ATLAS: Automated Approximation of Transformers for Efficient Homomorphic Inference in One Hour

Jianhang Xie
City University of Hong Kong
Hong Kong, China
jianhang.xie@my.cityu.edu.hk

Sicheng Tan
Shandong University
Qingdao, China
sqtsc@mail.sdu.edu.cn

Vishnu Naresh Boddeti
Michigan State University
East Lansing, MI, USA
vishnu@msu.edu

Zhichao Lu
City University of Hong Kong
Hong Kong, China
zhichao.lu@cityu.edu.hk

Abstract

Fully homomorphic encryption (FHE) provides strong cryptographic guarantees for private inference, but deploying transformer models under FHE remains prohibitively expensive. A key bottleneck is that non-linear operations such as softmax, normalization, and activation must be replaced with polynomial approximations compatible with the CKKS scheme, and the multiplicative depth consumed by these approximations dominates inference cost. Recent frameworks have advanced approximation techniques, yet all rely on manually configured approximation hyperparameters (e.g., number of iterations, polynomial degree), applied uniformly across all layers. While convenient, this uniform-configuration approach is overly rigid: different layers can tolerate different levels of approximation error without degrading predictive accuracy, and uniform configurations cannot exploit this variability to reduce latency. Allowing each layer to adopt its own configuration, however, causes the search space to explode with model depth, reaching roughly 10^{84} configurations for BERT/ViT (12 layers) and 10^{225} for LLaMA3 (32 layers), rendering manual exploration practically impossible.

We present ATLAS, an automated framework that configures per-layer approximation settings by formulating the problem as a multi-objective optimization over latency and predictive accuracy. The resulting problem is inherently difficult: 1) competing objectives over a large decision space (120 or 320 variables for BERT/ViT or LLaMA3); 2) expensive evaluation, as each configuration takes 70–1,000 seconds even in cleartext; and 3) sparse optimization signals, as 35–50% of candidate configurations yield numerically invalid solutions. ATLAS addresses these challenges through a two-stage optimization strategy that progressively relaxes layer-wise constraints, combined with surrogate models to accelerate evaluation. Compared to iterative softmax (Cho et al., CCS 2024) and THOR (Moon et al., CCS 2025), ATLAS reduces multiplicative depth by ~35% and ~17% with negligible accuracy loss, translating to ~25% and ~20% reduction in end-to-end inference latency, respectively, while completing the entire configuration search in one hour. As a post-processing step compatible with existing approximation advancements, ATLAS generalizes across encoder-only (BERT), decoder-only (LLaMA3-8B), and vision (ViT) transformers, and complements parallel work on packing and matrix multiplication to substantially lower the barrier to deploying cryptographically secure inference. Code: https://github.com/jianhayes/ATLAS

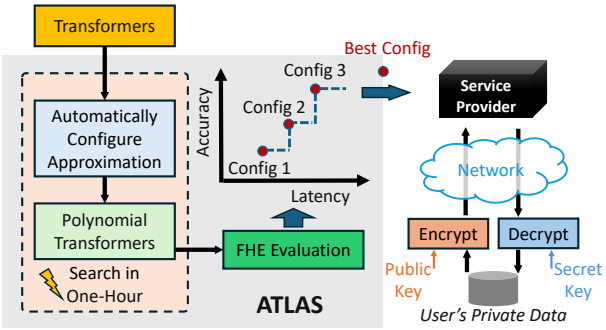


Figure 1: ATLAS can automatically design polynomial approximation configurations for FHE compatible transformers. ATLAS solutions span the trade-off between accuracy and latency for ciphertext inference and can be deployed on the cloud server to satisfy a range of customer requirements.

Keywords

Fully Homomorphic Encryption, CKKS Scheme, Transformers, Secure Inference, Multi-Objective Optimization

1 Introduction

The widespread deployment of machine learning has created a growing tension between model utility and data privacy. In sensitive applications such as medical diagnosis, financial assessment, and legal document analysis, a client wishes to obtain predictions from a server-hosted model without revealing the underlying input data. Fully homomorphic encryption offers a compelling cryptographic solution to this problem: as shown on the right side of Figure 1, under an honest-but-curious server assumption, FHE allows the server to evaluate the model entirely on encrypted data, providing strong privacy guarantees without requiring the server's trust beyond adherence to the protocol.

Transformer architectures have become the dominant paradigm for such tasks, spanning language understanding (BERT [22], LLaMA [48]) and vision (ViT [24]). Deploying transformers under FHE, however, remains extremely expensive. The Cheon-Kim-Kim-Song (CKKS) scheme [10, 11, 14], which supports approximate arithmetic on real-valued data, is the most practical choice for neural network

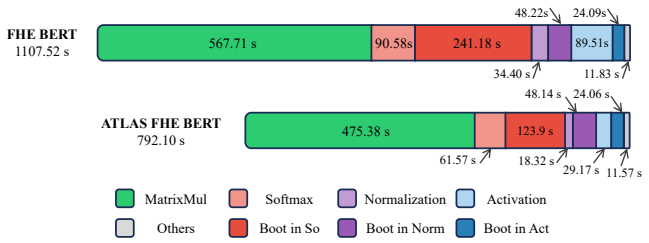


Figure 2: Runtime breakdown of end-to-end FHE BERT [22] on an RTX 4090. ATLAS achieves a 28.5% speedup, and saves 29.01 s, 16.08 s, and 60.34 s in Softmax, normalization, and activation approximations, respectively, which further reduces bootstrapping overhead by ~117 s.

inference, but it imposes a fundamental constraint: operations are limited to polynomial arithmetic, and the number of sequential multiplications (the multiplicative depth) determines both the ciphertext parameter size and the overall latency. As shown in Figure 2, a baseline FHE implementation of a transformer devotes approximately 46% of inference time to non-linear operations alone. This overhead arises because non-linear functions such as softmax, layer normalization, and GELU are incompatible with FHE and must be replaced with polynomial approximations, each consuming significant multiplicative depth.

A substantial body of recent work has sought to reduce this cost. One line of work targets the linear components of transformers, particularly ciphertext-ciphertext matrix multiplication (CCMM) and plaintext-ciphertext matrix multiplication (PCMM) [28, 35, 38]. A second line of work addresses approximations of non-linear operations such as Softmax, LayerNorm, and GELU directly, either through polynomial approximations or through iterative computations [16, 41, 54]. A third line of work seeks to improve the computational efficiency of core cryptographic primitives, including bootstrapping [3, 8, 9, 12, 17] and polynomial multiplication [31].

Despite this progress, a critical design decision remains unaddressed across all existing frameworks: how to set the hyperparameters governing each polynomial approximation. Current practice treats this as a manual, per-function problem. A practitioner selects an approximation for each non-linear function type (e.g., a degree d minimax polynomial for GELU) and applies it uniformly across all layers. This approach is intuitive but suboptimal. The true objective is not to approximate individual functions accurately in isolation, but to preserve the end-to-end task accuracy of the pretrained transformer model under the constraints of FHE. Optimizing for this global objective opens the possibility of heterogeneous, per-layer configurations in which different layers use different approximation configuration hyperparameters, trading local approximation fidelity for global depth reduction. The challenge is that this search space is vast. With multiple tunable hyperparameters per operator and dozens of layers per model, the space of valid configurations spans 10^{84} to 10^{225} possibilities depending on the architecture.

We present ATLAS, a framework-agnostic post-processing method that automatically discovers depth-efficient approximation configurations for any FHE-compatible transformer. ATLAS formulates the problem as multi-objective optimization over multiplicative depth and end-to-end accuracy, and solves it through a two-stage evolutionary search. ATLAS treats any existing FHE transformer framework as a black box, operating entirely post-training on the frozen pretrained model and requiring no retraining, fine-tuning, or changes to its packing strategy or encryption parameters. An overview of ATLAS is shown in Figure 1. The contributions of this paper are:

(1) **Problem formulation.** We identify the selection of per-layer polynomial approximation hyperparameters as a critical and underexplored bottleneck in FHE transformer inference, and formulate it as a multi-objective combinatorial optimization problem over multiplicative depth and end-to-end accuracy.
(2) **ATLAS.** We present a framework-agnostic post-processing method that solves this problem through a two-stage evolutionary search, designed to handle conflicting objectives, expensive evaluations, and a high density of invalid configurations. ATLAS is complementary to existing frameworks, including THOR [38], NEXUS [54], and MOAI [55], and can be applied on top of any FHE-compatible transformer without modifying the underlying encryption or packing strategy.
(3) **Empirical evaluation.** Evaluated on BERT-base, LLaMA-3 8B, and ViT-base, ATLAS reduces multiplicative depth by 35 percent and inference latency by 25 percent relative to expert-designed baselines, with negligible accuracy loss. The search completes in under 30 minutes for BERT and ViT, and under one hour for LLaMA-3 8B.

2 Preliminaries

2.1 FHE Scheme and Bootstrapping

CKKS Scheme. The Cheon-Kim-Kim-Song (CKKS) [14] is a leveled homomorphic encryption scheme. Compared with previous FHE schemes, e.g., BGV [6], BFV [5], and TFHE [15], which only support integer number encryption, the CKKS can encrypt real and complex number calculations.

Under RNS-CKKS, the plaintexts and ciphertexts are elements in a residue cyclotomic polynomial ring $\mathcal{R}_{Q_\ell} = \mathbb{Z}_{Q_\ell}[X]/(X^N+1)$. The modulus is $Q_\ell = \prod_{i=0}^{\ell} q_i$, where $0 \le \ell \le D$, the ℓ is *level*, the D is level budget. N is the polynomial degree in RNS-CKKS, and a ciphertext has $N/2$ slot counts for single instruction multiple data (SIMD) processing. We denote the above procedure from cleartext u to SIMD ciphertext $\mathrm{Enc}(u)$ as $\mathrm{Enc}(\cdot)$, and the opposite direction decryption-decoding algorithm is denoted as $\mathrm{Dec}(\cdot)$. Specifically, assuming that homomorphic addition is $\boxplus$ and homomorphic multiplication is $\boxtimes$, the operations can be described in the following with u and v as:

$$\begin{aligned} \mathrm{Dec}(\mathrm{Enc}(u) \boxplus \mathrm{Enc}(v)) &= \mathrm{Dec}(\mathrm{Enc}(u)) + \mathrm{Dec}(\mathrm{Enc}(v)) \approx u + v \\ \mathrm{Dec}(\mathrm{Enc}(u) \boxtimes \mathrm{Enc}(v)) &= \mathrm{Dec}(\mathrm{Enc}(u)) \times \mathrm{Dec}(\mathrm{Enc}(v)) \approx u \times v \end{aligned} \tag{1}$$

Bootstrapping. Leveled homomorphic encryption schemes like CKKS only support a finite number of homomorphic multiplications, each of which consumes one level due to rescaling. So, when the level of a ciphertext becomes zero, the decryption would fail, so we need to perform a *bootstrapping* [10] operation to reset to a high level if it is too low to do a computation. The bootstrapping allows

us to evaluate circuit of arbitrary depth, as it homomorphically evaluates the decryption circuit and raises the modulus from Q_0 to Q_D by leveraging the isomorphism $\mathcal{R}_{q_0} \cong \mathcal{R}_{q_0} \times \mathcal{R}_{q_1} \times \cdots \times \mathcal{R}_{q_D}$ [4]. A freshly encrypted ciphertext starts with D levels, but bootstrapping consumes K levels and reduces it to $D-K$ levels. As bootstrapping requires a lot of key switching operations (KSO), it becomes the most time-consuming operation in RNS-CKKS.

2.2 Transformers

Transformer [49] models are typically encoder-only (BERT [22]), decoder-only (e.g., GPT [43] and LLaMA [48]), or vision encoder (ViT [24]). Despite this distinction, the structure of the Transformer layer is similar in both encoders and decoders: a stack of L layers. A basic Transformer layer $f(\boldsymbol{x})$, includes Attention, Multilayer Perceptron (MLP), and LayerNorm, as shown in Equation (2).

$$f(\boldsymbol{x}) = \text{LayerNorm2}(\text{MLP}(\text{LayerNorm1}(\text{Attention}(\boldsymbol{x})))), \quad (2)$$

where $\boldsymbol{x} \in \mathbb{R}^{n \times d}$ is the input embedding, n is the number of tokens and d is the hidden size.

For attention, the weight matrices are $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_O \in \mathbb{R}^{d \times d}$, yielding $\boldsymbol{Q}=\boldsymbol{x}\boldsymbol{W}_Q$, $\boldsymbol{K}=\boldsymbol{x}\boldsymbol{W}_K$, and $\boldsymbol{V}=\boldsymbol{x}\boldsymbol{W}_V$. For standard multi-head attention (MHA) with h heads, $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are partitioned into $\boldsymbol{Q}_h, \boldsymbol{K}_h, \boldsymbol{V}_h \in \mathbb{R}^{n \times d_k}$, where $d_k=d/h$ is the head dimension. The attention computation for a single head is shown in Equation (3).

$$\text{Attention}(\boldsymbol{Q}_h, \boldsymbol{K}_h, \boldsymbol{V}_h) = \text{Softmax}\left(\frac{\boldsymbol{Q}_h \boldsymbol{K}_h^T}{\sqrt{d_k}}\right)\boldsymbol{V}_h \quad (3)$$

Finally, the outputs of all heads are concatenated and projected via $\boldsymbol{W}_O$ to produce the final attention result.

