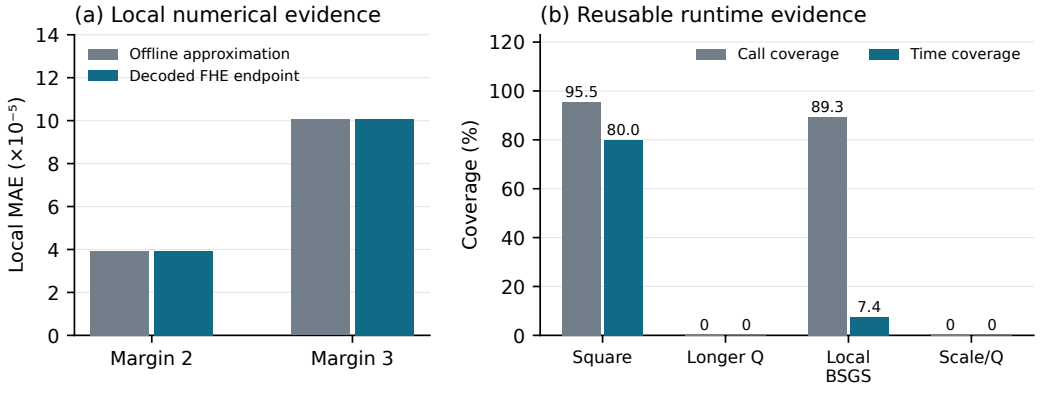

Figure 7: Evidence for configuration decisions. (a) Local approximation MAE and decoded FHE endpoint MAE track the effect of range margin; both compare against exact ReLU. (b) Fractions of planned calls and measured network time covered by compatible earlier timings. Local BSGS reuses many calls but little execution time. Exact values are in Table 11.

Table 13: Prototype comparison with independent one-shot LLM proposals on the same backend, under a fixed 128-bit security target; security entries are as reported.

Model	Method	Precision (bits)↑	MAE↓	FHE time [s]↓	Reported sec. [bits]↑
MLP	One-shot (best of 10)	17.37	5.89×10^{-6}	1.31	≥ 128
	FHE-AGENT	24.82	≈ 0	0.91	≥ 128
LeNet	One-shot (best of 10)	23.07	1.13×10^{-7}	9.08	≥ 128
	FHE-AGENT	22.42	1.79×10^{-7}	3.19	≥ 128
LoLA	One-shot (best of 10)	19.54	1.31×10^{-6}	2.10	≥ 128
	FHE-AGENT	21.16	≈ 0	0.79	≥ 128
AlexNet-style	One-shot (best of 10)	N/A	N/A	N/A	N/A
	FHE-AGENT	21.81	≈ 0	262.5	≥ 128

F HISTORICAL PROTOTYPE COMPARISON

The earlier prototype used a dual-socket AMD EPYC 9454 server (48 cores per socket, 192 hardware threads, 1.5 TiB RAM) and Orion with the Lattigo v5.0.2 CKKS backend. Orion and FHE-AGENT configurations ran single-threaded. The controller used an LLM to select optimization directions and deterministic tools for numerical analysis. The controller used Gemini 2.5 Flash through its public API; per-row decoding settings were not recorded. The historical study reports MLP, LeNet (LeCun et al., 1998), and LoLa variants on MNIST and an AlexNet-style variant on CIFAR-10, implemented with Orion (Ebel et al., 2025). The AlexNet-style variant follows the architectural family of AlexNet (Krizhevsky et al., 2012). The prototype report states that Orion reference configurations were used when available; these Orion adaptations are not specified by the model papers.

The earlier prototype reports MAE against plaintext, effective precision, encrypted runtime per input excluding key generation, and security. These historical results, including the prototype's effective-precision definition, security entries, and evaluation budgets, are retained as reported rather than re-derived with the MAE-derived precision of the instrumented searches.

