CTRLFORMER: EFFICIENT NON-INTERACTIVE SECURE TRANSFORMER INFERENCE WITHOUT FEATURE EXPLOSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Fully homomorphic encryption (FHE) enables non-interactive, secure transformer inference. Despite numerous efforts to optimize FHE algorithms and replace non-polynomial functions with FHE-friendly alternatives, existing approaches still suffer from feature explosion and low inference efficiency. To address these issues, we propose CtrlFormer, a novel FHE-based transformer inference framework. Firstly, we propose a bounded, FHE-friendly Softmax alternative that avoids the feature explosion present in many existing Softmax replacements. Secondly, we unify activation evaluation and bootstrapping into a single operation via functional bootstrapping, eliminating the need for separate activation evaluations that typically consume about ten multiplicative levels. Finally, by substantially reducing the multiplicative depth, CtrlFormer supports a smaller ciphertext modulus, enabling faster homomorphic matrix multiplications. CtrlFormer achieves an amortized time of 1.8 minutes per input for the revised BERT-base model on a single GPU, achieving a significant reduction in computational cost while improving performance on the GLUE benchmark compared to prior works.

1 INTRODUCTION AND RELATED WORK

Fully homomorphic encryption (FHE) (Gentry, 2009), which allows direct and non-interactive computation on encrypted data, offers a promising solution for preserving data privacy in AI-as-a-Service (AIaaS) scenarios such as transformer inference on cloud platforms. Compared to secure multi-party computation (MPC)-based methods (Pang et al., 2024; Dong et al., 2025; Lu et al., 2023; Li et al., 2023; Zeng et al., 2025), which incur substantial interactive communication cost and require the client to remain online throughout the whole pipeline, FHE enables outsourcing inference in a non-interactive manner. However, the substantial computational overhead of homomorphic evaluation remains a major obstacle, making efficiency the primary bottleneck for practical FHE-based non-interactive inference.

Recently, several works have explored efficient FHE evaluation algorithms to improve inference efficiency without modifying the transformer architecture. NEXUS (Zhang et al., 2025a) (NDSS'25) initiated this line of research by proposing FHE evaluation algorithms for core transformer operations, including matrix multiplication, Softmax, Layernorm, and GELU. THOR (Moon et al., 2025) (CCS'25) achieved end-to-end encrypted inference and reported a latency of 10.43 minutes per input for a BERT-base model. ARION (Yang et al., 2025) further optimized attention head evaluation and applied batching to reduce amortized inference time, achieving 3.75 minutes per input for the same model. In these approaches, evaluating activation functions like Softmax, Layernorm, and GELU is particularly challenging. For instance, the division operations in Softmax and Layernorm are typically implemented using the Goldschmidt method, which requires more than 10 multiplicative levels to support a reasonable predetermined input range and precision. This results in large ciphertext moduli and frequent bootstrapping operations[1], and substantially increases computational overhead.

[1] Bootstrapping is an essential and expensive part to refresh ciphertexts. It is required whenever the remaining multiplicative levels are insufficient for subsequent operations. Please refer to Section 2.2 and Appendix C for details.

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

To further improve inference efficiency, some works adapt activation functions in transformer architectures to polynomial or other arithmetic-friendly alternatives. Zimerman et al. (2024b) proposed the first polynomial transformer model. A state-of-the-art FHE-friendly transformer architecture was proposed by Powerformer (Park et al., 2025), which replaces Softmax and Layernorm in the BERT-base model with power and linear functions, achieving 5.74 minutes per input on a single GPU with no accuracy loss. MOAI (Zhang et al., 2026) applies batching techniques to reduce the amortized inference cost and reports an average latency of 2.54 minutes per input when deployed on Powerformer's architecture. Despite these efforts to reduce latency, two major challenges remain in practical deployment of FHE-based transformer inference.

(1) Feature Explosion in FHE-friendly alternatives. Although alternatives like Park et al. (2025) consume much fewer multiplicative levels and reduce the number of required bootstrapping operations, their unbounded output range of the power-function-based Softmax replacement can cause feature explosion where activations grow rapidly and compound across layers. Feature explosion can lead to unpredictable behavior during transformer inference and result in reduced output quality, incoherent or repetitive generations, and increased hallucinations (Ji et al., 2023; Suresh et al., 2025).

(2) High multiplicative levels and low efficiency. Despite substantial efforts to optimize inference efficiency in FHE-enabled models, existing works still require a large ciphertext modulus to secure enough multiplicative depth (more than 11 levels prior to bootstrapping) for evaluating activation functions such as GELU. This dependence arises because accurate polynomial approximations of non-polynomial activations demand deep multiplicative circuits under FHE (Park et al., 2025; Cho et al., 2024). Consequently, this strategy reduces throughput and increases latency in real-world inference.

Our Contribution. Our proposed CtrlFormer is an FHE-based secure transformer that mitigates feature explosion and boosts inference efficiency. We summarize our contributions as follows:

(1) Bounded FHE-friendly Activation. We point out that FHE-friendly alternatives are not limited to polynomials, and introduce *XControl*, a bounded and FHE-friendly activation that replaces Softmax to mitigate feature explosion. Unlike prior polynomial-based replacements, XControl ensures bounded outputs and numerical stability. To our knowledge, this is the first exploration of non-polynomial alternatives in FHE-friendly transformer architectures.

(2) Reduced Multiplicative Levels and Faster Inference. CtrlFormer reduces the required multiplicative levels by adapting CKKS functional bootstrapping, which combines element-wise function evaluation and ciphertext refreshing into a single step. This eliminates the need for separate activation computations, which would otherwise consume approximately 10 multiplicative levels. [2] In addition, by consuming fewer multiplicative levels, CtrlFormer allows for much smaller ciphertext modulus, which (i) enables faster homomorphic matrix multiplications and improves overall computational efficiency, and (ii) reduces memory usage of ciphertexts, making FHE-based inference more scalable and practical. For comparison, in our CtrlFormer only 5 multiplicative levels are consumed before entering the functional bootstrapping process; NEXUS and MOAI use 20 levels, ARION uses 14 levels, THOR uses 13 levels, and Powerformer uses 11 levels prior to bootstrapping.

Combined with a series of HE algorithm optimizations, we achieve an amortized inference time of 1.8 minutes per input for the BERT-base model on a single GPU. Compared to the state-of-the-art FHE-friendly transformer architecture Powerformer (Park et al., 2025), CtrlFormer addresses the feature explosion issue, and attains a 68.5% reduction in inference time under the same environment while achieving higher GLUE (Wang et al., 2018) scores.

Related works on secure transformer inference. There are two main lines of work in secure transformer inference: (1) *Interactive solutions*, which require server–client interaction during inference; (2) *Non-interactive solutions*, where the client can go offline after uploading encrypted inputs. In interactive settings, secure multi-party computation (MPC) is often combined with FHE to reduce computational overhead through communications. THE-X (Chen et al., 2022) introduced the first secure inference protocol for the BERT-tiny model and was subsequently extended and improved by later works (Pang et al., 2024; Lu et al., 2023; Dong et al., 2025; Jha & Reagen, 2025; Zeng et al., 2025; Zhang et al., 2025c; Xu et al., 2025). However, interactive inference may require

[2] NEXUS (Zhang et al., 2025a), THOR (Moon et al., 2025), and MOAI (Zhang et al., 2026) require 14, 11, and 7 multiplicative levels, respectively, for GELU.

large-scale communication, and data owners have to be online during the computation (Rho et al., 2024). NEXUS (Zhang et al., 2025a) proposed the first non-interactive FHE-based solution, which was extended by THOR (Moon et al., 2025) to achieve end-to-end secure inference for BERT-base. Other works (Zhang et al., 2026; 2025b; Yang et al., 2025; Zimerman et al., 2024b;a; Park et al., 2025) explore different methods to improve efficiency such as batching and architectural adaptations. Another line of work (Avitan et al., 2025; Lim et al., 2025; Rho et al., 2025) focuses on next-token generation with encrypted input such as the Argmax function.

Related works on Softmax, Layernorm and GELU in FHE. One line of work focuses on *accurately approximating these non-polynomial functions*. NEXUS (Zhang et al., 2025a) and THOR (Moon et al., 2025) apply various approximation techniques to evaluate Softmax, Layernorm, and GELU. However, these approaches suffer from low computational efficiency. To address this, the other line of work focuses on *replacing them with FHE-friendly alternatives*. MPCFormer (Li et al., 2023) replaces GELU and exponential in Softmax with quadratic functions, but incurs notable accuracy degradation on BERT-base. SecFormer (Luo et al., 2024) improves accuracy by using segmented polynomials and Fourier series to approximate GELU, and introduces a deflation technique for evaluating Layernorm. Gaussian kernels are used to avoid division in Rho et al. (2024), though results are limited to shallow BERT-style models. Park et al. (2026) reformulates the Softmax denominator using the cumulant-generating function but does not evaluate the approach in an end-to-end FHE pipeline. Zimerman et al. (2024b); Park et al. (2025); Zimerman et al. (2024a) replace non-polynomial components with polynomials or power functions to improve efficiency. However, these unbounded alternatives can lead to feature explosion, especially in large models, as observed in Powerformer.

2 PRELIMINARIES

In this work, we use bold lowercase letters for *row* vector $\mathbf{v} \in \mathbb{R}^n$ where $\mathbf{v} = [v_0, v_1, ..., v_{n-1}]$. A *column* vector is $\mathbf{v}^\intercal$. We use $[v_i]_{i\in[n]}$ to represent the vector $\mathbf{v}$ where $[n]$ is the set $\{0, 1, ..., n-1\}$. Let a matrix $A \in \mathbb{R}^{m\times d}$ where $A = (a_{ij})_{i\in[m],j\in[d]}$. $\mathbf{a_j}^\intercal$ is the j-th column of A: $A = (\mathbf{a_0}^\intercal \parallel \cdots \parallel \mathbf{a}_{d-1}^\intercal)$.

2.1 FHE-BASED NON-INTERACTIVE SECURE TRANSFORMER INFERENCE

We focus on the BERT-base model, which is widely used in prior FHE-based transformer inference studies (Zhang et al., 2025a; Moon et al., 2025; Park et al., 2025; Zhang et al., 2026; Yang et al., 2025). The core computations include matrix multiplication, Softmax, Layernorm, and GELU. Additional details are provided in Appendix A. In FHE-based transformer inference, the client encrypts the input and sends it to the cloud, which evaluates the model directly on encrypted data without interacting with the client and returns the encrypted result. Consequently, the cloud learns nothing about either the client's input or the inference output. Same as prior works (Zhang et al., 2025a; Moon et al., 2025; Zhang et al., 2026; Park et al., 2025), we adopt the *honest-but-curious* setting: both the server and the client honestly follow the protocol but may attempt to infer additional information.

2.2 FULLY HOMOMORPHIC ENCRYPTION - CKKS SCHEME

The Cheon-Kim-Kim-Song (CKKS) scheme (Cheon et al., 2017) is an FHE scheme that supports approximate arithmetic operations over complex-valued vectors in the SIMD (Single Instruction, Multiple Data) style. Let N be an integer that is a power of 2. The CKKS scheme supports SIMD-style computation by encoding $N/2$ messages into a single ciphertext. Let $\mathbf{v}, \mathbf{v}'$ be two vectors in $\mathbb{C}^{N/2}$. Let $\mathrm{ct} = \mathrm{Enc}(\mathbf{v})$ and $\mathrm{ct}' = \mathrm{Enc}(\mathbf{v}')$. The CKKS scheme contains:
- $\mathrm{Add}(\mathrm{ct}, \mathrm{ct}')$: Ciphertext of $\mathbf{v} + \mathbf{v}'$.
- $\mathrm{Mult}(\mathrm{ct}, \mathrm{ct}')$: Ciphertext of the Hadamard product $\mathbf{v} \odot \mathbf{v}'$.
- $\mathrm{PlainMult}(\mathrm{ct}, \mathbf{v}')$: Ciphertext of $\mathbf{v} \odot \mathbf{v}'$.
- $\mathrm{Rot}_k^{\mathrm{HE}}(\mathrm{ct})$: Ciphertext of $[v_k, ..., v_{N/2-1}, v_0, ..., v_{k-1}]$.