For the MLP, let d_{ff} be the intermediate size. The MLP weights consist of two matrices $\boldsymbol{W}_{\text{up}} \in \mathbb{R}^{d \times d_{\text{ff}}}$, $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{d_{\text{ff}} \times d}$, with an activation $\sigma(\cdot)$ placed between them. Typically $\sigma(x)$=GELU(x). For the LLaMA MLP, $\sigma(x)$=SiLU(x) and it adopts a gated structure.

For normalization, standard Transformers use LayerNorm [2]. Modern LLMs such as LLaMA employ RMSNorm [53] to reduce computational cost. The placement of normalization depends on the specific architecture.

2.3 FHE Transformers

Since homomorphic matrix multiplication is nearly lossless in precision, the precision loss arises from the approximation of non-linear components. Thus, constructing FHE inference requires a primary focus on these non-linearities. For Transformers, the non-linear components include *Softmax* in Attention, *normalization*, and *activation* in the MLP. Compared to CNNs [1, 32] where only ReLU requires approximation, building a Transformer inference system under FHE is considerably more challenging. Currently, the approximation of these components [16, 38, 44, 52, 54, 55] relies on expert-designed heuristics and manually configured hyperparameters.

Softmax Approximation. The standard softmax function is defined as Softmax$(\boldsymbol{x})_i$=$\exp(x_i)/\sum_j \exp(x_j)$, which contains the non-linear exponential $\exp(x)$ and inverse $1/x$. Some works replace the Attention with HE-friendly alternatives, such as Gaussian-kernel Attention [44], activation-based Attention [56], and BPMax [40], but these replacements all require model retraining.

Recent works [52, 54, 55] directly approximate $\exp(x)$ and $1/x$ separately over a given interval. For example, $\exp(x)$ is approximated via limit $(1+x/2^r)^{2^r}$ [54, 55] for $x \in [-2^r, 0]$, or via a Chebyshev polynomial [52]; and $1/x$ via Goldschmidt iteration [54, 55] or a Chebyshev-initialized Goldschmidt refinement [52].

To improve numerical stability, the row maximum is subtracted: Softmax$(\boldsymbol{x})$=Softmax$(\boldsymbol{x}-\max_i x_i)$. In FHE, dynamically computing $\max_i x_i$ is prohibitively expensive; thus, a statistical hard-coded constant maximum c is used instead.

Although this separate approximation incurs low multiplicative depth, the effective domains of the $1/x$ and $\exp(x)$ approximations jointly constrain the softmax input range. From the NEXUS [47] codebase, we measure the usable softmax interval to be roughly $[-2.19, 0]$. As a result, the valid input range may be narrower than the range of $x_i - c$, leading to approximation failure.

A state-of-the-art alternative employs an *iterative softmax* [16] with normalize-and-square strategy. It assumes that inputs after subtracting the maximum are non-positive, i.e., $\boldsymbol{x} \in [-M, 0]^n$. The input is scaled by $1/2^k$ and recovered via k iterations. Consequently, the effective input range of the iterative softmax can be estimated as $[-2^k \ln n, 0]$. For its default setting k=5 and n=256, this gives a lower bound of roughly $M \approx 177$. So this mitigates the limited-domain problem. THOR [38] also proposes a similar square-and-normalize approach that reduces the required number of iterations.

The precision of the above methods depends on the approximation degree of the inverse square root (or inverse) polynomial and the number of iterations k. In iterative softmax [16], these hyperparameters are manually configured based on expert heuristics.

Normalization Approximation. LayerNorm requires the *inverse square root* (invsqrt): LayerNorm(x_i)=$w_i \cdot (x_i-\mu)/\sqrt{\sigma^2+\epsilon}+b_i$, where μ and σ^2 denote the mean and variance of input $\boldsymbol{x}$, w_i and b_i are affine transform coefficients. RMSNorm is analogous by omitting the mean. Nearly all existing works [52, 54, 55] approximate invsqrt via Newton's method [42], often combined with Goldschmidt iterations [27] for high-precision refinement. For example, the NEXUS [54] codebase [47] uses ν=4 Newton steps to compute $1/\sqrt{x}$, followed by γ=2 Goldschmidt steps; MOAI [55] and ARION [52] adopt a similar variant.

The effectiveness of LayerNorm and RMSNorm under FHE hinges on whether the input variance (or root mean square) lies within the valid approximation interval of the invsqrt. At the same time, the numbers of Newton and Goldschmidt iterations in the above methods remain hand-tuned fixed values.

Activation Approximation. The GELU function is defined as $\text{GELU}(x) = x \cdot \Phi(x)$, where $\Phi(x)$ is the cumulative distribution function for Gaussian distribution, or by its tanh approximation. Early FHE Transformers [19, 54] and some interactive methods [23, 36, 39] use a piecewise GELU approximation, which employs a sign function approximated by minimax composition [33, 34] for segments selection. However, the piecewise GELU approximation is valid only for a small domain, e.g., $x \in [-8, 8]$ claimed in NEXUS [54], which is insufficient to cover the input range encountered during Transformer inference. To address this limitation, recent works [25, 38, 52, 55] explicitly define an approximate GELU with polynomials. For example, MOAI [55] uses a degree-23 polynomial; ARION [52]

adopts a degree-255 Chebyshev polynomial; and THOR [38] employs a composite polynomial with degree-31 and degree-27. For LLaMA's SiLU, MOAI [55] does not specify a degree, deferring to Orion [25]. However, the polynomial degrees are largely determined by heuristics and manual tuning in all these works.

2.4 Threat Model

In the context of secure inference, the following threat model is generally assumed [1, 32, 54]: a customer uploads encrypted data to the Cloud; the Cloud service provider then processes the ciphertext using neural networks deployed on its servers and returns the encrypted output to the customer; finally, the customer decrypts the result locally using a secret key. Under this model, the Cloud provider cannot access the sensitive information contained in the customer's input or output data, while the client remains unaware of the details of neural networks.

3 ATLAS: Adapting Transformers for FHE

ATLAS is an automated, framework-agnostic post-processing method that designs efficient polynomial approximation configurations for FHE transformers. Rather than applying a single set of hand-tuned hyperparameters uniformly across all layers, ATLAS assigns each layer its own approximation hyperparameters. We adopt a multi-objective evolutionary search algorithm to navigate the combinatorial space of layerwise approximation hyperparameters and discover configurations that are (i) significantly more efficient than hand-tuned hyperparameters, and (ii) span the accuracy-efficiency trade-off. Specifically, we adopt a two-stage evolutionary procedure that progressively relaxes layer-wise constraints.

3.1 Problem Formulation

Given a cleartext transformer f, we would like to develop its polynomial approximation $\tilde{f}_{\boldsymbol{\lambda}}$ by choosing appropriate hyperparameters $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_{softmax}, \boldsymbol{\lambda}_{norm}, \boldsymbol{\lambda}_{act})$ to configure softmax $\boldsymbol{\lambda}_{softmax}$, normalization $\boldsymbol{\lambda}_{norm}$, and activation function $\boldsymbol{\lambda}_{act}$ approximations. The goal is to minimize latency overhead without sacrificing security, while ensuring that $\tilde{f}_{\boldsymbol{\lambda}}$ exhibits similar predictive performance under FHE. This can be mathematically formulated as:

$$\begin{aligned} &\text{minimize}_{\boldsymbol{\lambda}}\ Lat(\tilde{f}_{\boldsymbol{\lambda}}, \mathcal{D}), \\ &\text{subject to}\ \left|Acc(f, \mathcal{D}) - Acc(\tilde{f}_{\boldsymbol{\lambda}}, \mathcal{D})\right| \le \epsilon_{acc}, \end{aligned} \tag{4}$$

where $Lat(\cdot)$ measures runtime latency of the approximated model $\tilde{f}_{\boldsymbol{\lambda}}$ under FHE, $Acc(\cdot)$ computes its accuracy over a representative set of data samples $\mathcal{D}$, and ϵ_{acc} signifies the error tolerance which is typically a small number approaching zero.

Directly solving Equation (4) imposes two main challenges: (1) measuring latency reliably and consistently on hardware is not trivial; (2) $Acc(\cdot)$ metric may not always be sensitive to deviations in model outputs due to approximation error. Accordingly, a more practical formulation of the problem is as follows.

$$\begin{aligned} &\text{minimize}_{\boldsymbol{\lambda}}\ Mul\_Depth(\tilde{f}_{\boldsymbol{\lambda}}), \\ &\text{subject to}\ \left|f(x) - \tilde{f}_{\boldsymbol{\lambda}}(x)\right| \le \epsilon\ \forall x \in \mathcal{D}, \end{aligned} \tag{5}$$

where $Mul\_Depth(\cdot)$ computes the total multiplicative depth and ϵ controls the tolerance on output deviation of a FHE transformer model $\tilde{f}_{\boldsymbol{\lambda}}$ from its cleartext counterpart f.

Unfortunately, choosing an appropriate ϵ in Equation (5) is also not straightforward, as different transformer models and tasks require different levels of precision (e.g., ViT can tolerate large deviation in outputs without dropping accuracy on ImageNet-1K classification compared to BERT on GLUE benchmark). An over-constrained ϵ leads to sub-optimal run-time latency, while an under-constrained ϵ leads to a noticeable drop in model performance. As opposed to adaptively/carefully tuning ϵ, we propose to formulate the problem as a multi-objective optimization problem

$$\begin{aligned} &\text{minimize}_{\boldsymbol{\lambda}}\ \left(Mul\_Depth(\tilde{f}_{\boldsymbol{\lambda}}), MAE(f, \tilde{f}_{\boldsymbol{\lambda}})\right), \\ &\text{where}\ MAE(f, \tilde{f}_{\boldsymbol{\lambda}}) = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}}\left|f(x) - \tilde{f}_{\boldsymbol{\lambda}}(x)\right|, \end{aligned} \tag{6}$$

to simultaneously balance run-time latency and approximation precision, circumventing the need of choosing an ϵ.

3.2 Search Space and Encoding

A decision vector $\boldsymbol{\lambda}$ specifies how softmax, normalization, and activation functions are approximated in each of the L Transformer layer ($L = 12$ for BERT-Base and ViT-Base, $L = 32$ for LLaMA3-8B). The decision variables are grouped by operators (i.e., softmax, normalization, and activation) and arranged in ascending layer order (i.e., from Layer 1 to L), as follows.

$$\begin{aligned} \boldsymbol{\lambda} = (&\underbrace{p_1^{(1)}, \ldots, p_5^{(1)}, \cdots, p_1^{(L)}, \ldots, p_5^{(L)}}_{\boldsymbol{\lambda}_{softmax}}, \\ &\underbrace{v_{attn}^{(1)}, \gamma_{attn}^{(1)}, v_{mlp}^{(1)}, \gamma_{mlp}^{(1)}, \cdots, v_{attn}^{(L)}, \gamma_{attn}^{(L)}, v_{mlp}^{(L)}, \gamma_{mlp}^{(L)}}_{\boldsymbol{\lambda}_{norm}}, \\ &\underbrace{p_{act}^{(1)}, \cdots, p_{act}^{(L)}}_{\boldsymbol{\lambda}_{act}}), \end{aligned} \tag{7}$$

resulting in a total of $5L + 4L + L = 10L$ integer variables.

Softmax $\boldsymbol{\lambda}_{softmax}$ Encoding. We adopt the iterative method proposed by Cho et al. [16] for softmax approximation, with one modification to use Chebyshev polynomials instead of power basis for coefficient estimation. Algorithm 1 outlines our procedure.

Algorithm 1: ITER. SOFTMAX$(x; k, p_1, \cdots, p_k)$ in ATLAS

Input: $x \in [-M, 0]^n$, $k \in \mathbb{Z}_{>0}$, $(p_1, \cdots, p_k)$
Output: $y \approx \text{Softmax}(x)$

$y \leftarrow \texttt{ChebyPolyEvalExp}(x/2^k, 15)$;
for $j \leftarrow 1$ **to** k **do**
  $a \leftarrow \texttt{ChebyPolyEvalInvsqrt}(\sum_{i=1}^{n} y_i^2, 2^{p_j} - 1)$;
  // Chebyshev invsqrt with degree $2^{p_j} - 1$
  $y \leftarrow (a \cdot y)^2$;
end
return y;

Given an input $\boldsymbol{x} \in [-M, 0]^n$, the algorithm first divides the input by 2^k and applies a 15-degree Chebyshev exponential approximation `ChebyPolyEvalExp`. It then applies k iterations, each consisting of: (i) summing the squared intermediate values, (ii) evaluating a `ChebyPolyEvalInvsqrt` that approximates the invsqrt, and (iii) multiplying and squaring the result. The key hyperparameters are the **total number of iterations** k and the **degree of the Chebyshev polynomial** p in each iteration. We embed the choice of k into the degree variables by allowing the p_j to be zero, indicating that the j-th and all its subsequent iterations are skipped.