One-shot results. For each model, the same LLM generated ten independent full Orion configurations, each checked by the prototype's tool suite; the best feasible proposal (or N/A) is reported in Table 13. FHE-AGENT has lower reported runtime for MLP, LeNet, and LoLa, with the shown precision/MAE tradeoffs; no feasible one-shot AlexNet proposal is reported. These results are not compared under matched budgets or against expert configurations, so they establish neither an advantage over expert configurations nor the search-efficiency benefit of feedback.

The reported MLP times decrease from 1.31 s to 0.91 s while effective precision rises from 17.37 to 24.82 bits. LeNet changes from 9.08 s to 3.19 s, a runtime ratio of about $2.85\times$, with slightly lower reported precision (23.07 to 22.42 bits) and higher MAE (1.13×10^{-7} to 1.79×10^{-7}). LoLa changes from 2.10 s to 0.79 s, about $2.66\times$, with reported precision increasing from 19.54 to 21.16 bits. For the AlexNet-style model, only the agent row has a reported feasible result, at 262.5 s; the N/A reference does not define a speed ratio. These comparisons concern the final configurations, not the cost of finding them.

G HISTORICAL LENET CASE STUDY

Table 14 reports the historical layer and network measurements. This case study uses Orion's MNIST LeNet example with a fixed CKKS parameter set ($\log N = 15$), a 256-bit security target, and the backend's activation polynomial family. Across trials, the CKKS parameters and total modulus size remain fixed; the controller changes layerwise packing and backend-exposed activation degrees.

The historical controller roles InitAgent and RegimeAgent use Orion's reference configuration to set the CKKS regime and activation family, then search packing and scale schedules within that regime. They select Trial 0 with reported MAE 3.0×10^{-4} and 11.63 bits of effective precision. The LayerProfiler reports 4.006 s for `conv2`; the table retains the other reported per-layer measurements.

Table 14: LeNet case-study measurements under the prototype's fixed CKKS regime ($\log N = 15$, 256-bit security target). Parentheses give each reported layer time divided by the displayed total runtime, recomputed to one decimal place.

	Trial (fixed CKKS security, $\log N = 15$)			
Global metrics	0	1	2	3
Total runtime [s]	7.89	6.25	5.04	8.51
MAE	3.0×10^{-4}	2.9×10^{-2}	2.9×10^{-2}	1.6×10^{-3}
Precision [bits]	11.63	5.12	5.12	9.27
# rotations	8	7	7	8
# multiplications	4	4	4	4
# bootstraps	0	0	0	0
Act. degree (conv1)	31	31	31	15
Layer	*Observed encrypted layer contribution [s (share of total)]*			
conv1	2.483 (31.5%)	2.561 (41.0%)	2.555 (50.7%)	2.516 (29.6%)
conv2	4.006 (50.8%)	2.794 (44.7%)	1.610 (31.9%)	4.304 (50.6%)
fc1	0.906 (11.5%)	0.548 (8.8%)	0.450 (8.9%)	1.016 (11.9%)
fc2	0.368 (4.7%)	0.298 (4.8%)	0.399 (7.9%)	0.584 (6.9%)

Layer and total times are as reported by the earlier prototype. Percentages use the displayed total runtime as a common denominator; they are descriptive ratios, not an exhaustive cost decomposition, and the four layer times do not sum to the full forward-pass time.

The first two refinements apply more aggressive packing to `conv2`. Trial 1 lowers its reported time from 4.006 s to 2.794 s and total runtime to 6.25 s, but its MAE rises to 2.9×10^{-2} and precision falls to 5.12 bits. Trial 2 reduces `conv2` to 1.610 s and total runtime to 5.04 s, with MAE and precision still 2.9×10^{-2} and 5.12 bits. Both are rejected for numerical quality. Figure 8 shows this speed–quality tradeoff.