Functional Bootstrapping. Let Q be a product of distinct primes: $Q = \prod_{i=0}^{L} q_i$, where L is called the *level* of the ciphertext. A ciphertext from level l drops to $l-1$ after a homomorphic multiplication. During inference we must ensure that the remaining level is above 0 otherwise the decryption will fail. Bootstrapping (Cheon et al., 2018) enables continued evaluations as it can

"refresh" the ciphertext and restore the remaining levels. Bootstrapping includes: (1) Raise the ciphertext modulus(ModRaise); (2) Convert the message from coefficient domain to the slot domain (CoeffToSlot); (3) Evaluate an interpolation of the modulo function; (4) Convert the message back to the coefficient domain (SlotToCoeff). The idea of functional bootstrapping is to merge ciphertext refreshing and element-wise function evaluation into a single operation (Ducas & Micciancio, 2015; Lu et al., 2021; Liu & Wang, 2023). Recently, a line of research (Alexandru et al., 2025; Dumezy et al., 2025; Yang et al., 2026) extended this concept to the CKKS scheme, enabling the evaluation of functions during the CKKS bootstrapping process. Assume that the input ciphertext ct' is at level l', functional bootstrapping can raise the ciphertext to level L'_0 and simultaneously evaluate a function on encrypted messages. Similar to bootstrapping, CKKS functional bootstrapping also consists of 4 steps: (1) SlotToCoeff; (2) ModRaise; (3) CoeffToSlot; (4) Evaluate a polynomial from interpolation to remove the added redundancy and compute a predetermined function at the same time. Additional details are provided in Appendix C.

3 Architecture of CtrlFormer

Many existing Transformer-based FHE-based studies (Park et al., 2025; Zimerman et al., 2024b) have concentrated on effectively adapting activation functions to facilitate stable and computationally efficient homomorphic operations. However, they suffer from the serious problem of feature explosion, leading to increased computational complexity and longer training times. Worse still, high dimensionality makes interpretation difficult, obscuring the relationships between features and predictions. To tackle this issue, we propose a novel paradigm for distilling FHE-friendly models from a teacher model, allowing for stable training without the risk of feature explosion. We propose the *X-Control Function* (XControl) as a replacement for Softmax, and the *Dynamic Shift Function* (DSF) as an alternative to conventional Layernorm (LN). Our proposed alternatives mitigate feature explosion and incur low computational overhead when evaluating over encrypted data.

X-Control function. The development of functional bootstrapping, which merges ciphertext refreshing and element-wise function evaluation into a single operation with negligible additional cost, suggests that non-polynomial functions can also be considered FHE-friendly, provided they are applied element-wise. Therefore, the space of FHE-compatible alternatives is not restricted to polynomials. In our design of XControl, we retain the $\exp(\cdot)$ component of Softmax. Additionally, we constrain the exponent to be negative to prevent excessively large outputs, thereby mitigating potential numerical instability. Additionally, this design promotes gradient stability, allowing the model to update weights more uniformly during the learning process, which further lowers the likelihood of feature explosion. Formally, XControl is as:

$$\mathrm{XControl}(x) = \frac{A_s(x)}{\max_i \sum_l A_s(x_{i,j,k,l})} \to \frac{A_s(x)}{R_d}, \quad A_s(x) = \begin{cases} e^{\mathcal{T}x-1}, & x < 0, \\ e^{\tanh(\mathcal{T}x)-1}, & x \geq 0. \end{cases},$$

where $\mathcal{T}$ acts as a temperature-like scaling factor that regulates the sharpness of the attention distribution. Moreover, we utilize batch-level statistics in training and reuse maximum values R_d as a constant during inference to avoid summation across vector in denominator. The mathematical proof demonstrating that XControl serves as an alternative to Softmax can be found in Appendix B.

Dynamic shift function. Current LN-based methods do not adapt to changing input distributions and lack learnable parameters for scaling and shifting, which limits their flexibility in adjusting normalized outputs. Furthermore, the computation of mean and variance can introduce overhead and may still lead to gradient flow issues in deep networks. Along this line, we devise a simple yet effective DSF to replace conventional LNs as: $\mathbf{DSF}(x) = \gamma(x - \mu) + \beta$. Instead of explicitly normalizing with activation variance, we regard DSF as a soft constraint that internalizes normalization behavior through teacher-guided distillation. This approach allows the model to learn normalization patterns from a more knowledgeable teacher, enabling it to adaptively adjust to varying input distributions.

Distillation of FHE-friendly model. While XControl enhances numerical stability through the use of bounded exponential activations, it fails to preserve the stringent statistical properties of Softmax without requiring explicit normalization. Consequently, aligning the scale and behavior of the teacher's attention layer may introduce optimization conflicts, ultimately undermining the

stability of the training process during distillation. To address these concerns, we have refined the existing one-step distillation approach by eliminating the loss associated with the attention matrix in each Transformer block.

Moreover, our observations indicate that for shallow semantic tasks, such as SST-2, the distillation signal derived from the teacher model's output is often weaker than the ground-truth labels, which can hinder convergence toward optimal checkpoints. To address this challenge, we introduce a task-specific hyperparameter θ that finely tunes the relative weighting between the hard-label supervision loss and the logits distillation loss, thereby enhancing overall training efficacy.

$$\mathcal{L} = \theta\mathcal{L}_{\text{task}} + (1-\theta)\mathcal{L}_{\text{logit}} + \mathcal{L}_{\text{embed}} + \mathcal{L}_{\text{norm}} \ ,$$

where $\mathcal{L}_{\text{task}}$ and $\mathcal{L}_{\text{logit}}$ represent the cross-entropy loss with ground-truth labels and the soft cross-entropy loss using the teacher's output logits, respectively; $\mathcal{L}_{\text{embed}}$ denotes the mean squared error (MSE) from the embedding layer; and $\mathcal{L}_{\text{norm}}$ refers to the MSE calculated after centered activations $(x-\mu)$ in the affine normalization module, computed prior to applying the learnable scaling and bias terms. This design enables XControl to offer a numerically stable formulation. We further use distillation supervision to align the output scale of a LayerNorm-style normalization module between the student and the teacher. Additionally, we implement DSF with affine normalization instead of LayerNorm. By removing input-dependent variance normalization, this approach reduces optimization instability caused by variance-driven rescaling that can disrupt teacher-provided signals.

4 FHE Evaluation Algorithms in CtrlFormer

4.1 Pipeline of CtrlFormer

CtrlFormer eliminates the need for separate activation evaluations that typically consume approximately 10 multiplicative levels. By reducing the overall multiplicative depth of the inference pipeline, this design substantially improves inference efficiency and lowers memory consumption. Figure 1 summarizes the multiplicative level consumption for each component of the transformer model.[3] We show that only 5 multiplicative levels are consumed before entering the functional bootstrapping process. Consuming fewer multiplicative levels enables the use of a smaller ciphertext modulus throughout the evaluation pipeline, yielding the following benefits:

(1) Homomorphic matrix multiplications are performed at lower levels, significantly improving computational efficiency. In practice, the runtime of homomorphic matrix multiplication grows approximately linearly in the ciphertext level because CKKS uses the Residue Number System (RNS), where a ciphertext at level L is represented across L RNS components. Homomorphic matrix multiplication consists of element-wise operations and is executed independently over all RNS components, causing the overall computational complexity to scale linearly with L. Therefore, reducing the ciphertext level directly decreases the computational cost of these operations.

(2) A smaller ciphertext modulus also reduces the size of ciphertexts and evaluation keys, leading to lower memory consumption. This improves the scalability of FHE-based inference and enables more efficient evaluation of large-scale transformer models.

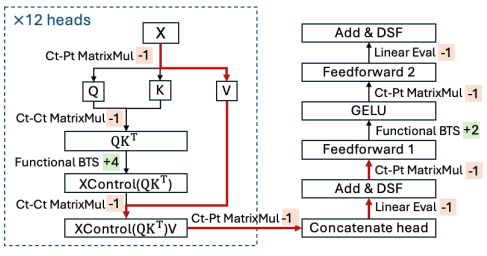

Figure 1: Level consumption of CtrlFormer.

Table 1: Flow of matrix multiplications.

Operation	Function	Input	Output
Attention layer	$\text{Enc}(X)W_*^{(h)}, * = Q, K, V$	Col-packing	Col-packing
	$\text{Enc}(Q^{(h)})\text{Enc}((K^{(h)})^\intercal)$	Col-packing	Diag-packing
	$\text{Enc}(\sigma(Q^{(h)}(K^{(h)})^\intercal)\text{Enc}(V^{(h)})$	Diag-packing +Col-packing	Col-packing
	$\text{Enc}((X^{(0)} \parallel \ldots \parallel X^{(H-1)}))W_O$	Col-packing	Col-packing
Feedforward layer	$\text{Enc}(X)W_{\text{fc1}}$	Col-packing	Col-packing
	$\text{Enc}(X)W_{\text{fc2}}$	Col-packing	Col-packing

[3] For simplicity, we omit the classification layer, which is applied only once after the 12 encoder layers. We apply bootstrapping at the end of each layer to restore multiplicative levels, ensuring consistent and correct inference throughout the entire model.

4.2 XControl and GELU Evaluation and Optimizations

This work optimizes the CKKS functional bootstrapping framework. More concretely, given a function $f(\cdot)$ (e.g., GELU), a polynomial $T_f(x)$ is constructed from interpolation and $f(\cdot)$ such that $T_f(x + q_0 I) \approx f(x)$, where $q_0 I$ is the redundant term added by ModRaise. [4]

SlotToCoeff and CoeffToSlot. We modify the functional bootstrapping flow so that it supports complex vector inputs and outputs. In addition, we accelerate them by methods in Han et al. (2019); Ju et al. (2024). Please refer to the details in Appendix D.1.

Functional evaluation. We optimize the CKKS evaluation of the polynomial interpolations of XControl and GELU using the Paterson–Stockmeyer algorithm (Paterson & Stockmeyer, 1973; Chen et al., 2019), reducing the multiplicative depth from $O(d)$ to $O(\sqrt{d})$ for degree-d polynomials (Appendix D.2).

4.3 Matrix Multiplications and Optimizations

CtrlFormer adopts a combination of column packing and diagonal packing strategies (Chen et al., 2024; Zhang et al., 2026; Yang et al., 2025) for matrix multiplications. Beyond prior work, CtrlFormer introduces several key optimizations that further improve efficiency, including (1) using the imaginary part of CKKS complex slots for computation and (2) applying lazy operations across multiple rotations or multiplications. While Powerformer (Park et al., 2025) also utilizes complex slots to speed up matrix multiplications, our method leverages complex-slot packing in a different way thereby achieving higher throughput.

Packing strategies and interleaved batching. Column packing stores each column of a matrix within the slots in a single ciphertext, while diagonal packing arranges elements along diagonals within ciphertext slots. Table 1 shows the evaluation flow of matrix multiplications we used in CtrlFormer, from prior work (Zhang et al., 2026; Yang et al., 2025). The evaluation flow ensures consistent packing formats, thereby avoiding costly format conversions and encrypted matrix transpositions. To fully utilize the large number of plaintext slots available in one ciphertext (typically 2^{15} slots), CtrlFormer targets high-throughput inference by batching a set of inputs into each inference round to reduce the amortized time cost. CtrlFormer offers flexibility by supporting a wide range of sequence lengths and batch sizes under various scenarios. CtrlFormer can support a sequence length of 512, which is the maximum token length of BERT. CtrlFormer adopts interleaved baching (Aharoni et al., 2023; Adir et al., 2024; Zhang et al., 2026; Yang et al., 2025), which interleaves inputs across slots by position, enabling efficient "internal rotations". [5] We elaborate two optimizations in matrix multiplications in Appendix E.

Imaginary part. ARION(Yang et al., 2025) only uses the real parts of plaintext slots. Recall that CKKS can encrypt a vector in $\mathbb{C}^{N/2}$, so leveraging all the imaginary parts can further improve the efficiency. Let $i = \sqrt{-1}$. First, ciphertext-plaintext matrix multiplications like $\text{Enc}(Q) = \text{Enc}(X)W$ are based on computing each column $\text{Enc}(\mathbf{q}_j^\intercal) = \sum_k w_{k,j}\text{Enc}(\mathbf{x}_k^\intercal)$. Let $W = (W_1||W_2)$ and define a complex matrix $\tilde{W} := W_1 + iW_2$, which has half the number of columns as W. So it reduces the number of multiplications by half. $\text{Enc}(Q)$ encrypts $Q_1 + iQ_2$ for $Q = (Q_1||Q_2)$. Next, in each head we can write $Q = (Q_1||Q_2), K = (K_1||K_2)$ and $QK^\intercal = Q_1K_1^\intercal + Q_2K_2^\intercal$. If we have encrypted $Q_1 + iQ_2, K_1 - iK_2$ (under col-packing), then $\text{Enc}(QK^\intercal)$ can be obtained by taking the real part of the encrypted data in $\text{Enc}((Q_1 + iQ_2)(K_1 + iK_2))$. In the multiplication with $\text{Enc}(V)$, which is encrypting $V_1 + iV_2$, it reduces half of the matrix-vector multiplications. Please refer to Table 8, 9, 10 in Appendix E for a detailed comparison with ARION.