In summary, each layer $i_{\in 1,\cdots,L}$ is assigned five integer variables $(p_1^{(i)}, p_2^{(i)}, p_3^{(i)}, p_4^{(i)}, p_5^{(i)})$, where $p_1^{(i)} \in [1, 7]$ and $p_2^{(i)}, \ldots, p_5^{(i)} \in [0, 7]$. A non-zero value determines the polynomial degree for the invsqrt function approximation in an iteration, and a zero value omits that iteration and all the subsequent ones. The upper bound is set to 7 following the settings from [16], and the lower bound is chosen such that at least one iteration is executed. The full softmax configuration contributing $5L$ integer variables is then encoded as:

$$\boldsymbol{\lambda}_{softmax}=(p_1^{(1)}, \cdots, p_5^{(1)}, \ \cdots, \ p_1^{(L)}, \cdots, p_5^{(L)}). \tag{8}$$

Normalization $\boldsymbol{\lambda}_{norm}$ Encoding. Both the Attention and MLP modules in a Transformer layer use normalization (e.g., LayerNorm and RMSNorm). Algorithm 2 outlines the Newton–Goldschmidt approximation of the invsqrt used in normalization. The two loop counters ν and γ directly control the accuracy–depth trade-off: larger values improve the approximation quality at the cost of additional multiplicative depth. We therefore adopt the default setting from the NEXUS codebase [47] ($\nu = 4$, $\gamma = 2$) as a reference and treat **both** (ν, γ) **counters as searchable integer parameters**.

Algorithm 2: NEWTON–GOLD INVSQRT$(x; \nu, \gamma)$ in ATLAS

Input: Input x; Newton steps ν, Goldschmidt steps γ
Output: $y \approx 1/\sqrt{x}$

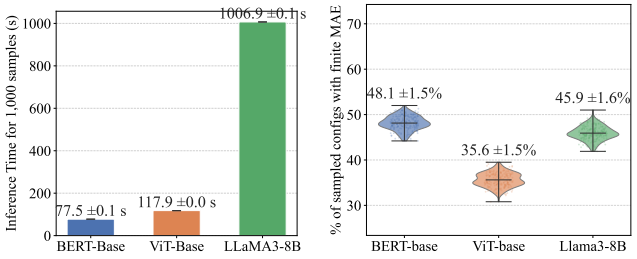
```
y ← −1.29×10^−4 · x + 0.129 ;                    // initial guess
for i ← 1 to ν do
    y ← y · (1.5 − 0.5 · x · y^2) ;                // Newton Iter
end
a, b ← xy, y/2;
for i ← 1 to γ do
    a, b ← a(1.5 − ab), b(1.5 − ab) ; // Goldschmidt Iter
end
return y ← 2b;
```

For each layer $i_{\in 1,\cdots,L}$, four integer variables are assigned: attention Newton iterations $\nu_{attn}^{(i)} \in [1, 4]$, attention Goldschmidt iterations $\gamma_{attn}^{(i)} \in [1, 2]$, MLP Newton iterations $\nu_{mlp}^{(i)} \in [1, 4]$, and MLP Goldschmidt iterations $\gamma_{mlp}^{(i)} \in [1, 2]$. The normalization variables of layer i and the full configuration vector with $4L$ integer variables are encoded as:

$$\boldsymbol{\lambda}_{norm} = (\nu_{attn}^{(1)}, \gamma_{attn}^{(1)}, \nu_{mlp}^{(1)}, \gamma_{mlp}^{(1)}, \ \cdots, \ \nu_{attn}^{(L)}, \gamma_{attn}^{(L)}, \nu_{mlp}^{(L)}, \gamma_{mlp}^{(L)}). \tag{9}$$

Activation $\boldsymbol{\lambda}_{act}$ Encoding. The non-linear activation function (e.g., GELU or SiLU) is approximated by a single Chebyshev polynomial [25, 52]. The degree directly determines the multiplicative depth consumed; we therefore include the **degree p_{act} in the search space** for exploring the accuracy–depth trade-off. Each layer i can select an integer $p_{act} \in [1, 9]$, yielding a polynomial degree of $2^{p_{act}}-1$. The full activation configuration vector is then:

$$\boldsymbol{\lambda}_{act} = (p_{act}^{(1)}, \ p_{act}^{(2)}, \ \cdots, \ p_{act}^{(L)}). \tag{10}$$



(a) Expensive evaluation **(b) Fraction of finite MAE**

Figure 3: Key challenges in searching for FHE approximation configurations. (a) Even a single cleartext evaluation of MAE is time-consuming, and FHE evaluation is orders of magnitude slower. (b) A large portion of the search space produces invalid (NaN/∞) MAE, resulting in a sparse feasible region.

3.3 Search Challenges

The search space defined in Section 3.2 is inherently combinatorial. For a single layer, the number of unique configurations is:

$$\begin{aligned} |\mathcal{S}_{layer}| &= |\mathcal{S}_{softmax}| \times |\mathcal{S}_{norm}| \times |\mathcal{S}_{act}| \\ &= \left(\sum_{k=1}^{5} 7^k\right) \times (4 \cdot 2 \cdot 4 \cdot 2) \times 9 \\ &= 19\,607 \times 64 \times 9 = 11\,293\,632 \approx 1.13 \times 10^7, \end{aligned} \tag{11}$$

where $|\mathcal{S}_{softmax}| = 19\,607$, $|\mathcal{S}_{norm}| = 64$, and $|\mathcal{S}_{act}| = 9$. When configurations are assigned per-layer, the total search volume grows to $|\mathcal{S}_{layer}|^L$, yielding $11\,293\,632^{12} \approx 4.3 \times 10^{84}$ for L=12 (BERT/ViT) and $11\,293\,632^{32} \approx 4.9 \times 10^{225}$ for L=32 (LLaMA3-8B). Beyond the sheer size of this space, the optimization must overcome several additional fundamental difficulties.

- **NP-Hard structure.** Selecting per-layer approximation hyperparameters under a depth–accuracy budget is a constrained integer program that generalizes multi-dimensional knapsack and per-layer precision assignment, both NP-hard. No polynomial-time exact algorithm is therefore expected, motivating heuristic, population-based search rather than exhaustive or exact optimization.
- **Conflicting objectives.** The two objectives—multiplicative depth and MAE—are inherently conflicting: higher-degree polynomials and more iterations reduce MAE but consume more depth, and vice versa. Finding a single solution that simultaneously minimizes both objectives is generally infeasible; instead, we seek a set of Pareto-optimal solutions that represent the efficient trade-off between the two.

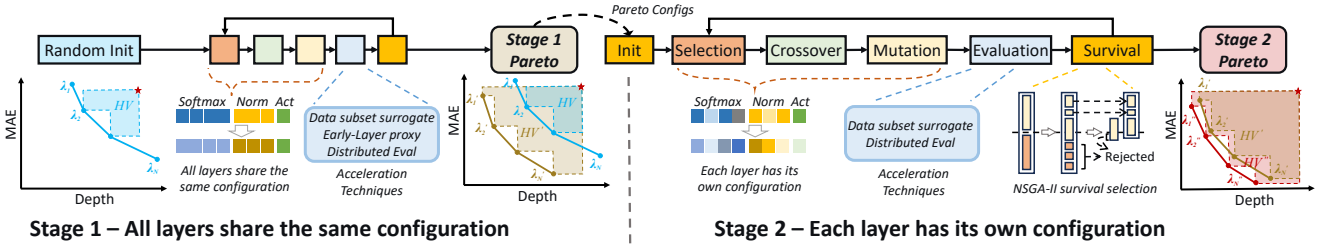


Figure 4: ATLAS is a two-stage process. Stage 1 identifies a set of efficient layer-uniform approximations, and Stage 2 refines these into heterogeneous layer-wise configurations to further improve the depth–accuracy trade-off.

- **Expensive evaluation.** Each candidate configuration must be instantiated as a model and evaluated to obtain its MAE and depth. Even in the cleartext polynomial setting used during search, a single function evaluation (FE) is costly for computing MAE. As shown in Figure 3(a), a single FE under cleartext already takes 77.5 s for BERT, 118 s for ViT, and 1007 s for LLaMA; under end-to-end FHE, the latency is even higher. This cost compounds in population-based methods, where every generation evaluates an entire population, making naive deployment of evolutionary search prohibitive without aggressive evaluation acceleration.
- **Sparse feasible region.** A large fraction of the search space yields configurations that produce NaN or infinite MAE, rendering them useless. Figure 3(b) shows that the proportion of configurations with finite MAE is below 50% for all three models. This sparse signal makes the optimization landscape highly discontinuous, further complicating the search.

3.4 Two-Stage Search with NSGA-II

In ATLAS, to address the challenges outlined in Section 3.3, we develop a two-stage evolutionary optimization strategy based on the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [20]. The core idea is to decompose the search: first explore the compact LayerProblem, where all L layers share the same configuration; then refine them into per-layer settings by solving NetworkProblem, where each layer can have different configurations. As shown in Figure 4, the first stage is computationally efficient and produces a set of layer-uniform yet high-quality configurations that serve as a **warm start** for the second stage, thereby drastically accelerating convergence.

3.4.1 Preliminaries on NSGA-II. ATLAS adopts NSGA-II [20] to handle the two competing objectives—multiplicative depth and MAE—through Pareto dominance: $\boldsymbol{\lambda}_1$ dominates $\boldsymbol{\lambda}_2$ if it is no worse in both objectives and strictly better in at least one. The non-dominated set forms the Pareto front, whose quality we measure by the *hypervolume* (HV) [57] with respect to a fixed reference point; a larger HV in the (depth, MAE) space indicates a front that achieves lower depth, lower error, or a better trade-off between the two.

NSGA-II evolves a population of solutions (i.e., configuration vectors) over generations through four operators: *selection*, *crossover*, and *mutation* produce new candidate solutions from the current population, and *survival selection* then determines which solutions advance to the next generation. The survival step is the core of NSGA-II: given a combined pool of current and newly generated solutions, *non-dominated sorting* partitions the pool into fronts $\mathcal{F}_1, \mathcal{F}_2, \ldots$, admitting solutions front by front; when a front cannot be fully admitted, *crowding distance*—the objective-space distance to neighbors within the same front—serves as a tiebreaker, favoring solutions in less densely populated regions. Together, the two criteria balance convergence toward the Pareto front with diversity along it, as illustrated in Figure 4. The main evolutionary operators are summarized below.

- `Selection.` Promising solutions, referred to as *parents*, are chosen via binary tournament selection: two individuals are randomly sampled, and the one with better non-dominated rank is kept. This process repeats until enough parents are selected.
- `Crossover.` A standard two-point crossover is used to exchange sub-components between two parents at each crossover point to create a new set of solutions, referred to as *offspring*.
- `Mutation.` An integer step mutation operator then perturbs each variable in the offspring solution by +1 or -1 following a binomial distribution with a probability proportional to one over the number of variables, rounding to the nearest integer within the bounds.

3.4.2 Two-Stage Framework. However, directly applying NSGA-II to the $10L$-variable NetworkProblem suffers from the curse of dimensionality: the initial population is extremely sparse, and the search wastes many evaluations on infeasible or poor regions. We observed that a configuration where all layers are identical already yields a reasonable baseline; evaluating it on the LayerProblem is fast, and the HV grows quickly. In contrast, the per-layer search requires tens of thousands of FEs to reach a comparable HV. This motivates an efficient two-stage design:

- LayerProblem — all L layers share the same configuration by constraining:

$$\begin{cases} \boldsymbol{p}^{(1)} = \boldsymbol{p}^{(2)} = \cdots = \boldsymbol{p}^{(L)}, \text{ where } \boldsymbol{p} = (p_1, p_2, p_3, p_4, p_5) \\ v_{attn}^{(1)} = v_{attn}^{(2)} = \cdots = v_{attn}^{(L)},\ v_{mlp}^{(1)} = v_{mlp}^{(2)} = \cdots = v_{mlp}^{(L)} \\ \gamma_{attn}^{(1)} = \gamma_{attn}^{(2)} = \cdots = \gamma_{attn}^{(L)},\ \gamma_{mlp}^{(1)} = \gamma_{mlp}^{(2)} = \cdots = \gamma_{mlp}^{(L)} \\ p_{act}^{(1)} = p_{act}^{(2)} = \cdots = p_{act}^{(L)}. \end{cases} \tag{12}$$

- NetworkProblem — each layer has its own configuration, i.e. the $10L$-variable bounded integer space represented by $\boldsymbol{\lambda}$ in (7).

Stage 1. NSGA-II searches the single-layer configuration that is applied identically to all L layers. With a moderate budget (e.g., 1/10 of the total budget), the algorithm produces a Pareto set of high-quality layer-uniform configurations $\boldsymbol{\lambda}_{layer}$.

Stage 2. These $\boldsymbol{\lambda}_{layer}$, together with a small subset of randomly created configurations, form a *seeded* initial population of the NetworkProblem for NSGA-II. Because $\boldsymbol{\lambda}_{layer}$ already resides in promising regions of the objective space, the search starts from a favorable basin and efficiently refines per-layer configurations. The Stage-2 budget (e.g., 9/10 of the total budget) is set to match the total evaluation budget.

3.5 Accelerating the Search

Even with the two-stage strategy, each evaluation call remains expensive. To further reduce the cost of the search, we introduce two surrogate models that approximate the true objectives at a fraction of the cost and a distributed evaluation technique.