In the earlier prototype, updated layerwise profiles and noise margins led the GlobalTradeoffAgent to consider `conv1`; the LayerwiseAgent then lowers its activation degree from 31 to 15 and sets `max_parallel_blocks`=2. Trial 3 therefore changes `conv1`, not the faster `conv2` override. Its reported MAE is 1.6×10^{-3} and precision 9.27 bits, at 8.51 s, compared with Trial 0's 7.89 s. Thus it is not a speed improvement over the first feasible initialization result. It is feasible under the prototype's gate but does not meet the 10^{-3} MAE criterion used in the main evaluation. Across the four reported trials, multiplications remain 4 and bootstraps 0, while rotations vary from 7 to 8. Updated layerwise profiles and noise margins inform the GlobalTradeoffAgent and LayerwiseAgent in Phase C; the StaticAnalyzer and LayerProfiler provide analysis, and the PatchGateAgent screens proposals before encrypted evaluation, with the stated goal of recovering precision without exhausting the noise budget. The prototype reports four verified encrypted trials, with remaining reasoning performed offline.

H Deployment and Configuration Details

H.1 Deployment and Security Assumptions

We consider private inference (Gilad-Bachrach et al., 2016) with a semi-honest server that holds a plaintext neural network. The client uses the public parameters and tensor-to-slot layout of the selected configuration to generate cryptographic keys and encode its input. It encrypts that input and supplies the evaluation keys needed for server-side computation. The server evaluates the compiled network using Orion/Lattigo (Ebel et al., 2025) and returns an encrypted prediction, which the client decrypts. This setting protects client inputs; it does not provide model confidentiality or verification against a malicious server, and does not address side-channel or traffic-analysis attacks.

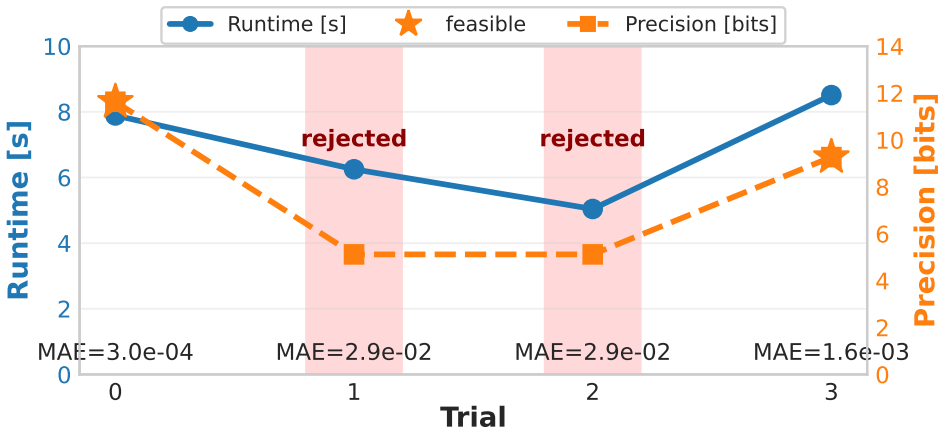

Figure 8: Historical LeNet refinement under a fixed CKKS regime. Trial 1 and Trial 2 reduce runtime but are rejected for numerical error; Trial 3 has higher runtime than Trial 0. Stars mark results feasible under the prototype's own gate; they do not indicate acceptance under the 10^{-3} MAE criterion used in the main evaluation.

Table 15: Configuration fields and derived quantities in the evaluated implementation. N is the polynomial-ring degree; Q is the ciphertext modulus and P is an auxiliary key-switch modulus. Lists of prime bit lengths are not operation counts.

Group	Examples	Interpretation
CKKS parameters	`LogN, LogScale, LogQ, LogP, H, RingType`	Explicit fields; when a security profile is selected, ring and secret parameters must agree with it.
Orion options	`embedding_method, margin`	Packing method and activation-range margin, subject to backend support.
Bootstrap parameter	`boot_params.LogP`	Auxiliary-prime setting for the bootstrap backend; not a placement list.
Layer-local options	`embedding_method, bsgs_ratio`	Overrides for eligible layers; the second controls baby-step/giant-step packing.
Derived execution facts	Levels, diagonals, rotations, bootstrap positions and slots	Computed by Orion for each effective configuration.