Lazy operations. In ciphertext rotation or multiplication, an additional process is required to convert the ciphertext back to its "normal state" so that it is ready for the next operation. [6] The process is expensive. We speed up by reducing the number of such processes as in Halevi & Shoup (2018).

[4] In brief, the interpolation polynomial LUT can (1) remove $q_0 I$, which is essential in bootstrapping; and (2) evaluate the given function $f(x)$. This is why it is named *functional bootstrapping*.

[5] Elements from each vector are not supposed to be rotated to the slots of other vectors.

[6] More concretely, relinearization after each multiplication, or key-switching after each rotation.

5 EXPERIMENTS AND ANALYSES

Model and dataset. To ensure a fair comparison, we utilize the same BERT-base model with a sequence length of 128 as used in previous works (Zhang et al., 2025a; Moon et al., 2025; Zhang et al., 2026; Park et al., 2025; Yang et al., 2025). More details on the model selection are provided in Appendix A.1. Specifically, we conduct experiments on the RTE, MRPC, and SST-2 tasks from the GLUE benchmark.

Hyperparameters. We use the BERT-base pretrained model and set early stopping to 10 epochs for fine-tuning on each task. During the knowledge distillation training, we set a batch size of 64 and a learning rate of 5×10^{-5}. To ensure a fair comparison of model performance, we fix the early stopping criterion for distillation at 20 epochs, consistent with the approach used in Powerformer.

FHE parameters and implementations. We set the degree of the polynomial ring as $N = 2^{16}$, yielding 2^{15} plaintext slots in one ciphertext. The ciphertext modulus Q is chosen to ensure a 128-bit security level (Albrecht et al., 2022), and 5 multiplicative levels before entering functional bootstrapping. CtrlFormer is implemented using the PhantomFHE library[7]. All of our experiments are executed in a single NVIDIA A100 GPU. Our code is available. [8]

5.1 PLAINTEXT EVALUATION

In our experiments, the BERT-base model is fine-tuned individually for each task. These fine-tuned models serve as teachers for knowledge distillation, with their accuracies used as the baseline.

Parameter selection. We conduct a grid search to determine the optimal parameter t for XControl and the loss weighting coefficient θ. For the RTE and MRPC tasks, we search t ranging from 0.2 to 0.5 in increments of 0.1. For the SST-2 task, we search t ranging from 1.0 to 1.4 in the same increments. The loss weight θ is chosen from $\{0, 1\}$. The best parameter combination is selected based on the accuracy on the validation set. Empirically, we find that $t = 0.4, \theta = 0$ works best for RTE and MRPC, and $t = 1.3, \theta = 1$ performs best for SST-2. We attribute this variation to the role of t in controlling the sharpness of the XControl output distribution: larger t values yield sharper outputs with wider saturation regions and higher confidence, which benefits shallow semantic tasks such as SST-2, whereas smaller t values allow the function to remain sensitive over a wider range of input logits, making it more suitable for RTE and MRPC. Similarly, setting $\theta = 0$ yields a milder teacher signal that helps stabilize training for RTE and MRPC, while $\theta = 1$ provides stronger supervision from ground-truth labels which benefits the SST-2 task. The resulting models are distilled under these settings and then submitted to the official GLUE benchmark server to obtain the reported scores.

Main results. We compare CtrlFormer against BERT and Powerformer. In Table 2, [9] it clearly indicates that our method consistently outperforms both approaches across all three tasks: RTE, MRPC, and SST-2. Overall, we achieve an average accuracy of 80.77%, higher than BERT (79.26%) and Powerformer (80.23%). These improvements can be attributed to more effective feature learning and task-specific distillation strategies. In Appendix G, we evaluate our method on more tasks and model. In Appendix H, we present the calibration performance comparison with Softmax.

Ablation study of CtrlFormer. Table 3 presents ablation results for the XControl and DSF. We set the following ablation groups: (1) w/o XControl (i.e., Softmax + DSF), (2) w/o DSF (i.e., XControl + Layernorm), and (3) CtrlFormer (i.e., XControl + DSF). XControl improves performance across all three tasks, while DSF improves MRPC and SST-2; their combination achieves the highest average accuracy. When both components are applied, the model achieves the highest overall average accuracy of 80.77%. These confirm that each proposed component is effective individually, and that their combination provides complementary benefits, leading to the best overall performance in our framework. More ablation experiments are in Appendix F.

[7] https://github.com/encryptorion-lab/phantom-fhe

[8] https://anonymous.4open.science/r/CtrlFormer-2C33

[9] * BERT is the pretrained model "BERT-base" fine-tuned on each task under same fine-tuning setting of Powerformer's baseline. ** The scores are obtained by reimplementing Powerformer based on their official open-source code and submitting to official test server of GLUE benchmark.

Table 2: Main results on GLUE test set (%).

Model	RTE	MRPC	SST-2	Average
BERT*	64.9	80.5	92.4	79.3
Powerformer **	67.2	80.7	92.8	80.2
Ours	**67.5**↑	**81.8**↑	**93.0**↑	**80.8**↑

Table 3: Ablation study of CtrlFormer (%).

Model	RTE	MRPC	SST-2	Average
w/o XControl	66.6	80.7	92.8	80.0
w/o DSF	**67.6**	81.7	92.1	80.5
CtrlFormer	67.5	**81.8**↑	**93.0**↑	**80.8**↑

Table 4: Functional Bootstrapping.

Steps	Time cost (seconds)			
	without (1)(2)	with (1)	with (2)	with (1)(2)
SlotToCoeff	0.1	**0.04**	0.1	0.04
ModRaise	0.001	0.001	0.001	0.001
CoeffToSlot	0.56	**0.21**	0.56	0.21
EvalLUT	1.18	1.18	**0.54**	0.54
Total	1.84	1.43	1.2	**0.79**

Table 5: Matrix multiplications in one layer.

MatrixMul	Time cost (milliseconds)			
	without (I)(II)	with (I)	with (II)	with (I)(II)
$\mathrm{Enc}(X)W_*, * = Q, K, V$	17	8.8	5.7	**2.96**
$\mathrm{Enc}(Q)\mathrm{Enc}(K^\intercal)$	32.9	16.9	22.6	**10.96**
$\mathrm{Enc}(X)\mathrm{Enc}(V)$	22.1	12.6	7.6	**4.1**
$\mathrm{Enc}(X)W_{\mathrm{O}}$	142	73.5	47.6	**24.1**
$\mathrm{Enc}(X)W_{\mathrm{fc1}}$	282	145	96.5	**47.8**
$\mathrm{Enc}(X)W_{\mathrm{fc2}}$	274	138.8	92	**46.8**

5.2 CIPHERTEXT EVALUATION

Microbenchmarks: Functional bootstrapping. We modify the CKKS functional bootstrapping procedure so that it supports complex-valued vector input, and further improve its efficiency by applying (1) faster SlotToCoeff and CoeffToSlot techniques and (2) parallelized Paterson-Stockmeyer evaluation. The optimized functional bootstrapping achieves a $2.3\times$ speedup (Table 4).

Microbenchmarks: Matrix multiplication. CtrlFormer leverages (I) the imaginary parts of slots and (II) lazy-style operations to speed up matrix multiplications. We evaluate the two optimizations within one transformer layer and report results in Table 5. Beyond optimizations in matrix multiplication algorithm, CtrlFormer also reduces the number of multiplicative levels required throughout the inference pipeline. This enables matrix multiplications to be evaluated at lower levels with smaller ciphertext modulus. Figure 2,3 show the time cost on different levels.[10] Across both types of matrix multiplications and varying ciphertext levels, CtrlFormer consistently outperforms ARION, MOAI, THOR, and Powerformer. Moreover, by performing matrix multiplications at lower levels (Section 4.1), CtrlFormer further reduces runtime, achieving a $2.5\times$ to $24.6\times$ speedup in total matrix multiplication cost for BERT-base inference.

End-to-end evaluation. CtrlFormer is an end-to-end FHE-based transformer architecture that integrates all proposed techniques. Table 6 reports GLUE scores under unencrypted / encrypted inputs. Figure 4 show error propagation during encrypted evaluation, which demonstrates that the error grows in the first few layers then stabilizes and remains bounded. Overall, accuracy under encrypted inputs matches that of the unencrypted inputs. The results demonstrate that the computational noise from HE operations has negligible effect.

We compare CtrlFormer with THOR, Powerformer and MOAI under the same environment [11] and report in Table 7. CtrlFormer achieves an amortized inference latency of 1.8 minutes per input. Compared to state-of-the-art FHE-friendly transformer architectures (Powerformer), CtrlFormer **mitigates feature explosion**, achieves $3.17\times$ and $1.4\times$ speedups over Powerformer and MOAI-on-Powerformer's-architecture, respectively.

In Table 7 we also show a feasible ablation of CtrlFormer. In each setting, we remove one component and then retune the ciphertext levels and bootstrapping positions to obtain a valid executable pipeline. For w/o XC and w/o DSF, we follow MOAI's method (Zhang et al., 2026) to evaluate Softmax and LayerNorm, respectively. For w/o FB, we follow MOAI's method (Zhang et al., 2026) to evaluate GELU and use the commonly adopted degree-128 polynomial approximation for the exponential in XControl, as in MOAI (Zhang et al., 2026) and THOR (Moon et al., 2025).

[10] All results are obtained using a single thread. We benchmark ciphertext-plaintext matrix multiplication in the self-output layer, with dimensions $(128 \times 768) \times (768 \times 768)$, in Figure 2, and ciphertext-ciphertext matrix multiplication in the attention heads, with dimensions $(128 \times 128) \times (128 \times 64) \times 12$, in Figure 3.

[11] MOAI reports inference time for the original BERT-base model on both H200 and A100 GPUs; however, the time cost for MOAI deployed on Powerformer's architecture is reported only on the A100 GPU.

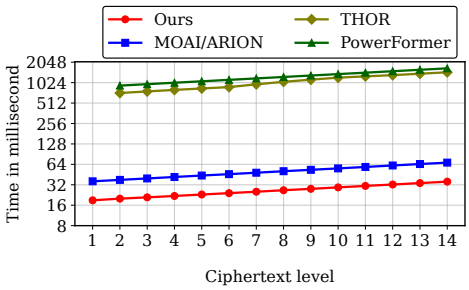

Figure 2: Time of ciphertext–plaintext matrix multiplications (per input).

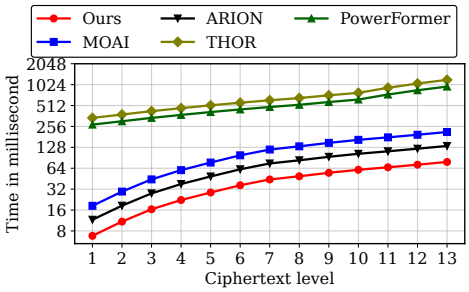

Figure 3: Time of ciphertext–ciphertext matrix multiplications (per input).

Table 6: Inference accuracy (GLUE scores).

Input	RTE	MRPC	SST-2	Average
Unencrypted input	67.5	81.8	93.0	80.8
Encrypted input	67.5	81.8	93.0	80.8

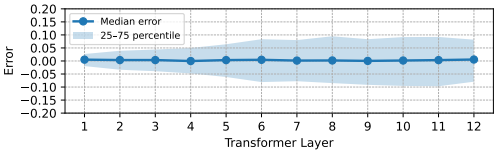

Figure 4: Error of 12 layers.

The ablations confirm that all three components contribute to the final speedup. Removing XControl, DSF, or functional bootstrapping increases latency from 108.6 s to 175.5 s (1.62×), 231.0 s (2.13×), and 195.4 s (1.80×), respectively, primarily due to additional bootstrapping and the higher ciphertext levels required to evaluate Softmax, LayerNorm, and nonlinear functions separately.

Table 7: End-to-end encrypted BERT-base inference time (second) on a single NVIDIA A100. Power. denotes Powerformer; MOAI-P denotes MOAI evaluated with Powerformer's architecture; XC denotes XControl; and FB denotes functional bootstrapping.