Data Subset Surrogate. A single evaluation over the full dataset dominates the search time. We observe that the MAE computed on a small random subset of the data is highly correlated with the full-dataset MAE. For a given configuration, one can view the subset MAE as an estimate of the true full-dataset MAE. That is, following Equation (6), we are not evaluating all $x \in \mathcal{D}$, but take a sampled subset of $\mathcal{D}$. To assess the reliability of this surrogate, we measure the Kendall τ rank correlation between the two quantities: τ close to 1 indicates that the ordering of configurations is largely preserved.

Figure 5 (top right) reports τ and the resulting speed-up as a function of subset size. For BERT, a subset of 50 IMDB sentences achieves $\tau \approx 0.85$ with ~470× speed-up, sufficient for reliable ranking. For ViT, even 10 ImageNet validation images yield $\tau \approx 0.936$ with ~440× speed-up. Figure 5 (bottom left) and Figure 5 (bottom right) illustrate this correlation for the chosen sizes: each point represents one configuration whose evaluation is performed on a random subset, with the corresponding full-dataset MAE on the vertical axis. In practice, we set the proxy subset size to 10 for all three models (BERT, ViT, and LLaMA) throughout the search to achieve maximum speed-up.

Early-Layer Proxy. In the LayerProblem, all layers share the same configuration; consequently, the MAE measured at intermediate Transformer layers is strongly correlated with the final-layer MAE. Figure 5 (top left) plots Kendall τ between early layer-i MAE and last layer MAE against the resulting speed-up. For BERT, layer 4 already gives $\tau = 0.927$ with ~3× speed-up; for ViT, layer 7 achieves $\tau = 0.914$ with ~1.8× speed-up. Figure 5 (middle left) and Figure 5 (middle right) show the corresponding relations between last layer MAE and layer-i MAE. Stopping the forward pass at an early layer avoids the computation of all subsequent Transformer layers, yielding an acceleration proportional to L/i. This proxy is only valid when all layers are identical (Stage 1) and is disabled in Stage 2, where per-layer configurations vary independently.

For the surrogate models of LLaMA3-8B, we use a data subset surrogate with 10 IMDB sentences, with $\tau = 0.905$ and ~454× speed-up, and a layer-9 MAE as the early-layer proxy, with $\tau = 0.924$ and ~8.5× speed-up. The detailed experiments are shown in Appendix E.

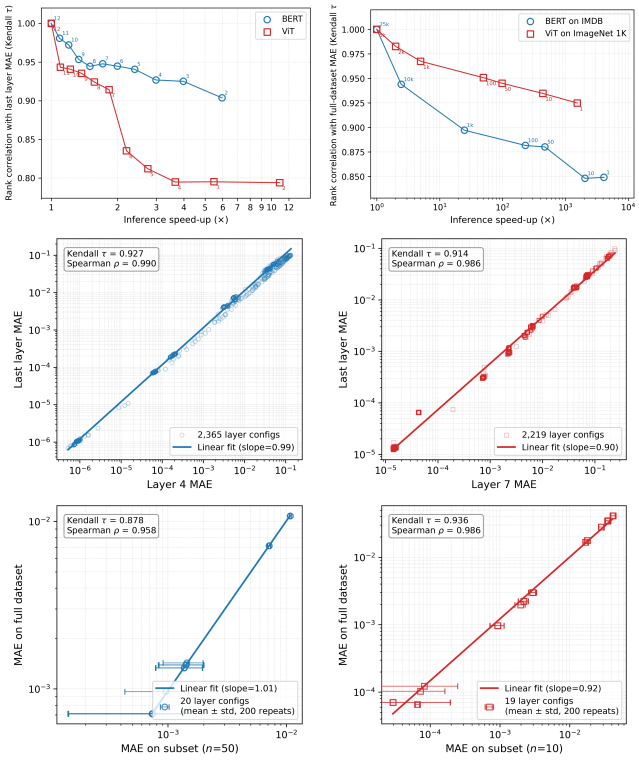


Figure 5: MAE at an early layer as a surrogate for MAE at the last layer, and the correlation between the MAE on the full and subset of the dataset, are shown on BERT and ViT.

Distributed Evaluation. Evaluation is further accelerated by distributing the population across multiple GPUs, exploiting the natural parallelism of evolutionary search: the fitness of every individual in a generation can be computed independently.

Overall Speed-Up. Combining the above techniques reduces the effective cost of a single evaluation by a factor of several hundred to over one thousand in Stage 1. In Stage 2, we drop the early-layer proxy as different layers may use different configurations, yet the search is already warm-started in a promising region of the objective space. Together, these techniques make the ATLAS tractable even for large models such as LLaMA3-8B.

4 Experimental Validation

ATLAS automatically approximates pre-trained transformers for efficient homomorphic encryption through evolutionary search. Here, we design experiments to answer the following research questions (RQs):

- **RQ1:** *Can* ATLAS *discover transformer approximation hyperparameters that improve the accuracy–depth (inference latency) trade-off achieved by existing hand-tuned secure transformer inference solutions?* (Section 4.2)
- **RQ2:** *What insights does* ATLAS *reveal about the approximation hyperparameters that improve the accuracy–depth?* (Section 4.3)

- **RQ3:** *How efficient and generalizable is* ATLAS *at approximating pre-trained transformers that span the accuracy-efficiency trade-off?* (Section 4.4)

4.1 Experimental Setup

Datasets. During search, both BERT and LLaMA use 10 randomly sampled IMDB [37] training sentences as the proxy subset for MAE evaluation. For the task evaluation, BERT is assessed on three GLUE benchmark tasks [50]: SST-2, RTE, and QNLI; LLaMA is evaluated on MMLU [29], GSM8K [18], and HumanEval [7]. Specifically, we evaluate MMLU on 64 sampled test examples from its abstract algebra subset. ViT uses 10 random ImageNet-1K [21] images as the search proxy, and the full ImageNet-1K validation set for final Top-1 accuracy measurement.

Hardware, Searching and FHE Libraries. The multi-objective search is implemented on top of `PyMOO 0.6.1.6`, a Python framework for evolutionary optimization on cleartext polynomial model evaluation, with `HuggingFace` and `Transformers 4.57.1` providing off-the-shelf model architectures and weights, and our tensor backend is the `PyTorch 2.6.0` + `CUDA 12.4`.

The FHE backend is `Phantom-FHE` [51], a GPU-accelerated C++ homomorphic encryption library that offers an interface similar to Microsoft `SEAL` [45] but operates on CUDA-enabled devices. Our FHE inference system additionally relies on `LibTorch 2.6.0` + `CUDA 12.4` for model weights loading.

All search runs and FHE inference latency measurements were conducted on a server equipped with the NVIDIA RTX 4090 48 GB GPUs and an AMD EPYC 7343 16-core 32-thread processor with 256 GB RAM. We implemented the `ATLAS` FHE inference system in CUDA/C++ on top of NEXUS-EndtoEnd [47] and `Phantom-FHE` [51], under the RNS-CKKS scheme. The system adopts the ciphertext and plaintext row-packing strategies as well as the PCMM and CCMM from NEXUS-EndtoEnd, and extends the CCMM and PCMM operations to meet the requirements of ViT and LLaMA.

Bootstrapping placement in our FHE inference system follows a strategy similar to that of THOR [38]: a level check is placed before every heavy homomorphic operator, and bootstrapping is triggered only when the remaining level of a ciphertext is insufficient for the next operation. Each bootstrapping acts on the 32,768 slots, and the reported #Boot is the number of such 32K-slot bootstrappings consumed by a full forward pass.

Search Parameters. The two-stage NSGA-II shares a unified set of hyperparameters across all three architectures. In LayerProblem (Stage 1), the population size is 48 and the algorithm runs for 50 generations, yielding $48 \cdot (50 + 1) = 2{,}448$ FEs. In NetworkProblem (Stage 2), the population size is 96 and the algorithm runs for 225 generations, yielding $96 \cdot (225 + 1) = 21{,}696$ FEs. The total FE budget per run is therefore 24,144 across all models. Both crossover and mutation operators are consistent across stages: crossover is a two-point homogeneous operation with probability 0.9; mutation applies an integer step perturbation to each individual with probability 0.9, while the per-variable mutation rate is set to $1/n_{\text{var}}$.

We use two objective metrics: MAE (defined in Equation (6)) and the total approximation multiplicative depth. The *total approximation multiplicative depth* is the sum of the multiplicative depths of all non-linear approximation components in Transformers; the depth of homomorphic matrix multiplications is excluded because it is fixed for a given architecture and does not affect the search.

RNS-CKKS Parameters. We follow the cryptographic parameterization as prior FHE inference works [1, 32, 54]. The cyclotomic polynomial degree is $N = 2^{16}$; the ciphertext budget is $D = 28$; bootstrapping consumes $K = 14$ levels, leaving $D - K = 14$ multiplicative levels after a refresh. The base, special, and bootstrapping moduli each use 51 bits, while the default modulus uses 46 bits [13]. This configuration guarantees 128-bit security.

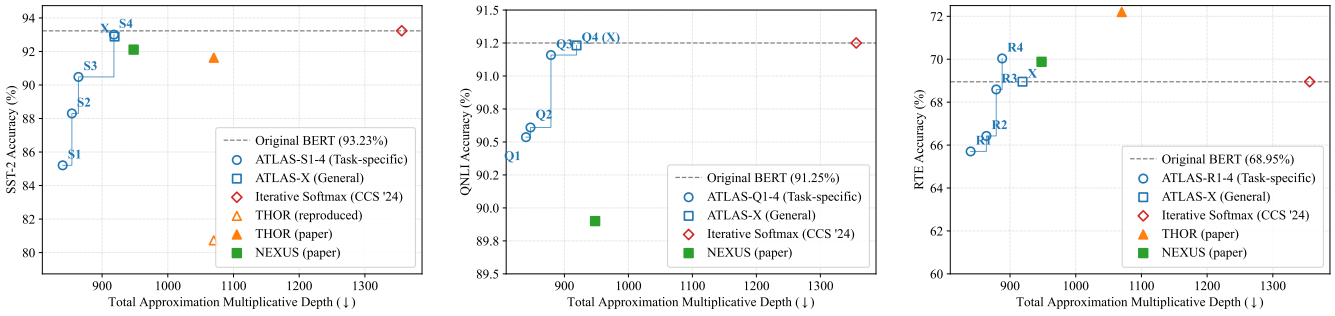
Baselines. Architecture Baseline: We benchmark the proposed `ATLAS` on three representative Transformer architectures with open-source weights in `HuggingFace`: the encoder of BERT-base (L=12), the vision encoder of ViT-Base (L=12), and the decoder of LLaMA3-8B (L =32). For BERT and LLaMA, the number of tokens is set to 128; the original ViT-Base (Patch16-224) sequence length of 197 is padded to 256 to match the slots.

Approximation Baseline: For comparison, we reproduce the non-linear approximation recipes of NEXUS [54] and THOR [38] within our FHE inference system. For the iterative softmax of Cho et al. [16], we pair it with NEXUS [54] LayerNorm setting (ν=4, γ=2) and a Chebyshev GELU [25, 52] of degree 511 to obtain a high-precision complete end-to-end pipeline. All baselines are evaluated under the same cryptographic and hardware settings. Because the level of input ciphertext, homomorphic MM, packing strategies, and bootstrapping placements vary substantially across frameworks, our reproducing end-to-end latencies of NEXUS [54] and THOR [38] reported here reflect only the approximation configuration and are not directly comparable to those in the original papers; we deliberately adopt a simple row-packing homomorphic MM instead of the highly optimized alternatives (discussed in Section 5.2) to ensure a fair, isolated evaluation of the approximation recipes alone.

4.2 Main Results of ATLAS

For each neural network architecture, `ATLAS` obtains a set of MAE–depth Pareto configurations with two-stage search (as shown in Figure 4) on a small set of proxy data (e.g., IMDB). Then, `ATLAS` evaluates these configurations on each downstream task to build the accuracy–depth Pareto front (or accuracy–latency Pareto front, see Appendix F for details).