Configuration search takes place offline, before deployment, using public or authorized calibration inputs. This separation permits the developer to inspect intermediate numerical errors and execution profiles while choosing a configuration. Intermediate decryption belongs to that development process, not to server-side inference on private client queries. The LLM controller receives model descriptions and tool summaries; deployed clients' plaintext inputs are outside the search interface.

H.2 Configuration Fields

Here, `LogN`$= \log_2 N$, the encoding scale is $2^{\texttt{LogScale}}$, and `H` is the secret's Hamming weight. `LogQ`/`LogP` list prime bit lengths: Q provides ciphertext levels; P supports key switching. Security checking uses the relevant moduli and secret distribution, including bootstrap contexts, not the ciphertext chain alone.

Equal-runtime ties use ring dimension and total modulus bits as memory proxies, followed by precision and MAE. These tie-breakers do not change the primary measured-time objective.

H.3 Interpreting Layerwise Evidence

Layerwise evidence connects an expensive or numerically problematic computation to an executable modification. The controller can inspect offline structure, module/backend timings, and decoded diagnostic outputs from authorized offline evaluations. Error comparisons must use matching semantic outputs, and a module time must not be added to the times of operations already included in it. These conditions make a layer observation interpretable before it is used to propose an action.

The primary implementation supports layer-local embedding methods and baby-step/giant-step ratios, subject to the effective backend packing. For example, substantial rotation work can motivate a different ratio; analysis then exposes changes in rotation groups and the resulting execution plan. The requested override is checked against its effective value before the measured layer and network outcomes are interpreted. Global fields must also be compared: the ResNet20 case in Section 4.3 includes a local request alongside a change in effective bootstrap parameters. Historical activation-degree and parallelism changes belong to the prototype case in Appendix G, not to this current local action space.

This chain—observation, proposal, effective settings, plan, and measured outcome—distinguishes an unapplied request from an applied but unhelpful change. Layer information may suggest an action; establishing that the action causes an improvement requires controlled comparisons; the full loop preserves the verified best in either case.

Level consumption follows backend semantics through residual joins and bootstrap resets. Multiplication count is distinct from sequential depth; bootstrap sites, ciphertexts at a site, and native invocations are distinct quantities. Numerical diagnostics can expose approximation or scale-propagation problems without measuring ciphertext noise or final output error. Missing analysis is reported explicitly rather than imputed.

I Search Cost

Across the three earlier searches in Table 7, including initialization, compilation totals 1,382.8, 6,835.2, and 3,552.7 s for LeNet, AlexNet, and ResNet20; diagnostic forward passes total 160.9, 1,504.4, and 4,145.8 s. This partial accounting excludes proposal analysis and API time. Whole-search peak process memory is 50.4, 325.4, and 206.8 GB (decimal), respectively, including compilation and search state. These are not inference-only memory measurements or sums of concurrent processes. Because evaluation counts differ and proposal/API time is not included, these totals are not an equal-budget efficiency comparison.

J Extended Background and Related Work

CKKS levels and numerical cost. CryptoNets (Gilad-Bachrach et al., 2016) demonstrated inference on encrypted data. In CKKS, a larger encoding scale can preserve more numerical information but requires sufficient modulus capacity; increasing Q can change the minimum ring dimension required for a security target. In a residue number system representation, Q comprises a chain of primes, and the ciphertext level identifies the active prefix. Multiplication changes scale, while rescaling divides by a modulus factor and consumes a level. Multiplicative depth counts sequential dependencies rather than total operations; independent operations can share a depth. Bootstrapping (Bossuat et al., 2021; Chen et al., 2019) refreshes ciphertexts at a runtime and approximation cost. A longer chain can reduce bootstrap frequency while increasing individual operation costs.