	Prior systems				CtrlFormer ablations			Ours
Operation	THOR	Power.	MOAI	MOAI-P	w/o XC	w/o DSF	w/o FB	Full
Q, K, V projections	49.77	57.65	16.54	14.29	8.70	3.20	7.90	**3.20**
$QK^\top$	16.25	28.76	14.07	6.61	3.80	1.35	3.30	**1.35**
Softmax/replacement	15.53	0.75	2.19	0.06	2.10	0	1.80	**0**
Attention $\times V$	13.08	18.95	0.29	0.39	2.70	0.39	0.81	**0.39**
Output projection	27.43	22.54	0.48	4.05	0.23	0.21	0.65	**0.21**
First LN/replacement	7.13	0.37	0.15	0.004	0.04	0.15	0.04	**0.04**
FFN projection 1	49.80	59.21	16.18	4.79	3.65	3.63	1.12	**3.64**
GELU	29.42	8.31	3.30	5.20	0	0	3.30	**0**
FFN projection 2	49.19	43.77	2.88	13.79	0.62	0.59	0.59	**0.59**
Second LN/replacement	4.10	0.30	0.15	0.004	0.03	0.15	0.03	**0.03**
(F)Bootstrapping	337.86	103.72	227.84	102.50	153.60	221.35	175.90	**99.17**
Total	599.56	344.32	284.07	151.70	175.50	231.00	195.40	**108.62**
Slowdown vs. full	5.52×	3.17×	2.61×	1.40×	1.62×	2.13×	1.80×	1.00×

6 Conclusion

We propose CtrlFormer, a novel FHE-based transformer inference framework that addresses feature explosion while improving efficiency. CtrlFormer introduces XControl, a bounded FHE-friendly alternative to Softmax, and reduces multiplicative depth by eliminating separate activation evaluations, enabling faster homomorphic matrix multiplications with smaller ciphertexts. We further present optimizations for matrix multiplication and functional bootstrapping. Together, CtrlFormer achieves improved GLUE performance with significantly reduced inference time.

AI USE STATEMENT

In this work, we only used generative AI tools for sentence-level language polishing and grammar checking. We have not used generative AI tools for core algorithm design, experiments or implementations. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Our code is available via https://anonymous.4open.science/r/CtrlFormer-2C33. It contains two folders of plaintext training experiments and ciphertext inference experiments, respectively. We have include README markdown files under each of the two folders, elaborating steps how to set up requirements and reproduce experimental results of this paper.

REFERENCES

Allon Adir, Ehud Aharoni, Nir Drucker, Ronen Levy, Hayim Shaul, and Omri Soceanu. *Homomorphic Encryption for Data Science (HE4DS)*. Springer, 2024.

Ehud Aharoni, Allon Adir, Moran Baruch, Nir Drucker, Gilad Ezov, Ariel Farkash, Lev Greenberg, Ramy Masalha, Guy Moshkowich, Dov Murik, et al. Helayers: A tile tensors framework for large neural networks on encrypted data. *Proceedings on Privacy Enhancing Technologies*, 2023.

Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption standard. In *Protecting privacy through homomorphic encryption*, pp. 31–62. Springer, 2022.

Andreea Alexandru, Andrey Kim, and Yuriy Polyakov. General functional bootstrapping using ckks. In *Annual International Cryptology Conference*, pp. 304–337. Springer, 2025.

Matan Avitan, Moran Baruch, Nir Drucker, Itamar Zimerman, and Yoav Goldberg. Efficient decoding methods for language models on encrypted data. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 1777–1794, 2025.

Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 34–54. Springer, 2019.

Jingwei Chen, Linhan Yang, Chen Yang, Shuai Wang, Rui Li, Weijie Miao, Wenyuan Wu, Li Yang, Kang Wu, and Lizhong Dai. Secure transformer-based neural network inference for protein sequence classification. Cryptology ePrint Archive, Paper 2024/1851, 2024. URL `https://eprint.iacr.org/2024/1851`.

Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. The-x: Privacy-preserving transformer inference with homomorphic encryption. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 3510–3520, 2022.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 409–437. Springer, 2017.

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

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2924–2936, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, Wen-Guang Chen, et al. Puma: Secure inference of llama-7b in five minutes. *Security and Safety*, 4:2025014, 2025.

Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 617–640. Springer, 2015.

Jules Dumezy, Andreea Alexandru, Yuriy Polyakov, Pierre-Emmanuel Clet, Olive Chakraborty, and Aymen Boudguiga. Evaluating larger lookup tables using ckks. *Cryptology ePrint Archive*, 2025.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 169–178, 2009.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in helib. In *Annual International Cryptology Conference*, pp. 93–120. Springer, 2018.

Kyoohyung Han, Minki Hhan, and Jung Hee Cheon. Improved homomorphic discrete fourier transforms and fhe bootstrapping. *IEEE Access*, 7:57361–57370, 2019.

Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009. ISBN 9780387848570. doi: 10.1007/978-0-387-84858-7. URL https://doi.org/10.1007/978-0-387-84858-7.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Nandan Kumar Jha and Brandon Reagen. Entropy-guided attention for private llms. *arXiv preprint arXiv:2501.03489*, 2025.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

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

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 252–262, 2018.

Dacheng Li, Hongyi Wang, Rulin Shao, Han Guo, Eric Xing, and Hao Zhang. Mpcformer: Fast, performant and private transformer inference with mpc. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Lawrence Lim, Jiaming Liu, Vikas Kalagi, Amr El Abbadi, and Divyakant Agrawal. Hyperion: Private token sampling with homomorphic encryption. Cryptology ePrint Archive, Paper 2025/2318, 2025. URL https://eprint.iacr.org/2025/2318.

Zeyu Liu and Yunhao Wang. Amortized functional bootstrapping in less than 7 ms, with o˜(1) polynomial multiplications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 101–132. Springer, 2023.

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

Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. Bumblebee: Secure two-party inference framework for large transformers. *Cryptology ePrint Archive*, 2023.

Jinglong Luo, Yehong Zhang, Zhuo Zhang, Jiaqi Zhang, Xin Mu, Hui Wang, Yue Yu, and Zenglin Xu. Secformer: fast and accurate privacy-preserving inference for transformer models via smpc. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 13333–13348, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

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

Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. Bolt: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 4753–4771. IEEE, 2024.

Dongjin Park, Eunsang Lee, and Joon-Woo Lee. Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11090–11111, 2025.

Hanjun Park, Byeongseo Min, Jiheon Woo, Min-Wook Jeong, Jongho Shin, Yongwoo Lee, Young-Sik Kim, and Yongjune Kim. Cgf-softmax: A cumulant-based softmax reformulation for efficient inference under homomorphic encryption, 2026. URL https://arxiv.org/abs/2602.01621.

Michael S Paterson and Larry J Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing*, 2(1):60–66, 1973.

Donghwan Rho, Taeseong Kim, Minje Park, Jung Woo Kim, Hyunsik Chae, Ernest K Ryu, and Jung Hee Cheon. Encryption-friendly llm architecture. *arXiv preprint arXiv:2410.02486*, 2024.

Donghwan Rho, Sieun Seo, Hyewon Sung, Chohong Min, and Ernest K Ryu. Traveling salesman-based token ordering improves stability in homomorphically encrypted language models. *arXiv preprint arXiv:2510.12343*, 2025.

Praneet Suresh, Jack Stanley, Sonia Joseph, Luca Scimeca, and Danilo Bzdok. From noise to narrative: Tracing the origins of hallucinations in transformers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, 2018.

Tianshi Xu, Wen-jie Lu, Jiangrui Yu, Yi Chen, Chenqi Lin, Runsheng Wang, and Meng Li. Breaking the layer barrier: Remodeling private transformer inference with hybrid {CKKS} and {MPC}. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2653–2672, 2025.

Linhan Yang, Jingwei Chen, Wangchen Dai, Shuai Wang, Wenyuan Wu, and Yong Feng. ARION: Attention-optimized transformer inference on encrypted data. Cryptology ePrint Archive, Paper 2025/2271, 2025. URL https://eprint.iacr.org/2025/2271.

Zhaomin Yang, Chao Niu, Benqiang Wei, Zhicong Huang, Cheng Hong, and Tao Wei. {RBOOT}: Accelerating homomorphic neural network inference by fusing {ReLU} within bootstrapping. In *35th USENIX Security Symposium (USENIX Security 26)*, pp. 121–140, 2026.

Wenxuan Zeng, Ye Dong, Jinjin Zhou, Junming Ma, Jin Tan, Runsheng Wang, and Meng Li. Mpcache: Mpc-friendly kv cache eviction for efficient private large language model inference. *arXiv preprint arXiv:2501.06807*, 2025.

Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen-jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. Secure transformer inference made non-interactive. In *NDSS*, 2025a.

Linru Zhang, Xiangning Wang, Xianhui Lu, Huaxiong Wang, and Kwok-Yan Lam. LEAF: A low-latency evaluation architecture for feedforward block in privacy-preserving transformer inference. Cryptology ePrint Archive, Paper 2025/956, 2025b. URL https://eprint.iacr.org/2025/956.

Linru Zhang, Xiangning Wang, Jun Jie Sim, Zhicong Huang, Jiahao Zhong, Huaxiong Wang, Pu Duan, and Kwok-Yan Lam. MOAI: Module-optimizing architecture for non-interactive secure transformer inference. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yancheng Zhang, Jiaqi Xue, Mengxin Zheng, Mimi Xie, Mingzhe Zhang, Lei Jiang, and Qian Lou. Cipherprune: Efficient and scalable private transformer inference. *arXiv preprint arXiv:2502.16782*, 2025c.

Itamar Zimerman, Allon Adir, Ehud Aharoni, Matan Avitan, Moran Baruch, Nir Drucker, Jenny Lerner, Ramy Masalha, Reut Meiri, and Omri Soceanu. Power-softmax: Towards secure llm inference over encrypted data. *arXiv preprint arXiv:2410.09457*, 2024a.

Itamar Zimerman, Moran Baruch, Nir Drucker, Gilad Ezov, Omri Soceanu, and Lior Wolf. Converting transformers to polynomial form for secure inference over homomorphic encryption. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=9HPoJ6ulgV.

A PRELIMINARY: THE BERT MODEL

The BERT model was proposed by Google Devlin et al. (2019) using the encoder-only transformer architecture. The BERT model is composed of a stack of $L = 12$ layers, where each layer consists of multi-head attention (12-head) followed by two normalizations and a feed-forward block in between.

Multi-Head Attention. Let $X \in \mathbb{R}^{m\times d}$ be the input, $W_Q, W_K, W_V \in \mathbb{R}^{d\times d'}$ be weight matrices, and $\mathbf{b}_Q, \mathbf{b}_K, \mathbf{b}_V$ be bias vectors. An attention block in head $h \in [H]$ is defined as:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\intercal}{\sqrt{d'}}\right)V \in \mathbb{R}^{m\times d'}$$

where $* := XW_* + \mathbf{1}^\intercal\mathbf{b}_*, * = Q, K, V$, and

$$\text{Softmax}(\mathbf{x}) = \left[\frac{\exp(x_i)}{\sum_{r\in[n]}\exp(x_r)}\right]_{i\in[n]}$$

The multi-head attention is the execution of H parallel attention blocks and we set the h-th result to be $X^{(h)}$ where $d' := d/H$. Then the H matrices are concatenated and the weight matrix W_O is multiplied and the bias $\mathbf{b}_\mathsf{O}$ is added:

$$X^{\text{Concat}} := (X^{(0)} \parallel X^{(1)} \parallel ... \parallel X^{(H-1)})W_\mathsf{O} + \mathbf{1}^\intercal\mathbf{b}_\mathsf{O}$$

The attention block ends with a layer normalization procedure with a residual connection:

$$X := \text{Layernorm}(X + X^{\text{Concat}})$$

where

$$\text{Layernorm}(\mathbf{x}) := \left[\gamma \cdot \frac{x_i - \text{mean}(\mathbf{x})}{\sqrt{\text{var}(\mathbf{x})}} + \beta\right]$$

γ, β are model parameters.

Feed Forward Block. The feed forward network contains two fully connected layers and one Gaussian Error Linear Unit (GELU) Hendrycks & Gimpel (2016) activation function in between. The feed forward block also ends with a layer normalization as defined above.

The GELU function is defined as $\text{GELU}(x) := \frac{x}{2}\left(1 + \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt\right)$. Since there is no closed-form expression for GELU, Hendrycks & Gimpel (2016) suggested an approximation by tanh:

$$\text{GELU}(x) \approx 0.5x\left(1 + \tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right)$$

Let $W_{\mathsf{fc1}}, W_{\mathsf{fc2}}$ be weight matrices, and $\mathbf{b}_{\mathsf{fc1}}, \mathbf{b}_{\mathsf{fc2}}$ be bias vectors. The feed forward network can be described as follows:

$$\text{Layernorm}(X + \text{GELU}(XW_{\mathsf{fc1}} + \mathbf{1}^\intercal\mathbf{b}_{\mathsf{fc1}})W_{\mathsf{fc2}} + \mathbf{1}^\intercal\mathbf{b}_{\mathsf{fc2}})$$

The output of feed forward block is set to be the input of the attention block in the next transformer layer.