4.2.1 BERT. The hand-tuned high-precision baseline using iterative softmax [16] (denoted IS) operates at total approximation depth 1356 with 588 bootstrappings, yielding a latency of 1107.5 s on RTX 4090 and accuracies of 93.2% on SST-2, 69.0% on RTE, and 91.3% on QNLI. Figure 6 reports the accuracy–depth Pareto fronts discovered by `ATLAS` on the three GLUE tasks: SST-2, RTE and QNLI. The fronts are steep: a modest increase in total approximation multiplicative depth yields configurations that match or exceed the IS in accuracy, while reducing latency substantially. For example, on SST-2, S4 comes within 0.2 pp of the IS at 93.0% accuracy and 23.6% lower latency; on RTE, R4 surpasses IS with 70.0% accuracy and 25.6% lower latency. The best configuration differs by task—S4 for SST-2, Q4 for QNLI, R4 for RTE—because different tasks have different preferences for the approximation configuration. This task-specific preference is exactly what a hand-crafted recipe cannot



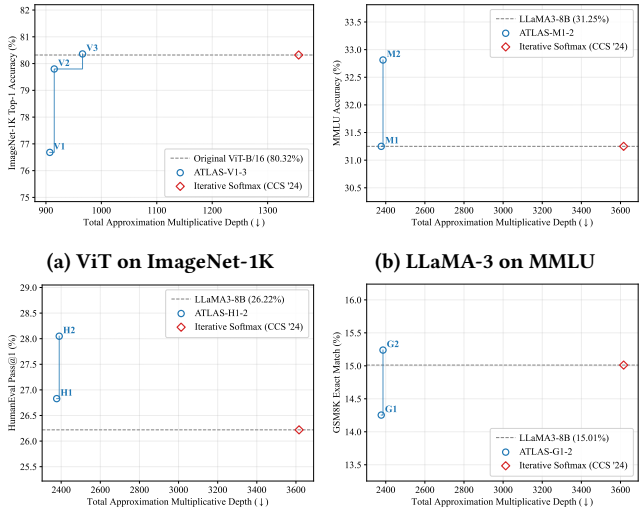
(a) SST-2 **(b) QNLI** **(c) RTE**

Figure 6: Pareto fronts discovered by ATLAS on three tasks of BERT. Each point represents a non-dominated λ configuration, plotted in the space of multiplicative depth vs. downstream task accuracy. The iterative softmax baseline is shown as a reference. Dashed horizontal lines indicate the accuracy of the underlying cleartext model.

capture, and what ATLAS discovers automatically. Table 1a summarizes the results of Pareto configurations of BERT. Although several configurations are shared across tasks (S1=Q1=R1, S3=R2, Q3=R3), the task-specific choices differ, confirming that ATLAS adapts to the demands of each downstream task. Notably, configuration Q4 still performs well on all three tasks, providing a general balanced choice (denoted X) for multi-task deployment. As a result, for **RQ1**, ATLAS can automatically discover configurations that preserve or improve accuracy at substantially lower depth and latency in encoder-only architectures, advancing the accuracy–depth frontier over the hand-tuned baseline.

Comparison with NEXUS and THOR. As shown in Table 1a, we reproduce the non-linear approximations of NEXUS [54] and THOR [38] in $D{-}K = 14$ levels, and evaluate them in our end-to-end system; these results are marked with †. The reproduced accuracies fall below reports in their original papers. We conjecture that these hand-crafted recipes hard-code approximation domains—such as the GELU input interval and the constant softmax maximum—that are calibrated to the activation statistics of the specific model weights used in each work (and, in THOR's case, to fine-tuned weights that are not publicly released). When the same recipes are applied to the open-source weights adopted in our reproduced setup, the activations fall outside these fixed domains, degrading accuracy. This highlights the generalizability of ATLAS. Notably, ATLAS configurations also outperform both reproduced NEXUS and THOR in efficiency: it achieves lower depth, fewer bootstrappings, and lower latency than NEXUS and THOR.

Latency Breakdown. For an example, Figure 2 decomposes the end-to-end latency of IS (denoted FHE BERT) and S2 (i.e., ATLAS FHE BERT in Figure 2) on an RTX 4090. ATLAS achieves a 28.5% speedup compared with baseline. Beyond the direct savings in the three approximations with ~220 s, ATLAS configuration yields an additional indirect benefit: because the iterative softmax terminates at a lower multiplicative level 3 (occasionally 4 or 5), the subsequent CCMM $\boldsymbol{So}_h\boldsymbol{V}_h$ ($\boldsymbol{So}_h = \mathrm{Softmax}(\boldsymbol{Q}_h\boldsymbol{K}_h{}^T/\sqrt{d_k})$) operates at a reduced level. In the baseline, each $\boldsymbol{So}_h\boldsymbol{V}_h$ consumes 1.89 s at level 14, whereas ATLAS executes it in only 0.62–0.82 s, accounting for roughly ~90 s of the total latency reduction.



(a) ViT on ImageNet-1K **(b) LLaMA-3 on MMLU**

(c) LLaMA-3 on HumanEval **(d) LLaMA-3 on GSM8K**

Figure 7: Pareto fronts discovered by ATLAS on ViT and three tasks of LLaMA. Each point represents a non-dominated λ configuration, plotted in the space of multiplicative depth vs. downstream task accuracy. The iterative softmax baseline is shown as a reference. Dashed horizontal lines indicate the accuracy of the underlying cleartext model.

4.2.2 ViT. The IS baseline achieves a top-1 accuracy of 80.32% with a latency of 2944.8 s, a total approximation multiplicative depth of 1356, and 1608 bootstrappings. Figure 7a shows the accuracy–depth Pareto front discovered by ATLAS on ImageNet-1K, which has three configurations V1, V2 and V3, and a similar steep shape. The best configuration V3 surpasses the baseline with 80.36% accuracy while reducing depth by 28.8%, bootstrappings by 23.5%, and latency by 18.5%. Table 1b lists the Pareto configurations of ViT. For **RQ1**, these results also confirm that ATLAS can surpass the hand-tuned configuration in the vision encoder architecture.

Table 1: Approximation depth, bootstrapping count, latency, and downstream accuracy of FHE Transformer inference. Mul Depth – total approximation multiplicative depth; #B – #Boot; Lat. – end-to-end FHE inference latency on RTX 4090. † indicates results produced with our framework. The Pareto-optimal ones are highlighted in bold.

	Mul Depth	#B	Lat. (s)	Accuracy (%)		
				SST2	QNLI	RTE
BERT-Base [22]	-	-	-	93.2	91.3	69.0
IterSoftmax [16]	1356	588	1108	93.2	91.3	69.0
NEXUS [54]	-	-	-	92.1	89.9	69.9
NEXUS†	948	912	1134	49.1	49.5	52.3
THOR [38]	-	-	-	91.6	-	72.2
THOR†	1070	732	1023	80.7	52.4	52.2
ATLAS-S1 (R1, Q1)	840	390	793	85.2	90.5	65.7
ATLAS-Q2	847	390	795	88.2	90.6	65.0
ATLAS-S2	854	390	792	88.3	90.5	65.3
ATLAS-S3 (R2)	864	393	806	90.5	89.7	66.4
ATLAS-R3 (Q3)	879	393	809	87.4	91.2	68.6
ATLAS-R4	888	402	824	85.8	90.8	**70.0**
ATLAS-S4	918	411	846	**93.0**	90.6	68.6
ATLAS-Q4 (X)	919	408	845	92.9	**91.2**	68.9

(a) BERT-Base (L=12).

	Mul Depth	#B	Lat. (s)	Top-1 Accuracy (%)
				ImageNet-1K
ViT-Base [24]	-	-	-	80.32
IterSoftmax [16]	1356	1608	2944.8	80.32
ATLAS-V1	907	1206	2320.1	76.68
ATLAS-V2	915	1248	2445.8	79.80
ATLAS-V3	966	1230	2398.5	**80.36**

(b) ViT-Base (L=12).

	Mul Depth	Accuracy (%)		
		MMLU	GSM8K	HumanEval
LLaMA3-8B [48]	-	31.25	15.01	26.22
IterSoftmax [16]	3616	31.25	15.01	26.22
ATLAS-M1 (G1, H1)	2377	31.25	14.25	26.83
ATLAS-M2 (G2)	2386	**32.81**	**15.24**	25.61
ATLAS-H2	2390	31.25	14.71	**28.05**

(c) LLaMA3-8B (L=32).

4.2.3 LLaMA3. On three benchmarks (MMLU, GSM8K, and HumanEval) under a zero-shot, no-chain-of-thought (no-CoT) setting with 128 tokens, the IS has a total approximation depth of 3616 and achieves 31.25% on MMLU, 15.01% exact match on GSM8K, and 26.22% pass@1 on HumanEval. Figure 7b (MMLU), Figure 7d (GSM8K), and Figure 7c (HumanEval) report the accuracy–depth Pareto fronts discovered by ATLAS, reducing depth by 33.9%–34.3% across all tasks while matching or exceeding baseline accuracy, and the fronts are also steep. As a result, for the best configurations, the M2 (also G2) improves MMLU to 32.81% and GSM8K to 15.24% with 34.0% lower depth; on HumanEval, H2 improves pass@1 to 28.05% with 33.9% depth reduction. Table 1c summarizes the above configurations. Notably, configurations are shared across tasks: M1=G1=H1 and M2=G2. Overall, for **RQ1**, these results also show that ATLAS can capture the task-specific configurations compared with a hand-crafted recipe in decoder-only architecture.

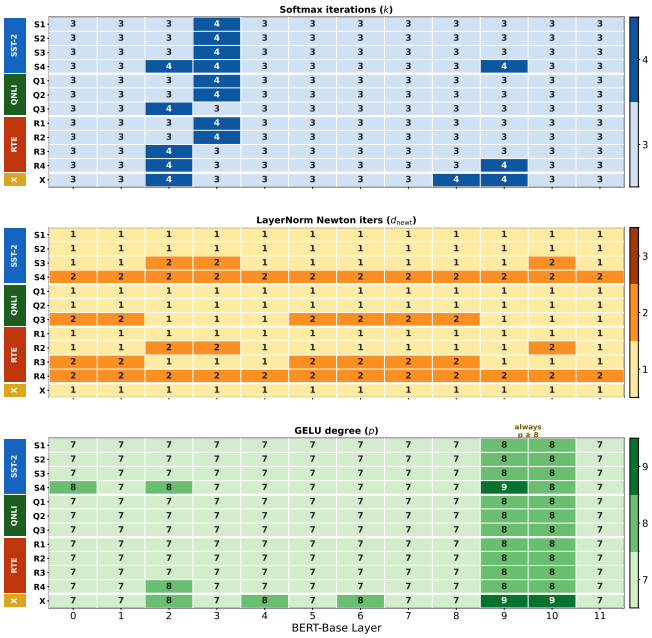


Figure 8: Per-layer visualization of the Pareto configs of ATLAS for BERT shown in Figure 6.

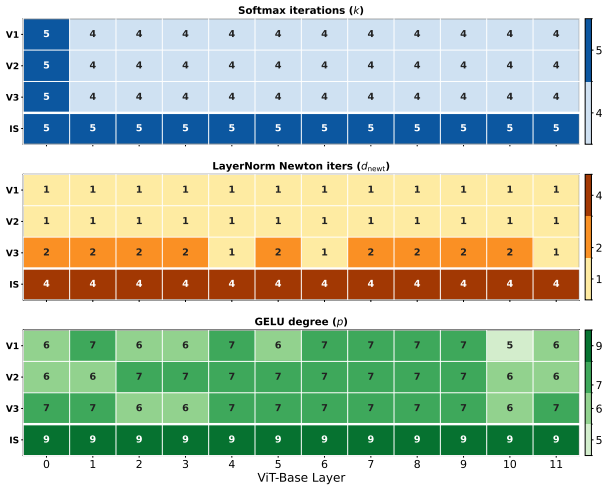


Figure 9: Per-layer visualization of the Pareto configs of ATLAS for ViT shown in Figure 7a.

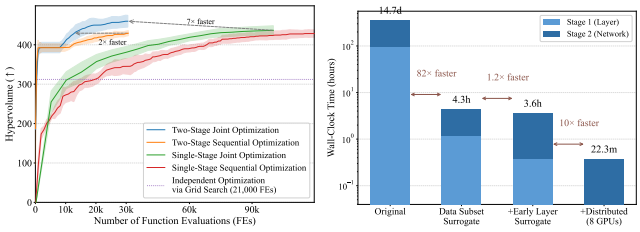
4.3 Layerwise ATLAS Analysis

For **RQ2**, to better understand how ATLAS achieves the accuracy–efficiency trade-offs reported earlier, we examine the per-layer hyperparameter configurations of several representative Pareto-optimal solutions. Two trends emerge from the visualizations in Figures 8 and 9.

Layerwise Allocation Can Expose Redundancy. As shown in Figure 8, for BERT, almost every solution across the three GLUE tasks sets the iterative softmax iteration count to $k = 3$, substantially lower than the default $k = 5$ used by IS from [16]. Because k determines the effective lower bound of the softmax input domain $(-2^k \ln 128)$, the $k = 3$ suggests that almost all BERT Attention scores lie within $[-38.8, 0]$, and the uniform $k = 5$ is largely redundant. Only layers 3, 4, 9, and 10 occasionally increase k to 4, indicating a limited need for a wider domain $[-77.6, 0]$. Also, a similar pattern holds for other modules: the Attention LayerNorm consistently uses $v = 1$ or 2 Newton iterations across all configurations, well below the NEXUS reference of $v = 4$; for the GELU activation, most layers settle at $p = 7$ (degree 127), while layers 10 and 11 require $p = 8$ or 9 (degree up to 511). These observations suggest that a large portion of the uniform safety margins built into manually designed approximations can be trimmed when layers are automatically configured independently.

Architecture-Specific Preferences. As shown in Figure 9, the optimal configuration pattern differs notably between BERT and ViT. In contrast to BERT, the ViT solutions use $k = 4$ for 11 of the 12 layers and $k = 5$ for the first layer, implying that the ViT Attention scores span a wider numerical range on the image data; and the Attention LayerNorm v values follow a similarly conservative profile. The GELU degrees are generally lower in ViT (i.e., $p = 5{\sim}7$ vs. $p = 7{\sim}9$ in BERT). Consequently, the relative latency reduction over IS is smaller for ViT than for BERT, which may simply reflect the different intrinsic approximation demands of the two architectures rather than any limitation of the search. Together, these findings indicate that the ideal approximation recipe is architecture- and dataset-dependent, and that an automated per-layer search, i.e., ATLAS, can help uncover such dependencies that are difficult to anticipate manually.