Layout and scheduling methods. Rotom (Chen et al., 2026) explores layout assignments. DaCapo (Cheon et al., 2024) uses ciphertext liveness and latency estimates to place bootstraps; ReSBM (Liu et al., 2025) uses region decomposition and min-cut optimization for scale and bootstrap management; Orbit (Zhou et al., 2026) formulates their joint placement as an integer linear program. Orion (Ebel et al., 2025) uses shortest-path optimization over level choices, including residual subgraphs, to optimize linear-operation and bootstrap costs jointly. FHE-Agent retains this scheduling responsibility in the backend. Its agents compare the consequences of exposed configuration choices rather than prescribe bootstrap insertion sites.

Inference models, packing, and protocols. Beyond early encrypted neural-network inference, CryptoDL (Hesamifard et al., 2018) studied training and classification over encrypted data. Packing methods expose structural differences in execution cost: HeLayers (Aharoni et al., 2023) represents tensors through tile tensors, multiplexed parallel convolutions (Lee et al., 2022) compact sparse channel outputs to reduce bootstrap work, and SpENCNN (Ran et al., 2023) jointly designs packing and model sparsity. Orion (Ebel et al., 2025) uses diagonal encoding and baby-step/giant-step evaluation to reuse rotations across groups.

Other systems extend model or protocol choices. THE-X (Chen et al., 2022) approximates non-polynomial Transformer operations, while Powerformer (Park et al., 2025) combines distillation, function approximation, and homomorphic matrix multiplication. GAZELLE (Juvekar et al., 2018) combines HE linear-algebra kernels with garbled circuits; BumbleBee (Lu et al., 2025) targets secure two-party Transformer inference. GuardML (Frimpong et al., 2024) combines symmetric cryptography with HE to reduce the burden on end devices. Their model adaptation, interaction, and execution assumptions differ from the fixed-network CKKS configuration problem considered here.

Libraries and compilation. SEAL (Chen et al., 2017), OpenFHE (Al Badawi et al., 2022), and Lattigo (Lattigo) implement homomorphic operations; Cheddar (Choi et al., 2026) optimizes execution for GPU architectures. Primitive costs depend on the parameters, implementation, and hardware, making the execution backend part of configuration assessment. Above these libraries, EVA (Dathathri et al., 2020) provides an encrypted-vector intermediate representation and optimizing compiler; HECO (Viand et al., 2023) lowers imperative programs through FHE-specific optimizations; Porcupine (Cowan et al., 2021) synthesizes verified vectorized kernels. For numerical management, ELASM (Lee et al., 2023) searches scale-management plans using output-error and latency estimates, while HALO (Cheon et al., 2025) handles programs with loops. These systems complement the layout and scheduling methods discussed in Section 2.

Parameter selection and profiling. AutoPrivacy (Lou et al., 2020) applies reinforcement learning to layerwise HE parameter selection within a hybrid HE and garbled-circuit protocol. Fuzzy-logic and linear-programming methods (Cabrero-Holgueras & Pastrana, 2023) translate security, precision, and performance priorities into parameter choices. CipherAPR (Wan et al., 2026) combines importance-guided polynomial design, restored-level utilization, and latency and accuracy estimates to screen candidates. HE-DAP (Park et al., 2026) profiles the execution environment to choose polynomial and iteration parameters for encrypted inverse-square-root computation. These methods provide automated selection and measured cost guidance for distinct optimization tasks.

Learned optimization and feedback. CHEHAB RL (Sefsaf et al., 2026) learns rewrite sequences that trade instruction latency against noise growth, evaluated on the integer-arithmetic BFV scheme. Its LLM generates training expressions, while a reinforcement-learning policy performs optimization. In FHE-AGENT, the LLM proposes and selects configurations at decision time using tool evidence.

Beyond FHE, ACCLAIM (Mikek et al., 2026) coordinates compiler tools and LLM agents across source code, intermediate representation, and assembly, explicitly allocating optimization effort across levels. A controlled study of hardware-aware LLM optimization (Redko et al., 2026) finds that feedback-driven improvement depends on representation familiarity and prior knowledge, and can fail in some settings. These findings motivate evaluating tool-guided decisions under stated search budgets. Making proposals, analyses, encrypted evaluations, and selected configurations explicit supports that evaluation without assuming that feedback alone improves search.