After 12 transformer layers, the first token is fed into a pooler, which is a dense layer followed by a $\tanh(\cdot)$. Let this vector be $\mathbf{x}_{[\text{CLS}]} \in \mathbb{R}^d$. The classification layer first computes $\tanh(\mathbf{x}_{[\text{CLS}]}W + \mathbf{1}^\intercal\mathbf{b}) \in \mathbb{R}^d$, denoted as $\mathbf{x}$. Finally let c be the number of classes, the output is $\mathbf{x}W_{\mathsf{cls}} + \mathbf{1}^\intercal\mathbf{b}_{\mathsf{cls}} \in \mathbb{R}^c$.

A.1 CHOICE OF MODEL

We provide a detailed justification for the model choice in our work. BERT-base Devlin et al. (2019), an encoder-only Transformer model commonly used in batch-processing tasks such as document classification, widely used as a benchmark in prior FHE-based transformer inference studies Zhang et al. (2025a); Moon et al. (2025); Park et al. (2025); Zhang et al. (2026); Yang et al. (2025) due to its balanced trade-off between model expressiveness and computational feasibility.

Limitation. FHE-based inference for large decoder-only models such as LLaMA-3-8B is currently impractical. Compared to encoder models, decoder inference requires autoregressive generation,

where each token depends on all previously generated tokens. Autoregressive generation enforces strictly sequential computation, preventing batching across tokens and requiring repeated FHE evaluations. Although NEXUS Zhang et al. (2025a) and MOAI Zhang et al. (2026) also considered inference of Llama-3-8B model, they only list the amortized cost for each component of the model. They do not propose an efficient solution for FHE-based autoregressive generation.

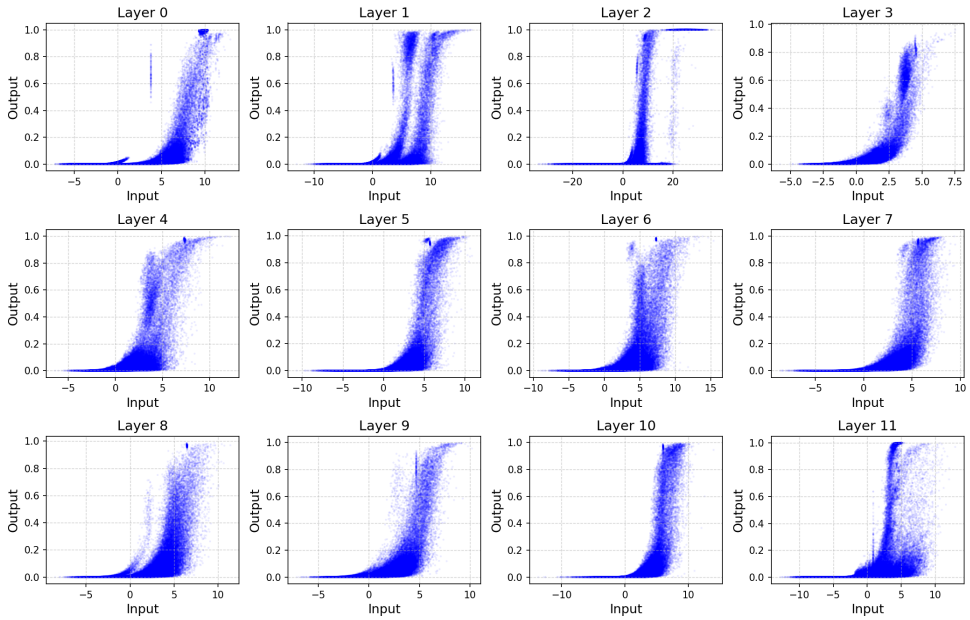

Figure 5: **Input-output distribution of the softmax function with max shifting operation across layers of BERT.** We use a single example from SST-2 task development set and plot the input / output values of softmax function across the 12 layers of a BERT model finetuned on SST-2. The S-shaped curves closely resemble those of our XControl function (see Figure 6). With appropriate input shifting and rescaling guided by distillation knowledge, the token-wise exponential-like relation is preserved while constraining the output values within a bounded range, similar to the effect of max-shifting operation. Furthermore, we introduce $\mathcal{T}$ to control the sharpness of XControl, facilitating more effective distillation under task discrepancies.

B Mathematical Proof: XControl Alternative to Softmax

B.1 Definitions

1. **Softmax Function**: The Softmax function is defined as:

$$P(y_i|z) = \frac{e^{z_i}}{\sum_j e^{z_j}}. \tag{1}$$

2. **XControl Function**: The XControl is defined as:

$$A_s(x) = \begin{cases} e^{\mathcal{T}\mathbf{x}-1}, & \mathbf{x} < 0, \\ e^{\tanh(\mathcal{T}\mathbf{x})-1}, & \mathbf{x} \geq 0. \end{cases} \tag{2}$$

The overall XControl function is given by:

$$\text{XControl}(x) = \frac{A_s(x)}{\max_i \sum_l A_s(x_{i,j,k,l})} \to \frac{A_s(x)}{R_d}, \tag{3}$$

where R_d is the maximum value used during inference.

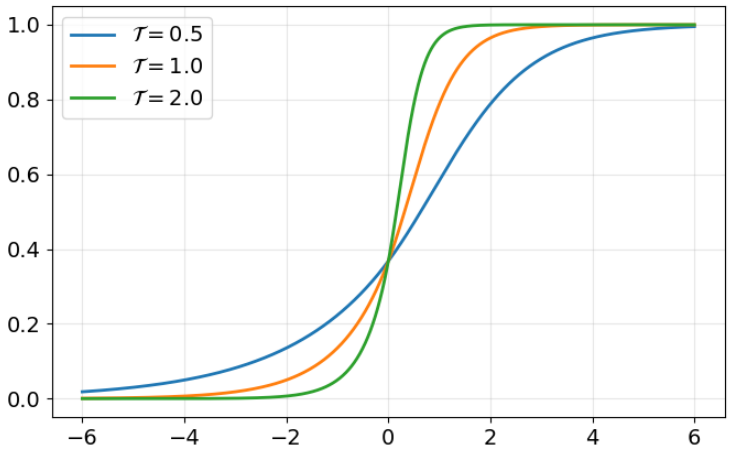

Figure 6: XControl function with three different $\mathcal{T}$

B.2 Step 1: Limit Behavior Analysis

To show that XControl can replace Softmax, we will analyze the behavior of both functions as x approaches extreme values.

B.2.1 Softmax Limit

As $z_i \to \infty$ (assuming z_j for $j \neq i$ remains bounded):

$$\lim_{z_i \to \infty} P(y_i|z) = \lim_{z_i \to \infty} \frac{e^{z_i}}{e^{z_i} + \sum_{j \neq i} e^{z_j}} = 1. \tag{4}$$

B.2.2 XControl Limit

For $x \to \infty$: Then $\mathbf{x} \geq 0$:

$$A_s(x) = e^{\tanh(\mathcal{T}x)-1} \to e^{1-1} = 1. \tag{5}$$

For $x \to -\infty$: Then $\mathbf{x} < 0$:

$$A_s(x) = e^{\mathcal{T}x-1} \to 0 \text{ as } x \to -\infty. \tag{6}$$

Thus, for large positive values of x:

$$\text{XControl}(x) \to \frac{1}{R_d} \text{ (where } R_d \text{ is the maximum value).} \tag{7}$$

B.3 Step 2: Continuity and Differentiability

Both functions are continuous and differentiable:

The Softmax function is continuously differentiable across its domain.

The XControl function is also continuous and differentiable due to the smooth nature of the exponential and hyperbolic tangent functions.

B.4 Step 3: Temperature Scaling

The temperature parameter $\mathcal{T}$ in XControl allows for control over the sharpness of the distribution:

As $\mathcal{T} \to +\infty$: The XControl function approaches a hard maximum, similar to the behavior of the Softmax function:

$$\text{XControl}(x) \approx \frac{e^{\mathcal{T}x}}{\sum_j e^{\mathcal{T}x_j}} \text{ (similar to Softmax).} \tag{8}$$

As $\mathcal{T}$ decreases, the function becomes smoother, allowing for a more uniform distribution while still preserving the overall structure. In conclusion, we have shown that:

- As x approaches infinity, both the XControl and Softmax functions converge to a similar limit.
- Both functions are continuous and differentiable, allowing for effective optimization.
- The temperature parameter $\mathcal{T}$ in XControl provides a mechanism to adjust the sharpness of the distribution, akin to the behavior of Softmax.

Thus, we establish that the XControl function can effectively serve as an alternative to the Softmax function (shown in Fig. 5), providing similar probabilistic interpretations with enhanced numerical stability and gradient properties.

C Details in functional bootstrapping

In this section, we include an extended introduction of the CKKS scheme and the functional bootstrapping under the CKKS scheme from the first work of this research line (Alexandru et al., 2025). In below, we first introduce the flow of normal CKKS bootstrapping Cheon et al. (2018); then we introduce the flow of functional bootstrapping Alexandru et al. (2025). Alexandru et al. (2025) proposed a novel method to evaluate an arbitrary function $f(\cdot) : \mathbb{Z}_p \to \mathbb{Z}_p$ with any integer $p \geq 2$, meanwhile the bootstrapping can be accomplished.

In Section D.1 we introduce our optimizations of using CKKS functional bootstrapping in transformer inference.

Notations. In this section, we adopt notation that differs slightly from that used in the main text, because more details of the CKKS scheme are elaborated. The CKKS scheme uses the *canonical embedding* from polynomials in $\mathbb{R}[X]/(X^N+1)$ where N is power-of-2, to $\mathbb{C}^{N/2}$. Let ζ be the fixed primitive complex $2N$-th root of unity: $\zeta^{2N} = 1$ but $\zeta^j \neq 1, \forall 0 < j < 2N$.

We can define the discrete Fourier transform $\mathbb{R}[X]/(X^N+1) \to \mathbb{C}^{N/2}$:

$$\mathsf{DFT}(a(X)) := [a(\zeta^{5^i})]_{i\in[N/2]} \in \mathbb{C}^{N/2}\,.$$

If we write $a(X)$ in a vector form by denoting $\boldsymbol{a}_{coeff} \in \mathbb{R}^N$ as all its coefficients in a *row* vector, and define matrix $V \in \mathbb{C}^{\frac{N}{2}\times\frac{N}{2}}$:

$$V := \begin{pmatrix} 1 & \zeta^{5^0} & (\zeta^{5^0})^2 & \cdots (\zeta^{5^0})^{N/2-1} \\ 1 & \zeta^{5^1} & (\zeta^{5^1})^2 & \cdots (\zeta^{5^1})^{N/2-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \zeta^{5^{N/2-1}} & (\zeta^{5^{N/2}-1})^2 & \cdots (\zeta^{5^{N/2}-1})^{N/2-1} \end{pmatrix}, \tag{9}$$

then $\mathsf{DFT}(a(X)) = (V||iV)\boldsymbol{a}^{\intercal}_{coeff} \in \mathbb{C}^{N/2}$. Let iDFT be its inverse process: $\mathbb{C}^{N/2} \to \mathbb{R}[X]/(X^N+1)$.

We apply the following notations in this section:
Message. It is a complex vector $\boldsymbol{z} \in \mathbb{C}^{N/2}$. Two kinds of methods to encode the message $\boldsymbol{z}$:

1. *Polynomial encoding* of $\boldsymbol{z}$: denoted as $z(X) \in \mathbb{R}[X]/(X^N+1)$, whose coefficient vector (N entries) is the concatenation $(\mathrm{Re}(\boldsymbol{z})||\mathrm{Im}(\boldsymbol{z}))$, i.e.,
$$\boldsymbol{z}_{coeff} \in \mathbb{R}^N, \boldsymbol{z}_{coeff} := (\mathrm{Re}(\boldsymbol{z})||\mathrm{Im}(\boldsymbol{z})).$$
2. *Integer polynomial encoding of* $\boldsymbol{z}$: In the CKKS scheme we may use a rounded integer polynomial $z_{int}(X) := \lceil \Delta \cdot z(X) \rfloor \in \mathbb{Z}[X]/(X^N+1)$.
3. *Canonical encoding* of $\boldsymbol{z}$: denoted as $\mathsf{Ecd}(\boldsymbol{z}) := \lceil \Delta \cdot \mathsf{iDFT}(\boldsymbol{z}) \rfloor \in \mathbb{Z}[X]/(X^N+1)$.