4.4 Search Efficacy



(a) HV vs. Num of FEs **(b) HV vs. Wall-clock Time**

Figure 10: Search efficiency of ATLAS on BERT. (a) Comparison of optimisation strategies in terms of hypervolume achieved for a given number of function evaluations. (b) Impact of the acceleration techniques introduced in Section 3.5: each curve adds one more technique on top of the previous.

Comparison of Optimization Strategies. Figure 10a tracks the HV of the Pareto front as a function of the number of FEs for five representative strategies. Recall that a larger HV in the (depth, MAE) space indicates a superior trade-off between multiplicative depth and approximation error. The most basic strategy, **independent optimization via grid search**, separately grids the softmax space and the combined normalization + activation space, then picks the best combination. It requires roughly 21,000 FEs to reach an HV of ∼310, serving as a lower-bound baseline.

Sequential optimization (either single-stage or two-stage) optimizes one group of variables at a time while freezing the others, cycling through softmax, normalization, and activation. In the single-stage variant, all layers are jointly configured; in the two-stage variant, a warm-start is used before per-layer tuning. Both are consistently outperformed by their **joint optimization** counterparts, where all variables are evolved together. For instance, two-stage joint optimization reaches an HV of ∼420+ in only ∼14,000 FEs, roughly twice as fast as two-stage sequential optimization (∼30,000 FEs). This gap confirms that **the three approximation blocks in Transformer are interdependent and should not be treated in isolation**.

The **two-stage joint optimization** (our ATLAS configuration) further accelerates progress: it achieves an HV of ∼450+ within ∼30,000 FEs, whereas single-stage joint optimization plateaus at a lower HV despite consuming more evaluations. The two-stage design therefore provides a 7× speed-up over single-stage joint optimization for reaching a given HV, highlighting the benefit of the layer-wise warm-start.

Impact of Acceleration Techniques. Figure 10b breaks down the wall-clock time required to reach a target HV when the acceleration techniques from Section 3.5 are incrementally turned on. The original two-stage NSGA-II without any surrogate would require roughly 14.7 days to complete the search budget.

The **data subset surrogate** alone reduces this time to 4.3 hours, an 82× speed-up, by evaluating MAE on only 10 representative samples instead of the full calibration set. Adding the **early-layer proxy** in Stage 1 further shortens the run to 3.6 hours (1.2× additional speed-up), because the forward pass is truncated at an intermediate layer where the MAE is strongly correlated with the last-layer MAE. Finally, **distributed evaluation** over 8 GPUs brings the search time down to 22.3 minutes, an extra 10× acceleration obtained from the natural parallelism of the evolutionary algorithm. Altogether, for **RQ3**, these techniques shrink the search time from more than two weeks to under half an hour, making the optimization pipeline practical not only for small models with encoder-only and vision encoder architecture, but also for large models such as LLaMA.

5 Related Work

5.1 Non-linear Approximation Design

Designing New Approximations. To enable secure Transformer inference under FHE, recent efforts focus on designing approximations for the key non-linear components: softmax, normalization, and activation. One line of works [40, 44, 56] design HE-friendly alternatives and retrain the entire model; however, full retraining

or fine-tuning is computationally intensive and expensive for modern LLMs. The most relevant direction to our work is therefore the post-training approach: replacing the non-linear operations directly with high-precision polynomial approximations while keeping the pretrained weights frozen, as done in NEXUS [54], THOR [38], and MOAI [55], etc. Additionally, several studies specifically tackle the most challenging operator in Transformer, softmax, with specialized approximation strategy [16, 38, 41].

Manual Configuration. Despite their effectiveness, all above methods rely on heuristics and expert experience to set approximation hyperparameters such as polynomial degree and iteration counts. These approximations are often adapted from numerical methods (e.g., Goldschmidt algorithm [27], Newton's method, and Remez algorithm for power-basis/Chebyshev polynomial, etc.), and a significant safety margin is deliberately built in to ensure correctness. Within deep neural networks, however, the precision requirements typically leave a considerable slack that heuristic or hand-tuned configurations often fail to exploit.

Automatic Configuration. The work closest to ours is AutoFHE [1], which uses multi-objective optimization and full fine-tuning to search per-layer ReLU approximations in FHE CNNs from a small set of three candidate degrees (identity, quadratic, and minimax composite [33, 34]). However, this approach is limited to CNNs and does not readily transfer to Transformers, especially modern LLMs. Since the number of non-linearities is much larger, the search space explodes, the functional forms are more diverse, and the fine-tuning cost becomes prohibitive. Prior to AutoFHE, several works applied neural architecture search to FHE CNNs to reduce the number of approximated ReLUs and select architectures [26, 30, 46]. These methods require neural architectural modifications and training from scratch, making them equally ill-suited for Transformers.

In contrast, ATLAS is the first method to automatically configure per-layer polynomial approximations for FHE Transformers. Unlike AutoFHE, whose search is coupled to gradient-based fine-tuning of the network, ATLAS operates entirely post-training on a frozen pretrained model, and therefore scales to settings where fine-tuning is prohibitively expensive, such as the 8B-parameter LLaMA3. It further generalizes across encoder-only (BERT), decoder-only (LLaMA3), and vision (ViT) Transformers—rather than the ReLU-only CNNs targeted by prior automated methods—systematically exploiting the precision slack that hand-tuned configurations leave unused.

5.2 FHE Inference System Design

Efficient Homomorphic Matrix Multiplication. Several recent frameworks have focused on optimizing packing and homomorphic MM to reduce overhead. For example, NEXUS [54] eliminates wasted ciphertext slots through compression/decompression and exploits monomial properties to lower communication. THOR [38] formulates MM via diagonals, using ciphertext rotations and entrywise products to compute matrix products efficiently, and packs multiple diagonals into one ciphertext for parallelism.

End-to-End Transformer Inference. While many works provide individual homomorphic operators, few deliver a complete, reproducible end-to-end pipeline. NEXUS [54] offers standalone components, but how multiple packing strategies should coexist or interconvert across modules remains unclear. THOR [38] presents an end-to-end system, yet it requires precomputing and storing over 180 GB of model weights and keys, and demands at least 80 GB of GPU memory for a single BERT inference; moreover, the supplied configuration is tied to its precomputed weights and cannot easily generalize to other downstream tasks. MOAI [55] and ARION [52] both achieve end-to-end inference, but at substantial cost: MOAI reports its end-to-end evaluation on a high-memory datacenter GPU (an H200), whereas ARION runs on CPU only. Also, their unamortized latency with batch size 256 reaches 10 and 16 hours per 128-token sequence, respectively, making the cost of deployment testing prohibitively high. Although their amortized throughput is competitive, the underlying packing algorithms do not support smaller batch sizes without substantial rework.

Given these barriers, we pivot to a basic row-packing implementation built upon and extended from an open-source codebase NEXUS-EndtoEnd [47]. While the simple row-packing homomorphic MM is not as competitive as the highly optimized alternatives, it does not affect the configuration of non-linear approximations and provides a fair, transparent baseline for evaluating different approximation choices.

Our system supports the full lifecycle from approximation search to end-to-end ciphertext model evaluation. Importantly, our approach is orthogonal to the design of efficient packing and MM; our search framework serves as a general-purpose tool that complements any of the above high-performance systems by providing automatic approximation configurations.

6 Concluding Remarks

The central insight of this paper is simple: existing FHE transformer frameworks fix approximation hyperparameters per function type and apply them uniformly across layers, leaving significant depth savings unrealized. ATLAS addresses this by treating approximation configuration as an optimization problem rather than a design choice, automatically discovering per-layer configurations that existing frameworks can adopt without modification. Applied to any hand-designed FHE transformer, ATLAS delivers a 35 percent reduction in multiplicative depth and a 25 percent reduction in inference latency, with negligible accuracy loss and no manual effort, in under one hour of search. These are gains that require no changes to the underlying cryptographic pipeline and only negligible accuracy tradeoff, obtainable entirely through automated post-processing.

We also demonstrate, for the first time, that this approach generalizes to vision transformers, extending the scope of private inference beyond the language models that prior work has exclusively targeted.

References

[1] Wei Ao and Vishnu Naresh Boddeti. 2024. AutoFHE: Automated adaption of CNNs for efficient evaluation over FHE. In *Proceedings of the 33rd USENIX Security Symposium*. 2173–2190.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[3] Youngjin Bae, Jaehyung Kim, Damien Stehlé, and Elias Suvanto. 2024. Bootstrapping small integers with CKKS. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 330–360.

[4] Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. 2021. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. 587–617.

[5] Zvika Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Annual Cryptology Conference*. 868–886.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)* 6, 3 (2014), 1–36.

[7] Mark Chen, Jerry Tworek, Heewoo Jun, et al. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374* (2021).

[8] Jung Hee Cheon, Wonhee Cho, Jaehyung Kim, and Damien Stehlé. 2023. Homomorphic multiple precision multiplication for CKKS and reduced modulus consumption. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 696–710.

[9] Jung Hee Cheon, Hyeongmin Choe, Minsik Kang, Jaehyung Kim, Seonghak Kim, Johannes Mono, and Taeyeong Noh. 2025. Grafting: decoupled scale factors and modulus in RNS-CKKS. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*. 1098–1112.

[10] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2018. Bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. 360–384.

[11] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2018. A full RNS variant of approximate homomorphic encryption. In *International Conference on Selected Areas in Cryptography*. 347–368. https://link.springer.com/chapter/10.1007/978-3-030-10970-7_16.

[12] Jung Hee Cheon, Guillaume Hanrot, Jongmin Kim, and Damien Stehlé. 2025. SHIP: A shallow and highly parallelizable CKKS bootstrapping algorithm. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 398–428.

[13] Jung Hee Cheon, Minki Hhan, Seungwan Hong, and Yongha Son. 2019. A Hybrid of Dual and Meet-in-the-Middle Attack on Sparse and Ternary Secret LWE. *IEEE Access* 7 (2019), 89497–89506.

[14] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*. Springer, 409–437.

[15] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology* 33, 1 (2020), 34–91.

[16] Wonhee Cho, Guillaume Hanrot, Taeseong Kim, Minje Park, and Damien Stehlé. 2024. Fast and accurate homomorphic softmax evaluation. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 4391–4404.

[17] Hyeongmin Choe, Jaehyung Kim, Damien Stehlé, and Elias Suvanto. 2025. Leveraging discrete CKKS to bootstrap in high precision. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*. 1083–1097.

[18] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168* (2021).

[19] Leo de Castro, Daniel Escudero, Adya Agrawal, Antigoni Polychroniadou, and Manuela Veloso. 2025. EncryptedLLM: Privacy-Preserving Large Language Model Inference via GPU-Accelerated Fully Homomorphic Encryption. In *Proceedings of the International Conference on Machine Learning*.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197. doi:10.1109/4235.996017

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 248–255.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.

[23] Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, and Wenguang Chen. 2023. Puma: Secure inference of llama-7b in five minutes. *arXiv preprint arXiv:2307.12533* (2023). https://arxiv.org/abs/2307.12533

[24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proceedings on the International Conference on Learning Representations*.

[25] Austin Ebel, Karthik Garimella, and Brandon Reagen. 2025. 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*. 734–749.

[26] Zahra Ghodsi, Akshaj Kumar Veldanda, Brandon Reagen, and Siddharth Garg. 2020. Cryptonas: Private inference on a relu budget. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 33. 16961–16971.

[27] Robert E Goldschmidt. 1964. *Applications of division by convergence*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[28] Seyda Nur Guzelhan, Lohit Daksha, Carlos Agulló Domingo, Gilbert Jonatan, John Kim, Jose L. Abellan, David Kaeli, and Ajay Joshi. 2026. ELLMo: Packing- and Depth-Aware Encrypted Transformer Inference. *Cryptology ePrint Archive, Paper 2026/198* (2026). https://eprint.iacr.org/2026/198

[29] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *Proceedings of the International Conference on Learning Representations*.

[30] Nandan Kumar Jha, Zahra Ghodsi, Siddharth Garg, and Brandon Reagen. 2021. Deepreduce: Relu reduction for fast private inference. In *Proceedings of the International Conference on Machine Learning*. PMLR, 4839–4849.

[31] Andrey Kim, Ahmet Can Mert, Anisha Mukherjee, Aikata Aikata, Maxim Deryabin, Sunmin Kwon, Hyung Chul Kang, and Sujoy Sinha Roy. 2024. Exploring the advantages and challenges of fermat NTT in FHE acceleration. In *Annual International Cryptology Conference*. Springer, 76–106.

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

[33] Eunsang Lee, Joon-Woo Lee, Jong-Seon No, and Young-Sik Kim. 2021. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Transactions on Dependable and Secure Computing* 19, 6 (2021), 3711–3727.

[34] Junghyun Lee, Eunsang Lee, Joon-Woo Lee, Yongjune Kim, Young-Sik Kim, and Jong-Seon No. 2023. Precise approximation of convolutional neural networks for homomorphically encrypted data. *IEEE Access* 11 (2023), 62062–62076.

[35] Lawrence Lim, Vikas Kalagi, Divyakant Agrawal, and Amr El Abbadi. 2025. Tricycle: Private Transformer Inference with Tricyclic Encodings. *Cryptology ePrint Archive, Paper 2025/1200* (2025). https://eprint.iacr.org/2025/1200

[36] Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. 2025. BumbleBee: Secure Two-party Inference Framework for Large Transformers. In *Proceedings of the 32nd Annual Network and Distributed System Security Symposium*.