When it is clear from the context, the factor Δ will be omitted.

CKKS encryption. Given a plaintext polynomial $p(X) = \mathsf{Ecd}(\boldsymbol{z}) \in \mathbb{Z}[X]/(X^N+1)$, the encryption algorithm generates $(b(X), a(X)) \in (\mathbb{Z}_Q[X]/(X^N+1))^2$ where Q is the *ciphertext modulus*. Then we can say a CKKS ciphertext $(b(X), a(X))$ is encrypting *polynomial* $p(X)$, or is encrypting message *vector* $\boldsymbol{z}$.

ModRaise. It turns a CKKS ciphertext encrypting $p(X)$ with modulus Q_0 into another ciphertext encrypting $p(X) + k(X)Q_0$ with modulus $Q \gg Q_0$ and $k(X)$ is an integer polynomial.

SlotToCoeff. Denoted as StC. It turns a CKKS ciphertext encrypting message vector $\boldsymbol{z} \in \mathbb{C}^{N/2}$ into another CKKS ciphertext encrypting the *integer polynomial encoding* of $\boldsymbol{z}$: $z_{int}(X) = \lceil \Delta z(X) \rfloor$, $z(X)$'s coefficients are in vector $(\mathrm{Re}(\boldsymbol{z})||\mathrm{Im}(\boldsymbol{z}))$.

CoeffToSlot. Denoted as CtS. It turns a CKKS ciphertext encrypting polynomial $z_{int}(X)$ into another CKKS ciphertext encrypting vector $\boldsymbol{z}$ whose *integer polynomial encoding* is $z_{int}(X)$.

We use the following flow to introduce the process of normal CKKS bootstrapping, omitting Δ. It shows the change of encrypting data.

$$\begin{aligned} \boldsymbol{z} &\xrightarrow{\mathsf{StC}} z_{int}(X) \xrightarrow{\mathsf{ModRaise}} z_{int}(X) + k(X) \cdot Q_0 \\ &\xrightarrow{\mathsf{CtS}} \boldsymbol{z} + Q_0 \boldsymbol{k} \xrightarrow{\mathsf{EvalMod}} \boldsymbol{z}. \end{aligned}$$

Here $\mathsf{EvalMod}$ is a polynomial evaluation that can approximate the function of $\mod Q_0$, therefore the term $Q_0\boldsymbol{k}$ can be removed.

Alexandru et al. (2025) observed that while evaluating $\mathsf{EvalMod}$, an arbitrary function $f(\cdot) : \mathbb{Z}_p \to \mathbb{Z}_p$ can be evaluated at the same time. More concretely, we start from the CKKS ciphertext encrypting bounded vector $\boldsymbol{m} \in \mathbb{Z}_p^{N/2}$. The functional bootstrapping flow is:

$$\begin{aligned} \frac{\boldsymbol{m}}{p} &\xrightarrow{\mathsf{StC}} \frac{m_{int}(X)}{p} \xrightarrow{\mathsf{ModRaise}} \frac{\Delta m(X)}{p} + \frac{k(X)}{p} \cdot Q_0 \\ &\xrightarrow{\mathsf{CtS}} \frac{\boldsymbol{m}}{p} + Q_0 \frac{\boldsymbol{k}}{p} \xrightarrow{\mathsf{EvalLUT}} [f(m_0), f(m_1), ..., f(m_{N/2-1})] \,. \end{aligned} \tag{10}$$

Here $\mathsf{EvalLUT}$ is a *first-order trigonometric Hermite interpolation* that satisfies $\forall m \in \mathbb{Z}_p$:

$$\mathsf{EvalLUT}(\frac{m}{p} + I) = f(m), \mathsf{EvalLUT}'(\frac{m}{p} + I) = 0 \,,$$

where I is a bounded integer. The functional bootstrapping (Eq. (10)) is correct as long as $||\frac{Q_0}{p}\boldsymbol{k}||_\infty$ is bounded by the bound of I.

Parameter setting. In our scheme, we follow the parameter settings in Alexandru et al. (2025). We set $N = 2^{16}$ so there are 2^{15} slots. We set $p = 2^9$ to ensure a 9-bit look-up-table in evaluating the arbitrary function $f(\cdot) : \mathbb{Z}_p \to \mathbb{Z}_p$. In $\mathsf{EvalLUT}$, we first approximate $e^{\frac{2\pi i x}{4}}$ for $x \in [-25, 25]$. Then using two double-angle iterations we obtain $e^{2\pi i x}$. The remaining step is to evaluate a polynomial of $e^{2\pi i x}$ with degree p. Its coefficients are determined by the *first-order trigonometric Hermite interpolation*. Please refer to Corollary 1 in Alexandru et al. (2025) for the details.

D Our optimizations for functional bootstrapping

D.1 Optimized StC and CtS

It turns out that treating the CKKS functional bootstrapping as a black box is suboptimal for transformer inference, particularly when aiming to fully use the message slots in complex-number. In this section we will show that FHE operations StC and CtS are effectively matrix-vector multiplications given an encrypted vector and a plaintext matrix. In addition, we will show how we modify the operations StC and CtS to fit our flow with encrypted complex vectors. By representing polynomial using the coefficient vector, we first show that the discrete Fourier transform $\mathsf{DFT}(\cdot)$ and its inverse $\mathsf{iDFT}(\cdot)$ are effectively matrix-vector multiplications.

DFT$(\cdot)$ Let $V \in \mathbb{C}^{\frac{N}{2}\times\frac{N}{2}}$ be defined as in Eq. 9. For $z(X) \in \mathbb{R}[X]/(X^N+1)$, write all its coefficients $\boldsymbol{z}_{coeff}$ as $(\boldsymbol{z}_1||\boldsymbol{z}_2)$. Then

$$\begin{aligned}\mathsf{DFT}(z(X)) = (V||iV)\boldsymbol{z}_{coeff}^{\intercal} &= (V||iV)\begin{pmatrix}\boldsymbol{z}_1^{\intercal}\\ \boldsymbol{z}_2^{\intercal}\end{pmatrix}\\ &= V(\boldsymbol{z}_1^{\intercal}+i\boldsymbol{z}_2^{\intercal})\,.\end{aligned}$$

Therefore, by defining the message complex vector $\boldsymbol{z} := \boldsymbol{z}_1 + i\boldsymbol{z}_2$, $z(X)$ is in fact the *polynomial encoding* of $\boldsymbol{z}$.

iDFT$(\cdot)$ Note that $\mathsf{iDFT}(\cdot)$ is the inverse of $\mathsf{DFT}(\cdot)$, so it is from $\mathbb{C}^{\frac{N}{2}\times\frac{N}{2}}$ to $\mathbb{R}[X]/(X^N+1)$. We can verify that $\bar{V}^{\intercal}V = \frac{N}{2}I_{\frac{N}{2}}$, therefore $\frac{2}{N}\bar{V}^{\intercal}$ is the inverse of V. Then we have $\mathsf{iDFT}(\cdot)$ maps $\boldsymbol{z} \in \mathbb{C}^{\frac{N}{2}\times\frac{N}{2}}$ to the real polynomial whose coefficients are in $(\mathrm{Re}(\frac{2}{N}\bar{V}^{\intercal}\boldsymbol{z})||\mathrm{Im}(\frac{2}{N}\bar{V}^{\intercal}\boldsymbol{z}))$.

SlotToCoeff It turns a CKKS ciphertext encrypting message vector $\boldsymbol{z}$ into another ciphertext encrypting polynomial $z_{int}(X) := \lceil \Delta z(X) \rfloor$, where $z(X)$ has coefficients from vector $(\mathrm{Re}(\boldsymbol{z})||\mathrm{Im}(\boldsymbol{z}))$. By identifying the message vector encrypted in the output ciphertext, we can determine how to apply HE operations to the ciphertext encrypting $\boldsymbol{z}$ to obtain the desired result.

In fact, the output ciphertext is encrypting message vector $\mathsf{DFT}(z(X))$. As discussed above, $\mathsf{DFT}(z(X)) = V\boldsymbol{z}$. Therefore StC is effectively evaluating a HE multiplication between plaintext matrix V and encrypted vector $\boldsymbol{z}$.

CoeffToSlot It turns a CKKS ciphertext encrypting polynomial $z_{int}(X) := \lceil \Delta z(X) \rfloor$ into another ciphertext encrypting message vector $\boldsymbol{z}$. Similarly, the input ciphertext is encrypting message vector $\mathsf{DFT}(z(X)) = V\boldsymbol{z}$. The process of **CoeffToSlot** is effectively an inverse of **SlotToCoeff**. Therefore CtS is evaluating a HE multiplication between plaintext matrix $V^{-1} = \frac{2}{N}\bar{V}^{\intercal}$ and encrypted vector $V\boldsymbol{z}$.

Optimization I: integrate functional bootstrapping in our inference flow. The proposed CKKS functional bootstrapping flow for integer message vector $\boldsymbol{m} \in \mathbb{Z}_p^{N/2}$ is:

$$\frac{\boldsymbol{m}}{p} \xrightarrow{\mathsf{StC}} \frac{m_{int}(X)}{p} \xrightarrow{\cdots} [f(m_0), f(m_1), ..., f(m_{N/2-1})]$$

for arbitrary function $f(\cdot) : \mathbb{Z}_p \to \mathbb{Z}_p$.

In our first functional bootstrapping in attention heads, the input ciphertexts are encrypting matrix $Q^{(h,0)}[K^{(h,0)}]^{\intercal} + Q^{(h,1)}[K^{(h,1)}]^{\intercal} + iC$ in diagonal packing, from CCMM Algorithm 4. We extend the flow by (1) modifying the input ciphertext to encrypt **complex message vector** $\boldsymbol{z}$ instead of integer message vector $\boldsymbol{m}$; and (2) discarding unused complex matrix C.

It suffices to consider a single ciphertext in the encrypted matrix, since all ciphertexts in the encrypted matrix undergo the same sequence of HE operations during functional bootstrapping. Consider such a ciphertext encrypting complex message vector $\boldsymbol{z}$. The first step is mapping $\boldsymbol{z}$ to another complex vector $\tilde{\boldsymbol{z}}$ s.t. $\mathrm{Re}(\tilde{\boldsymbol{z}}) \in \mathbb{Z}_p$, [12] and let $\boldsymbol{m} := \mathrm{Re}(\tilde{\boldsymbol{z}}) \in \mathbb{Z}_p^{N/2}$. Then the modified flow becomes:

$$\begin{aligned}\boldsymbol{z} \to \frac{\tilde{\boldsymbol{z}}}{p} &\xrightarrow{\mathsf{StC\text{-}real}} \frac{m_{int}(X)}{p} \xrightarrow{\mathsf{ModRaise}} \frac{\Delta m(X)}{p} + \frac{k(X)}{p}\cdot Q_0\\ &\xrightarrow{\mathsf{CtS}} \frac{\mathrm{Re}(\tilde{\boldsymbol{z}})}{p} + Q_0\frac{\mathrm{Re}(\boldsymbol{k})}{p}\\ &\xrightarrow{\mathsf{EvalLUT}} [f(m_0), f(m_1), ..., f(m_{N/2-1})]\,.\end{aligned}$$

Here $\mathsf{StC\text{-}real}$ is to discard $\mathrm{Im}(\boldsymbol{z})$ by integrating HE conjugation in StC.

In our second functional bootstrapping in feedforward layers, the input ciphertexts are encrypting matrix $X(W_{\mathsf{fc1}}^{(0)} + iW_{\mathsf{fc1}}^{(1)})$ in column packing (omitting bias for the sake of simplicity). We extend

[12] In implementation, suppose $\mathbb{Z}_p$ is represented by $\{-p/2, ..., 0, 1, ...p/2-1\}$. Suppose $||\mathrm{Re}(\boldsymbol{z})||_\infty$ is bounded by D. This step can be done by $\boldsymbol{z} \mapsto \tilde{\boldsymbol{z}} := \lceil \boldsymbol{z}p/D \rfloor$.

the functional bootstrapping flow by modifying the input ciphertext to encrypt **complex message vector** $\boldsymbol{z}$ instead of integer message vector $\boldsymbol{m}$.

$$\begin{aligned}\boldsymbol{z} \to \frac{\tilde{\boldsymbol{z}}}{p} &\xrightarrow{\mathsf{StC}} \frac{z_{int}(X)}{p} \xrightarrow{\mathsf{ModRaise}} \frac{\lceil \Delta z(X) \rfloor}{p} + \frac{k(X)}{p} \cdot Q_0 \\ &\xrightarrow{\mathsf{CtS'}} \begin{cases} \frac{\mathrm{Re}(\tilde{\boldsymbol{z}})}{p} + Q_0 \frac{\mathrm{Re}(\boldsymbol{k})}{p} \xrightarrow{\mathsf{EvalLUT}} \text{Output ciphertext 1} \\ \frac{\mathrm{Im}(\tilde{\boldsymbol{z}})}{p} + Q_0 \frac{\mathrm{Im}(\boldsymbol{k})}{p} \xrightarrow{\mathsf{EvalLUT}} \text{Output ciphertext 2} \end{cases} \\ &\xrightarrow{\text{Set Concat.}} \text{PCMM with } W_{\mathsf{fc2}} \ .\end{aligned}$$

Here CtS' generates two ciphertexts encrypting the real part and the imaginary part of $\tilde{\boldsymbol{z}}/p$ respectively, using HE conjugation.

Optimization II: Faster matrix-vector multiplication. To further accelerate the multiplication between plaintext matrix V and $\frac{2}{N}\bar{V}^\intercal$, we apply Han et al. (2019) to decompose $V, \bar{V}^\intercal$ into simpler forms. Recall that in BSGS-style plaintext-matrix encrypted-vector multiplication (Halevi & Shoup, 2018), reducing the number of non-empty diagonals in the plaintext matrix can accelerate. Note the specific structure of V as in Eq. 9, the main idea of Han et al. (2019) is to permute the columns of V using the bit-reversal permutation matrix $P \in \{0,1\}^{\frac{N}{2}\times\frac{N}{2}}$ and $P^{-1} = P$. Following the notations in Han et al. (2019); Ju et al. (2024), we write $T_{N/2}$ as the result of bit-reversal permutation on V. Then

$$T_{N/2} = \begin{pmatrix} I_{N/4} & W_{N/4} \\ I_{N/4} & W_{N/4} \end{pmatrix} \begin{pmatrix} T_{N/4} & 0 \\ 0 & T_{N/4} \end{pmatrix} = \cdots ,$$

where W_r is some diagonal matrix. As the number of non-empty diagonals is only 2 in the first matrix, this decomposition can actually reduce the total number of plaintext-ciphertext multiplications and HE rotations, at a cost of higher level consumption. $\bar{V}^\intercal$ is decomposed in a similar way.

In our implementation of functional bootstrapping, StC, StC-real, CtS and CtS are all applying the decomposition to accelerate, using 3 multiplicative levels.

D.2 Optimized Paterson-Stockmeyer polynomial evaluation

In Section 2.2, we introduce the SIMD style $\mathrm{Add}(\cdot,\cdot)$ and $\mathrm{Mult}(\cdot,\cdot)$ of CKKS scheme, which can be used to evaluate a polynomial $P(x)$ on each plaintext slot. However, the ciphertext-ciphertext multiplication function $\mathrm{Mult}(\cdot,\cdot)$ is slow and is one of the main bottlenecks of CKKS scheme. Thus, instead of naively evaluating a polynomial, it is important and necessary to reduce the number of *ciphertext-ciphertext* multiplications. The well-known Paterson-Stockmeyer algorithm (Paterson & Stockmeyer, 1973) has the desired property. It needs only $O(\sqrt{d})$ ciphertext-ciphertext multiplications to evaluate a polynomial with degree d. Thus it is widely used in homomorphic schemes to evaluate polynomial with encrypted input (e.g., Lu et al. (2021); Liu & Wang (2023)). LEAF (Zhang et al., 2025b) proposed multi-thread evaluation of $\{x, x^2, ..., x^k\}$ which reduces the number of ciphertext-ciphertext multiplications from $O(\sqrt{d})$ to $O(\log t + \sqrt{d}/t)$, where t is the number of threads. We include Paterson-Stockmeyer algorithm in Algorithm 1, and the multi-thread evaluation of $\{x, x^2, ..., x^k\}$ in Algorithm 2.

D.3 Discussion of alternative methods of non-linear function evaluations

CKKS functional bootstrapping. Our work does not introduce a new functional bootstrapping technique but integrates it into FHE-based transformer inference. We use the representative and well-established CKKS functional bootstrapping framework of Alexandru et al. (2025) to efficiently evaluate activation functions during Transformer inference. Therefore follow-up works and extensions of Alexandru et al. (2025) can potentially be used to further improve CtrlFormer's performance, especially inference latency. For example, Dumezy et al. (2025) optimizes large lookup tables (up to 20 bits) by replacing Alexandru et al. (2025)'s direct Hermite-interpolation evaluation. Yang et al. (2026) further optimizes the CKKS functional bootstrapping for continuous functions like $ReLU$. Dumezy et al. (2025) and Yang et al. (2026) are complementary rather than directly competitive with the method (Alexandru et al., 2025).

Algorithm 1 Paterson-Stockmeyer algorithm Paterson & Stockmeyer (1973).

1: **Input:** A degree-$(d-1)$ polynomial $P(\cdot) = a_0 + a_1 X + ... + a_{d-1} X^{d-1}$. Data x to be evaluated.
2: **Output:** $P(x)$.
3: Let $k := \lceil d \rceil$. Compute $\{x, x^2, ..., x^k\}$.
4: Use x^k to compute $\{x^{2k}, x^{3k}, ..., x^{k^2-k}\}$.
5: Evaluate $P(x)$ in the following form

$$\begin{aligned} P(x) = &(a_0 + a_1 x + ... + a_{k-1} x^{k-1}) \\ &+ (a_k + a_{k+1} x + ... + a_{2k-1} x^{k-1}) x^k \\ &+ ... \\ &+ (a_{d-k} + a_{d-k+1} x + ... + a_{d-1} x^{k-1}) x^{k^2-k} . \end{aligned}$$

Algorithm 2 Multi-thread computation of $\{x, x^2, ..., x^k\}$.

1: **Input:** Positive integer k, data x.
2: **Output:** $Power(x, k) := \{x, x^2, ..., x^k\}$.
3: Define set $D_l = \{x, x^2, ..., x^l\}$. Initially $D_1 = \{x\}$.
4: **while** $l < k$ **do**
5: $l := l \times 2$.
6: Evaluate $D_l = (D_{l/2} \times x^{l/2}) \cup D_{l/2}$ in multi-thread. $D_{l/2} \times x^{l/2}$ is to multiply each element in $D_{l/2}$ by $x^{l/2}$.
7: **end while**
8: Return $\{x, x^2, ..., x^k\} \subseteq D_{2^{\lceil \log k \rceil}}$.

In CtrlFormer, functional bootstrapping is designed as a distinct and replaceable module. Therefore, improved CKKS functional bootstrapping methods can be incorporated into CtrlFormer without redesigning the overall FHE Transformer inference pipeline.

Softmax alternatives. In addition to Powerformer (Park et al., 2025) and Powersoftmax (Zimerman et al., 2024a), a recent work (Park et al., 2026) also proposes an FHE-friendly softmax alternative. At a high level, XControl is designed as an FHE-friendly softmax replacement with bounded output and efficient evaluation under FHE. This boundedness helps control the dynamic range during encrypted inference and improves numerical stability when evaluating the exponential function. In contrast, Park et al. (2026) uses a cumulant-generating-function-based denominator approximation to avoid maximum and division operations. Its practical behavior still depends on the accuracy of the approximation. Therefore, XControl and Park et al. (2026) represent different tradeoffs between preserving softmax semantics and enforcing a bounded, FHE-oriented activation design.

E Details of HE matrix multiplication

In this section we elaborate the matrix multiplication algorithms of Section 4.3. Our matrix multiplications are the optimized version of Yang et al. (2025). We further elaborate how to make use of the imaginary part of the CKKS slots. This optimization can reduce half of the overhead in CPMM. For the sake of simplicity, we will omit the additive bias as it is straightforward and its time/space cost is almost negligible.

E.1 Packing strategies

We first give a brief description of pack definitions.

Definition E.1. Given a matrix $X \in \mathbb{R}^{m \times d}$, the *column packing* of X consists of d ciphertexts, defined as $\mathrm{Enc}_{col}(X) := \left\{\mathrm{Enc}(\mathbf{x}_0^\intercal), ..., \mathrm{Enc}(\mathbf{x}_{d-1}^\intercal)\right\}$, where $\mathbf{x_j}^\intercal$ is the j-th column of X.

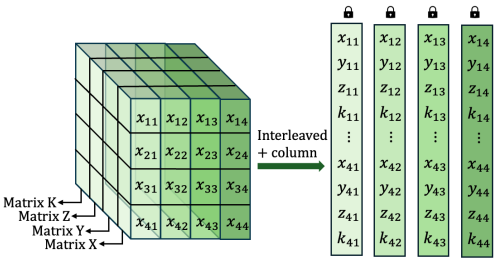

Figure 7: Interleaved column packing.

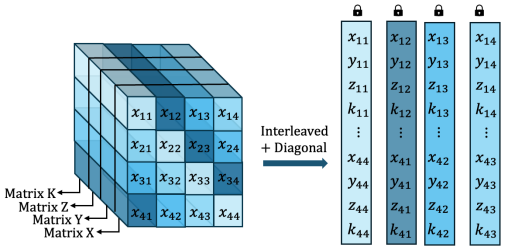

Figure 8: Interleaved diagonal packing.

Definition E.2. Given a square matrix $X \in \mathbb{R}^{m\times m}$, the *diagonal packing* of X consists of m ciphertexts, defined as $\mathrm{Enc}_{diag}(X) := \big\{\mathrm{Enc}(\mathrm{Diag}_0(X)), ..., \mathrm{Enc}(\mathrm{Diag}_{m-1}(X))\big\}$, where $\mathrm{Diag}_j(X)$ is the j-th diagonal of X.

E.2 Interleaved batching

For completeness, we give the definition of interleaved batching for matrices from MOAI (Zhang et al., 2026). Let m be the maximum token length. Given $N/(2m)$ matrices $\{X^{(r)}\}_{r\in[N/(2m)]}$, whose size is $m \times d$, we denote the interleaved batching matrix as $\bar{X} \in \mathbb{R}^{\frac{N}{2}\times d}$. Then we have,

$$\begin{aligned}\bar{\mathbf{x}}_i^{\intercal} := [&x_{i,0}^{(0)T}, x_{i,0}^{(1)T}, x_{i,0}^{(2)T}, ..., x_{i,0}^{(N/(2m)-1)T},\\ &x_{i,1}^{(0)T}, x_{i,1}^{(1)T}, x_{i,1}^{(2)T}, ..., x_{i,1}^{(N/(2m)-1)T},\\ &\cdots\\ &x_{i,m-1}^{(0)T}, x_{i,m-1}^{(1)T}, x_{i,m-1}^{(2)T}, ..., x_{i,m-1}^{(N/(2m)-1)T}]\,,\end{aligned} \tag{11}$$

where the i-th column of $\bar{X}$ is $\bar{\mathbf{x}}_i^{\intercal} \in \mathbb{R}^{N/2}$, which is the interleaved batching vector of all the i-th column of each $X^{(r)}$, for $r \in [N/(2m)]$.

Now we show the definitions of the *interleaved column packing* which we use to batch and encrypt the inputs. We omit the details of *interleaved diagonal packing*, as we focus only on the batching method applied at the input encryption stage. Please refer to MOAI (Zhang et al., 2026) and ARION (Yang et al., 2025) for the correctness.

Definition E.3 (Interleaved Column Packing)**.** Let $\{X^{(r)} \in \mathbb{R}^{m\times d}\}_{r\in[N/(2m)]}$ be a collection of matrices. Their *interleaved batching matrix* is $\bar{X} \in \mathbb{R}^{\frac{N}{2}\times d}$, whose i-th column is defined in Eq. (11) for $i \in [d]$. The *interleaved column packing* of $\bar{X}$ consists of d CKKS ciphertexts

$$\mathrm{Enc}_{\mathrm{col}}(\bar{X}) := \big\{\mathrm{Enc}(\bar{\mathbf{x}}_0^{\intercal}), \mathrm{Enc}(\bar{\mathbf{x}}_1^{\intercal}), \ldots, \mathrm{Enc}(\bar{\mathbf{x}}_{d-1}^{\intercal})\big\},$$

where $\bar{\mathbf{x}}_i^{\intercal} \in \mathbb{R}^{N/2}$ denotes the i-th column of $\bar{X}$. We refer to $\mathrm{Enc}_{\mathrm{col}}(\bar{X})$ as the *interleaved column packing* of the encrypted matrices $\{X^{(r)}\}_{r\in[N/(2m)]}$.

Figure 7 and Figure 8 in show toy examples of interleaved column/diagonal packing.