[37] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (Eds.). Association for Computational Linguistics, Portland, Oregon, USA, 142–150. https://aclanthology.org/P11-1015/

[38] Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. 2025. THOR: Secure transformer inference with homomorphic encryption. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*. 3765–3779.

[39] Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. 2024. BOLT: Privacy-Preserving, Accurate and Efficient Inference for Transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*. 4753–4771.

[40] Dongjin Park, Eunsang Lee, and Joon-Woo Lee. 2025. 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*. 11090–11111.

[41] Hanjun Park, Byeong-Seo Min, Jiheon Woo, Min-Wook Jeong, Jongho Shin, Yongwoo Lee, Young-Sik Kim, and Yongjune Kim. 2026. Efficient Softmax Reformulation for Homomorphic Encryption via Moment Generating Function. *arXiv preprint arXiv:2602.01621* (2026).

[42] Hongyuan Qu and Guangwu Xu. 2023. Improvements of homomorphic secure evaluation of inverse square root. In *International Conference on Information and Communications Security*. Springer, 110–127.

[43] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[44] Donghwan Rho, Taeseong Kim, Minje Park, Jung Woo Kim, Hyunsik Chae, Ernest K Ryu, and Jung Hee Cheon. 2025. Encryption-Friendly LLM Architecture. In *Proceedings of the International Conference on Learning Representations*.

[45] SEAL. 2020. Microsoft SEAL is an easy-to-use and powerful homomorphic encryption library. https://github.com/Microsoft/SEAL

[46] Wenting Zheng Srinivasan, PMRL Akshayaram, and Popa Raluca Ada. 2020. Delphi: A cryptographic inference service for neural networks. In *Proceedings of the 29th USENIX Security Symposium*. 2505–2522.

[47] Zhengyuan Su. 2025. Reproducing NEXUS with Phantom bootstrapping. https://github.com/timzsu/NEXUS-End2End

[48] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023). https://arxiv.org/abs/2302.13971

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[50] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings on the International Conference on Learning Representations.*
[51] Hao Yang, Shiyu Shen, Wangchen Dai, Lu Zhou, Zhe Liu, and Yunlei Zhao. 2024. Phantom: A cuda-accelerated word-wise homomorphic encryption library. *IEEE Transactions on Dependable and Secure Computing* 21, 5 (2024), 4895–4906.
[52] Linhan Yang, Jingwei Chen, Wangchen Dai, Shuai Wang, Wenyuan Wu, and Yong Feng. 2025. ARION: Attention-Optimized Transformer Inference on Encrypted Data. *Cryptology ePrint Archive, Paper 2025/2271* (2025). https://eprint.iacr.org/2025/2271
[53] Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. In *Proceedings of the Advances in Neural Information Processing Systems.*
[54] Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen-jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. 2025. Secure transformer inference made non-interactive. In *Proceedings of the 32nd Annual Network and Distributed System Security Symposium.*
[55] Linru Zhang, Xiangning Wang, Jun Jie Sim, Zhicong Huang, Jiahao Zhong, Huaxiong Wang, Pu Duan, and Kwok-Yan Lam. 2026. MOAI: Module-optimizing architecture for non-interactive secure transformer inference. In *Proceedings of the 14th International Conference on Learning Representations.*
[56] Itamar Zimerman, Moran Baruch, Nir Drucker, Gilad Ezov, Omri Soceanu, and Lior Wolf. 2024. Converting transformers to polynomial form for secure inference over homomorphic encryption. In *Proceedings of the International Conference on Machine Learning*. 62803 – 62814.
[57] E. Zitzler and L. Thiele. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* 3, 4 (1999), 257–271. doi:10.1109/4235.797969

A Hypervolume

For a set of solutions $\mathcal{S}$ and a reference point $\boldsymbol{r}$, the hypervolume [57] $\mathrm{HV}(\mathcal{S}, \boldsymbol{r})$ measures the portion of the objective space that is dominated by at least one solution in $\mathcal{S}$ and, dominates $\boldsymbol{r}$ at the same time. As shown in Figure 11, in the two-dimensional minimization case with f_1 vs. f_2 (where smaller values are better, e.g., MAE vs. multiplicative depth), this corresponds to the total area of the union of the rectangles that have a solution point and $\boldsymbol{r}$ as opposite corners. Given two solutions $\boldsymbol{a}$ and $\boldsymbol{b}$ (both better than $\boldsymbol{r}$ in all objectives), the hypervolume is exactly the area covered by the two rectangles defined by $(\boldsymbol{a}, \boldsymbol{r})$ and $(\boldsymbol{b}, \boldsymbol{r})$, without double-counting their overlap. Intuitively, *a larger hypervolume is better*, which indicates that the solution set covers more of the region between the solutions and the reference point, i.e., closer to the true Pareto front.

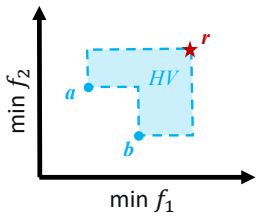


Figure 11: Illustration of the hypervolume in two-dimensional minimization case.

B FHE Inference System Implementation

We develop our GPU-accelerated FHE inference system based on our extensions of an open-source codebase, i.e., NEXUS-End2End [47]. Our experiment platform is a modified version RTX 4090 with 48GB. The encoder of BERT, decoder of LLaMA, and vision encoder of ViT can be run in constrained 40GB GPU based on our FHE inference system. As a result, our system's reliance on GPU memory is significantly lower than that of existing frameworks [38, 54, 55]. We have also integrated `LibTorch` into our library, allowing us to build models directly by reading Torch weights, rather than pre-storing large amounts of encoded plaintext weights. In our system, we can pre-compute the weights packing (this requires ∼200 GB RAM for BERT), or dynamically pack the weights in runtime.

B.1 Polynomial Evaluation with Paterson–Stockmeyer Method

The Chebyshev coefficients of exponential function, inverse square root, GELU, and SiLU used in this work were generated directly by the `chebyshevform` function in `Sollya 8.0`, with the intermediate precision set to 1024 bits. After generation, all coefficients with absolute values smaller than 10^{-14} were set to zero. The resulting coefficients were stored in scientific notation, with 40 decimal digits retained for each coefficient.

Also, we implement the Paterson-Stockmeyer (PS) polynomial evaluation method in our FHE inference system and apply it uniformly to both power-basis and Chebyshev-basis polynomials. Let $\deg(p)$ be the polynomial degree. For a power-basis polynomial the multiplicative depth is $\lceil \log_2(\deg(p) + 1) \rceil$; for a Chebyshev-basis polynomial, a single additional linear transformation is required on the input, yielding a depth of $\lceil \log_2(\deg(p) + 1) \rceil + 1$.

B.2 Details of Iterative Softmax in ATLAS

Following Cho et al. [16], the polynomial degree of exponential approximation is fixed to 15; the input interval for generating each degree-$(2^{p_j^{(i)}} - 1)$ polynomial depends on the iteration index j: $(0.085, 256)$ for j=1, $(0.003, 1.238)$ for $j \in \{2, 3, 4\}$, and $(0.0036, 1.0816)$ for j=5, as shown in Table 2.

Table 2: Search space for the iterative softmax approximation. The exponential function is approximated by a 15-degree polynomial. The inverse square root in iteration j uses a Chebyshev polynomial of degree $2^{p_j} - 1$, where $p_1 \in [1, 7]$ (non-zero) and $p_2, \ldots, p_5 \in [0, 7]$. A zero p_j omits that iteration.

	iter. j	degree	input range
$\exp(x)$	all	15	$(-8,\ 0)$
$1/\sqrt{x}$	1	$2^{p_1} - 1$	$(0.085,\ 256)$
	2	$2^{p_2} - 1$	$(0.003,\ 1.238)$
	3	$2^{p_3} - 1$	$(0.003,\ 1.238)$
	4	$2^{p_4} - 1$	$(0.003,\ 1.238)$
	5	$2^{p_5} - 1$	$(0.0036,\ 1.0816)$

B.3 Per-Layer Softmax Max-Constant

In FHE, dynamically evaluating $\max_i x_i$ in the softmax exponentiation is prohibitively expensive. Following prior work [54], each per-layer maximum is replaced by a fixed constant in searching

and FHE inference, which is calibrated on a single real-data sample using the iterative softmax [16]-based baseline.

For BERT-Base ($L = 12$), one IMDB sentence is used, yielding the 12 integer constants:

$$\boldsymbol{c}_{bert} = [\,13, 18, 34, 16, 12, 13, 13, 12, 13, 12, 15, 10\,], \tag{13}$$

for ViT-Base ($L = 12$), one ImageNet-1K validation image gives:

$$\boldsymbol{c}_{vit} = [\,108, 32, 22, 14, 15, 15, 15, 13, 15, 13, 13, 9\,], \tag{14}$$

for LLaMA3-8B ($L = 32$), the same IMDB procedure yields:

$$\begin{aligned}\boldsymbol{c}_{llama} = [\,&22, 7, 5, 6, 6, 9, 9, 8, 9, 10, 9, 7, 6, 9, 10, 9,\\ &9, 9, 13, 11, 12, 10, 8, 13, 9, 9, 10, 9, 12, 11, 11, 89\,].\end{aligned} \tag{15}$$

These vectors are injected directly into our softmax computation of all configurations in cleartext simulated evaluation and FHE inference.

B.4 Homomorphic Matrix Multiplication

For PCMM, we follow the BSGS-based implementation used in prior work [47]. Since each ciphertext has $32768 = 128 \times 256$ slots, we use ct.$(128, 128)$×pt.$(128, 128)$ as the basic PCMM operator, which fits within one ciphertext. The ciphertext is first expanded into 16 baby-step rotations. For each of the 16 giant steps, the corresponding plaintext diagonals are encoded and multiplied with the baby-step ciphertexts, and the partial sums are then rotated and accumulated to the final output. Overall, one ct.$(128, 128)$×pt.$(128, 128)$ PCMM requires $16 \times 16 = 256$ plaintext-ciphertext multiplications and 32 rotations, including 16 baby-step rotations and 16 giant-step rotations. The computation consumes one level.

For higher-dimensional PCMM, we decompose the computation into independent ct.$(128, 128)$×pt.$(128, 128)$ block operations and combine the partial results using homomorphic rotations and additions. For ct.(A, B)×pt.(C, D), this gives $(A \cdot B \cdot C)/(2 \cdot 128^3)$ basic block operations, while the extended PCMM still consumes only one level. We also extend the codebase [47] to support BSGS-based CCMM using rotations and precomputed masks. A ct.$(128, 128)$× ct.$(128, 128)$ CCMM consumes three levels due to masking and ciphertext-ciphertext multiplication. Unlike PCMM, this CCMM implementation is not dimension-generic, since each new matrix shape requires separately designed masks.

C Non-linear Approximation in NEXUS

C.1 Softmax in NEXUS

Limit Exponential Function. The NEXUS adopts the limit approximation $p(x) = (1 + x/2^r)^{2^r}, r = 7$ to approximate the exponential function $\exp(x)$, consuming 8 multiplicative depths.

Goldschmidt Inverse. The NEXUS fixes Goldschmidt steps $\gamma = 4$, so the multiplicative depth is 5. Our reproduction shows that, under the 1% relative-error criterion, the effective input range of this inverse module is limited to $[0.13, 1.87]$. For the complete exp-plus-inverse Softmax subroutine, the measured effective logits range is approximately $[-2.19, 0]$.

C.2 LayerNorm in NEXUS

The NEXUS adopts Newton + Goldschmidt invsqrt in LayerNorm, as shown in Algorithm 2, it computes $1/\sqrt{x}$ using an initial linear approximation, followed by $\nu = 4$ steps of the Newton-Raphson method ($y_{i+1} = 1.5\, y_i - 0.5\, x\, y_i^3$) and $\gamma = 2$ steps of the Goldschmidt iteration. In our reproduction, the invsqrt approximation achieves about 1% relative error when its input x, i.e., the LayerNorm variance, lies in the empirical range $[2.0, 954.3]$.

Algorithm 3: Goldschmidt $\textsc{Inv}(x; \gamma)$ in NEXUS

Input: Input x; iteration count γ
Output: $r \approx x^{-1}$

$r \leftarrow 2 - x$;
for $i \leftarrow 1$ **to** γ **do**
  $r \leftarrow r \cdot (1 + (1 - x)^2)$);
end
return r;

C.3 GELU in NEXUS

The GELU in NEXUS is fitting the Tanh-based GELU by a piecewise polynomial function, as shown in Equation (16).

$$\mathrm{GELU}(x) = \begin{cases} 0 & \text{if } x \le -4 \\ p_k(x) & \text{if } -4 < x \le -1.95 \\ q_d(x) & \text{if } -1.95 < x \le 3 \\ x & \text{if } x > 3 \end{cases} \tag{16}$$

where $p_k(x)$ is k-degree polynomial, and $q_d(x)$ is d-degree polynomial, and $k = 3, d = 6$. For the selection function, employing a sign function approximated by minimax composition [33], i.e., $\mathrm{AppGELU}(x) = b_0 \cdot 0 + b_1 p(x) + b_2 q(x) + b_3 x$, where $P(x)$ and $Q(x)$ are high-degree polynomials, and $b_i, i = 0, 1, 2, 3$ are segments control hyperparameters. For the polynomials $p_k(x)$ and $q_d(x)$, the coefficients come from the NEXUS codebase. Empirically, this piecewise GELU approximation satisfies the 1% relative-error criterion only on the effective input range $[-5, 5]$.

D Non-linear Approximation in THOR

D.1 Softmax in THOR

THOR first approximates $\mathrm{Softmax}(x/\delta_2)$ by scaling the input by $1/(\delta_1\delta_2)$, applying a polynomial exponential approximation, raising the result to the δ_1-th power, and normalizing it with a Goldschmidt-based inverse using adaptive relaxation. It then applies $\log \delta_2$ square-and-normalize steps to recover $\mathrm{Softmax}(x)$. In our reproduction, we implement the THOR setting $(\delta_1, \delta_2) = (2, 2)$. We empirically find that its effective encrypted-input range is approximately $[-26, 22]$, within which the reproduced module achieves a relative error of about 10^{-3} compared with cleartext softmax.

D.2 LayerNorm in THOR

THOR rewrites LayerNorm into a scaled form using $nx - \sum x$ as the numerator and a denominator term derived from $\sum x$ and $\sum x^2$. It then applies a Goldschmidt-based inverse-square-root approximation with adaptive relaxation and uses the result to normalize the scaled numerator.

Algorithm 4: LayerNorm via Iterative Invsqrt in THOR

Input: Input x; affine coefficients w, b, iteration T
Output: $y \approx \mathrm{LayerNorm}(x)$

Compute $\sum x$ and $\sum x^2$ by ciphertext rotations and additions;
$m \leftarrow nx - \sum x$;
$v \leftarrow n\sum x^2 - (\sum x)^2 + \tilde{\varepsilon}$;
$a_0 \leftarrow v,\ b_0 \leftarrow 1$;
for $t \leftarrow 0$ **to** $T-1$ **do**
  $a_{t+1} \leftarrow k_t a_t (3 - k_t a_t)^2/4$;
  $b_{t+1} \leftarrow \sqrt{k_t}\, b_t (3 - k_t a_t)/2$;
end
$y \leftarrow w \cdot (m \cdot b_T) + b$;
return y;

D.3 GELU in THOR

THOR approximates GELU through the tanh-based formulation, where the input x is first scaled as $z = x/64$ and $\tanh(z)$ is approximated by a composed polynomial $p_2(p_1(z))$ with two degree-15 polynomials. $p_1(\cdot)$ is obtained by least-squares fitting, and $p_2(\cdot)$ is used for residual refinement. The final GELU value is then recovered as $64z \cdot (1 + p_2(p_1(z)))/2$. In our reproduction, we empirically identify $[-70, 116]$ as the effective input range, where the reproduced GELU module keeps the absolute error below 1%.

E Surrogate Models of LLaMA

Figure 12 (top right) reports Kendall τ and the resulting speed-up as a function of the proxy subset size for LLaMA3-8B. As the subset size increases, τ approaches 1. A subset of 10 IMDB sentences, evaluated on eight NetworkProblem configurations, yields $\tau \approx 0.905$ with a ~454× speed-up (Figure 12, bottom right), which is sufficient for reliable ranking. Thus, ATLAS uses a subset size of 10 for LLaMA3-8B in practice.

Figure 12 (top left) plots τ between the early layer-i MAE and the last-layer MAE against the corresponding speed-up. As the layer index increases, τ similarly approaches 1. As illustrated for one example (Figure 12, bottom left), using the layer-9 MAE as the early-layer proxy gives $\tau = 0.924$ with an ~8.5× speed-up. Accordingly, ATLAS adopts the layer-9 MAE as the proxy for LayerProblem in LLaMA3-8B.

F Extended Experimental Results

F.1 Accuracy–Latency Pareto of BERT

Figure 13 shows the accuracy–latency Pareto fronts discovered by ATLAS on SST-2, RTE, and QNLI. The IS baseline is the same, operating at 1107.5 s with accuracies 93.2% (SST-2), 69.0% (RTE), and 91.3% (QNLI). The fronts exhibit the same steep shape observed in the accuracy–depth view as discussed in Section 4.2.1: a modest latency budget yields configurations that match or exceed IS in accuracy. For example, on SST-2, S3 matches IS at 93.0% accuracy with 23.6% lower latency (846.5 s); on RTE, R3 surpasses IS by 1.0 pp (70.0% vs. 69.0%) with 25.6% lower latency (824.5 s); on QNLI, Q3 matches IS accuracy (91.2%) with 23.7% lower latency (844.8 s).

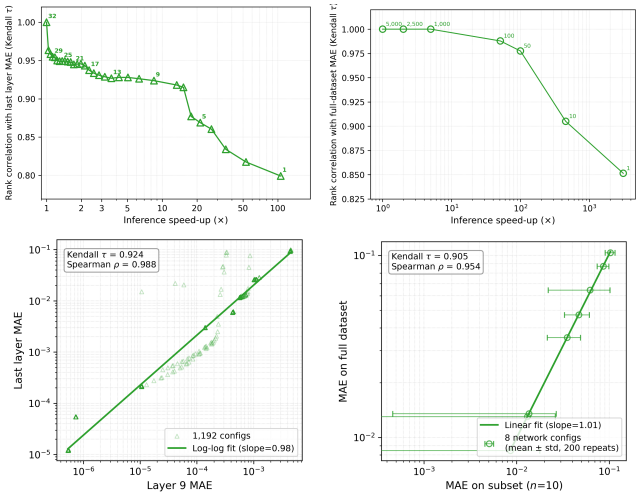


Figure 12: MAE at an early layer as a surrogate for MAE at the last layer, and the correlation between the MAE on the full and subset of the dataset, are shown on LLaMA.

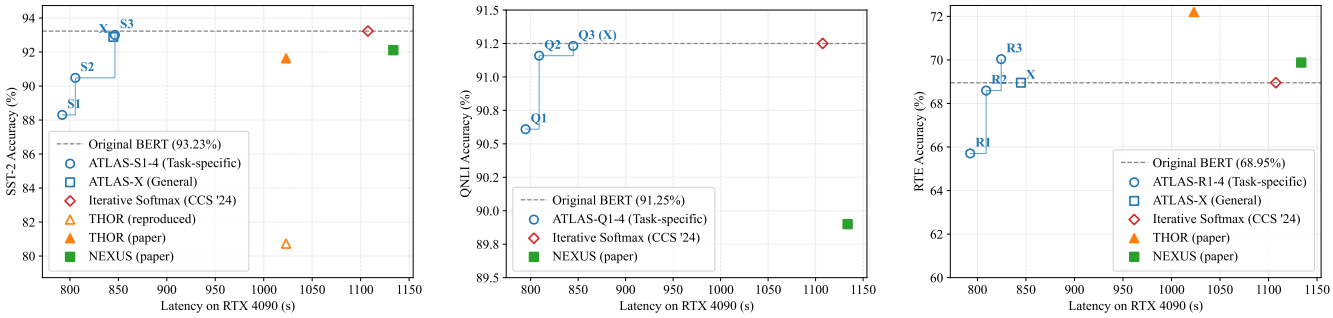
Table 3 summarizes these three representative configurations. As before, the best configuration differs by task—S3 for SST-2, Q3 for QNLI, R3 for RTE—reflecting that different tasks have different preferences for the approximation configuration. Notably, the configuration Q3 (denoted X) performs well across all three tasks, offering a balanced choice for multi-task deployment; X is also the same configuration identified in the accuracy–depth Pareto front. For **RQ1**, these results confirm that ATLAS automatically discovers configurations that advance the accuracy–latency frontier over the hand-tuned baseline in encoder-only architectures. Figure 14 visualizes the per-layer configurations for reference.

Table 3: Accuracy–Latency Pareto points for BERT-Base (L=12).

	Mul Depth	#B	Lat. (s)	Accuracy (%)		
				SST2	QNLI	RTE
BERT-Base [22]	-	-	-	93.2	91.3	69.0
IterSoftmax [16]	1356	588	1108	93.2	91.3	69.0
ATLAS-S1	854	390	792	88.3	90.5	65.3
ATLAS-R1	840	390	793	85.2	90.5	65.7
ATLAS-Q1	847	390	795	88.2	90.6	65.0
ATLAS-S2	864	393	806	90.5	89.7	66.4
ATLAS-R2 (Q2)	879	393	809	87.4	91.2	68.6
ATLAS-R3	888	402	825	85.8	90.8	**70.0**
ATLAS-Q3 (X)	919	408	845	92.9	**91.2**	68.9
ATLAS-S3	918	411	847	**93.0**	90.6	68.6

F.2 Accuracy–Latency Pareto of ViT

Figure 15 shows the accuracy–latency Pareto front discovered by ATLAS on ImageNet-1K. The IS baseline achieves 80.32% top-1 accuracy at 2944.8 s. The front contains two configurations V1 and



(a) SST-2 (b) QNLI (c) RTE

Figure 13: Accuracy–Latency Pareto fronts discovered by ATLAS on three tasks of BERT. Each point represents a non-dominated λ configuration, plotted in the space of FHE inference latency (RTX 4090) vs. downstream task accuracy. The iterative softmax baseline is shown as a reference. Dashed horizontal lines indicate the accuracy of the underlying cleartext model.

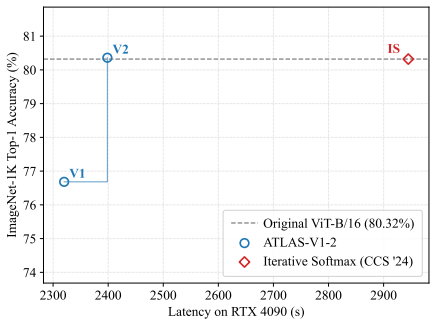


Figure 15: Accuracy–Latency Pareto front discovered by ATLAS on ImageNet-1K of ViT-Base.

Table 4: Accuracy–Latency Pareto points for ViT-Base (L=12).

	Mul Depth	#B	Lat. (s)	Top-1 Accuracy (%)
				ImageNet-1K
ViT-Base [24]	-	-	-	80.32
IterSoftmax [16]	1356	1608	2944.8	80.32
ATLAS-V1	907	1206	2320.1	76.68
ATLAS-V2	966	1230	2398.5	**80.36**

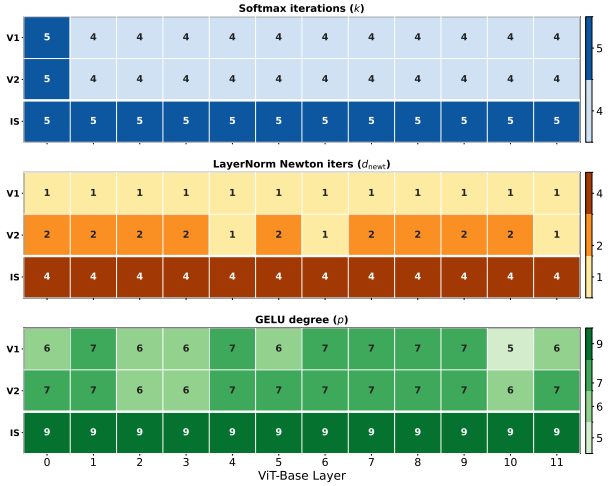


Figure 16: Per-layer visualization of the Accuracy–Latency Pareto configs of ATLAS for ViT-Base on ImageNet-1K shown in Figure 15.

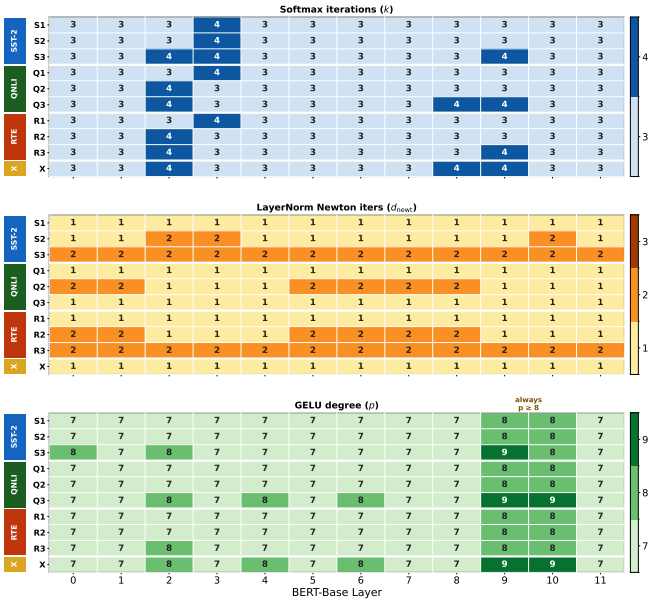


Figure 14: Per-layer visualization of the Accuracy–Latency Pareto configs of ATLAS for BERT shown in Figure 13.

V2 (Table 4). The best configuration V2 surpasses IS with 80.36% accuracy while reducing latency by 18.5% (2398.5 s). For **RQ1**, this confirms that ATLAS automatically discovers per-layer approximation recipes that improve the accuracy–latency Pareto front of ViT. Figure 16 visualizes the per-layer configurations for reference.