E.3 Ciphertext-plaintext matrix multiplication

Let $Q^{(h)} = XW_Q^{(h)}$, then the j-th *column* of Q satisfies $\mathbf{q}_j^{\intercal} = \sum_i w_{i,j}\mathbf{x}_i^{\intercal}$. Then the Ciphertext-plaintext matrix multiplication (CPMM) takes column packing input and generates column packing output, as shown in Chen et al. (2024); Zhang et al. (2026); Yang et al. (2025).

In attention block, the CPMM in $\mathbb{R}$ (e.g., Zhang et al. (2026); Yang et al. (2025)) includes computing $\mathrm{Enc}(X)W_*^{(h)} \in \mathbb{R}^{m\times\frac{d}{H}}$, $* = Q, K, V$ and $h \in [H]$. So totally $3 \cdot \frac{d}{H} \cdot H = 3d$ output ciphertexts are computed.

We start with computing $Q^{(h)}$. First define $\tilde{W}_Q^{(h)} \in \mathbb{C}^{d\times\frac{d}{2H}}$. For $h \in [H]$, let $W_Q^{(h)} = (W_Q^{(h,0)}||W_Q^{(h,1)})$, then let

$$\tilde{W}_Q^{(h)} := W_Q^{(h,0)} + i \cdot W_Q^{(h,1)}\,.$$

Algorithm 3 generates the column packing of encrypted output $\text{Enc}(X)\tilde{W}_Q^{(h)}$ which is

$$\text{Enc}(X[W_Q^{(h,0)} + iW_Q^{(h,1)}]) \in \mathbb{C}^{m\times\frac{d}{2H}} \ , \ h \in [H] \ ,$$

which only contains $d/2H$ ciphertexts under column packing. Therefore, applying Algorithm 3 to all H heads of Q,K,V will reduce the number of output ciphertexts to $3 \cdot \frac{d}{2H} \cdot H = 3d/2$, save one half of time cost. And given that CKKS is able to evaluate complex numbers on its slots, this optimization is achieved with negligible additional time cost.

Algorithm 3 Ciphertext-plaintext matrix multiplication with complex numbers in slots.

Input: Encrypted matrix in column packing $\text{Enc}_{col}(X)$, weight matrix $W_Q^{(h)} \in \mathbb{R}^{d\times\frac{d}{H}}$ in head $h \in [H]$.

Output: Encrypted matrix in column packing $\text{Enc}_{col}(Q^{(h,0)} + iQ^{(h,1)})$.

1: Parse $\text{Enc}_{col}(X) := \left\{\text{Enc}(\mathbf{x}_0^\intercal), ..., \text{Enc}(\mathbf{x}_{d-1}^\intercal)\right\}$.
2: Write $W_Q^{(h)} = (W_Q^{(h,0)}||W_Q^{(h,1)})$.
3: Let $\tilde{W} := W_Q^{(h,0)} + iW_Q^{(h,1)} \in \mathbb{C}^{d\times\frac{d}{2H}}$.
4: Let Ans $= \emptyset$.
5: **for** $j \in [d/2H]$ **do**
6: Compute $\text{Enc}(\mathbf{q}_j) := \sum_i w_{i,j}\text{Enc}(\mathbf{x}_i^\intercal)$.
7: Include $\text{Enc}(\mathbf{q}_j)$ in $\text{Enc}_{col}(Q^{(h,0)} + iQ^{(h,1)})$.
8: **end for**
9: Return Ans.

E.4 Ciphertext-ciphertext matrix multiplication

There are two kinds of ciphertext-ciphertext matrix multiplications (CCMM) in BERT. The first is $Q^{(h)}(K^{(h)})^\intercal$ in each head $h \in [H]$. The HE CCMM of $Q^{(h)}(K^{(h)})^\intercal$ is based on the following (e.g., Zhang et al. (2025a),Zhang et al. (2026)):

$$\mathbf{c}_j^{(h)} = \sum_{i=0}^{d/H-1} \mathbf{q}_i^{(h)} \odot \text{Rot}_j(\mathbf{k}_i^{(h)}) \ , \tag{12}$$

where $\mathbf{c}_j^{(h)}$ is the $j-th$ (upper) diagonal of $Q^{(h)}(K^{(h)})^\intercal \in \mathbb{R}^{m\times m}$. The state-of-the-art method by Yang et al. (2025) computes the rotated diagonals instead, by defining $m = BG, B, G = O(\sqrt{m})$:

$$\begin{aligned} &\forall j = b + gB \in [m], b \in [B], g \in [G], \text{Rot}_{-gB}(\mathbf{c}_j^{(h)}) \\ &= \text{Rot}_{-gB}(\mathbf{c}_{b+gB}^{(h)}) = \sum_{i=0}^{d/H-1} \text{Rot}_{-gB}(\mathbf{q}_i^{(h)}) \odot \text{Rot}_b(\mathbf{k}_i^{(h)}) \ , \end{aligned} \tag{13}$$

where $\mathbf{c}_j^{(h)}$ is the j-th (upper) diagonal of $Q^{(h)}(K^{(h)})^\intercal \in \mathbb{R}^{m\times m}$. By computing the rotated diagonals of $Q^{(h)}(K^{(h)})^\intercal$, the rotation steps in the next matrix multiplication with $V^{(h)}$ are pre-computed, saving the number of rotations in this BSGS-style multiplication from $O(m)$ to $O(\sqrt{m})$ (see *double-BSGS* from Chen et al. (2024); Yang et al. (2025)).

We further elaborate how to make use of the imaginary part of the CKKS slots to reduce overhead. Given h, we compute $\text{Enc}_{col}(Q^{(h,0)} + iQ^{(h,1)})$ using Algorithm 3. For K, we slightly modify Line 3 in Algorithm 3: Let $W := W_K^{(h,0)} - iW_K^{(h,1)} \in \mathbb{C}^{d\times\frac{d}{2H}}$. Then the output of Algorithm 3 for K is $\text{Enc}_{col}(K^{(h,0)} - iK^{(h,1)})$. Algorithm 4 gives the CCMM of encrypted matrices $Q^{(h,0)} + iQ^{(h,1)}$ and $K^{(h,0)} - iK^{(h,1)}$.

The output of Algorithm 4 is encrypting matrix $(Q^{(h,0)} + iQ^{(h,1)})(K^{(h,0)} - iK^{(h,1)}) \in \mathbb{C}^{m\times m}$ (in the form of rotated diagonals). Meanwhile we have

$$Q^{(h)}[K^{(h)}]^\intercal = (Q^{(h,0)}||Q^{(h,1)}) \begin{pmatrix} [K^{(h,0)}]^\intercal \\ [K^{(h,1)}]^\intercal \end{pmatrix} \in \mathbb{R}^{m\times m} \ ,$$

Algorithm 4 Ciphertext-ciphertext matrix multiplication $Q^{(h)}[K^{(h)}]^\intercal$ with complex numbers in slots.

Input:
Encrypted matrix in column packing $Q^{(h,0)} + iQ^{(h,1)}$.
Encrypted matrix in column packing $K^{(h,0)} - iK^{(h,1)}$.

Output:
Output rotated encrypted matrix in diagonal packing, encrypting $Q^{(h,0)}[K^{(h,0)}]^\intercal + Q^{(h,1)}[K^{(h,1)}]^\intercal + iC$, where C is another matrix in $\mathbb{R}^{m\times m}$ and is not used in the following steps. The $(j = b + gB)$-th (upper) diagonal in the output is rotated by $-gB$ steps.

1: $\text{Enc}_{col}(Q^{(h,0)} + iQ^{(h,1)}) = \{\text{Enc}(\mathbf{q}_0^\intercal)...\text{Enc}(\mathbf{q}_{\frac{d}{2H}-1}^\intercal)\}$.
2: $\text{Enc}_{col}(K^{(h,0)} - iK^{(h,1)}) = \{\text{Enc}(\mathbf{k}_0^\intercal)..\text{Enc}(\mathbf{k}_{\frac{d}{2H}-1}^\intercal)\}$.
3: Let $\text{Ans}_{diag} = \emptyset$.
4: Let $B = \lceil\sqrt{m}\rceil, G = \lceil m/B\rceil$.
5: **for** $s \in [d/2H]$ **do**
6: **for** $b \in [B]$ **do**
7: Compute $\text{Rot}_b^{\text{HE}}(\text{Enc}(\mathbf{k}_s^\intercal))$.
8: **end for**
9: **for** $g \in [G]$ **do**
10: Compute $\text{Rot}_{-gB}^{\text{HE}}(\text{Enc}(\mathbf{q}_s^\intercal))$.
11: **end for**
12: **end for**
13: **for** $j \in [m]$ **do**
14: Let $j = b + gB$ with $b \in [B], g \in [G]$.
15: Initialize $\text{Enc}(\mathbf{c}_j)$ as the naive ciphertext of $\mathbf{0}$.
16: **for** $s \in [\frac{d}{2H}]$ **do**
17: $tmp := \text{Mult}(\text{Rot}_{-gB}^{\text{HE}}(\text{Enc}(\mathbf{q}_s^\intercal)), \text{Rot}_b^{\text{HE}}(\text{Enc}(\mathbf{k}_s^\intercal)))$.
18: $\text{Enc}(\mathbf{c}_j) = \text{Enc}(\mathbf{c}_j) + tmp$.
19: **end for**
20: Include $\text{Enc}(\mathbf{c}_j)$ in Ans_{diag}.
21: **end for**
22: Return Ans_{diag}.

which is exactly the real part of the output of Algorithm 4.

Using Algorithm 4, the number of rotation for one h is reduced from about $2\sqrt{m} \cdot d/H$ in Yang et al. (2025) to $2\sqrt{m} \cdot d/(2H) = \sqrt{m}d/H$, and the total number of ciphertext-ciphertext multiplication is also reduced from md/H to $md/(2H)$. Note that we can apply *lazy-relinearization* to accelerate in computing $\text{Enc}(\mathbf{c}_j)$ (Line 17,18 in Algorithm 4), reducing the number of relinearizations from $md/(2H)$ to m.

Another kind of CCMM appears in the multiplication of attention weights and matrix V. First, we apply Algorithm 4 to obtain the input to functional bootstrapping: the encrypted complex vectors whose real parts are the upper diagonals of $Q^{(h)}[K^{(h)}]^\intercal$. [13] Before functional bootstrapping, we first apply Algorithm 5 to extract the real part[14] of the output of Algorithm 4, which is exactly $Q^{(h)}[K^{(h)}]^\intercal$. Then we can conclude that after functional bootstrapping the result is an encrypting matrix in diagonal packing. [15]

Let $\text{Enc}_{diag}(S^{(h)})$ be the output of functional bootstrapping, which is the encrypted attention weights, with $S^{(h)} \in \mathbb{R}^{m\times m}$, $V^{(h)} \in \mathbb{R}^{m\times(d/H)}$. We can let $\mathbf{c}_j^\intercal$ be the j-th column of $X^{(h)} := S^{(h)}V^{(h)} \in$

[13] The imaginary parts will be discarded during functional bootstrapping for free.

[14] Note that the factor $1/2$ in Line 5 Algorithm 5 will be merged into the plaintext weights (e.g., W_Q) before this step, so that one level can be saved. When using Algorithm 6, the factor $1/(2i)$ is merged in a similar way. This can save a multiplicative level.

[15] Omitting the factor $1/\sqrt{d/H}$ here, as this factor can be merged in to one plaintext matrix (e.g., W_Q). This can save a multiplicative level.

Algorithm 5 Extract real part of an encrypted matrix.

Input: Encrypted matrix in column/diagonal packing $A + iB \in \mathbb{C}^{m\times d}$.
Output: Encrypted matrix in column/diagonal packing $\mathrm{Re}(A + iB) = A \in \mathbb{R}^{m\times d}$.
1: Parse the input as $\{\mathrm{Enc}(\mathbf{c}_0), ..., \mathrm{Enc}(\mathbf{c}_{d-1})\}$ (either in column or diagonal packing).
2: Let Ans $= \emptyset$.
3: **for** $j \in [d]$ **do**
4: Compute conjugate: $\mathrm{Enc}(\bar{\mathbf{c}}_j)$.
5: Include $(\mathrm{Enc}(\mathbf{c}_j) + \mathrm{Enc}(\bar{\mathbf{c}}_j))/2$ in Ans.
6: **end for**
7: Return Ans.

Algorithm 6 Extract imaginary part of an encrypted matrix.

Input: Encrypted matrix in column/diagonal packing $A + iB \in \mathbb{C}^{m\times d}$.
Output: Encrypted matrix in column/diagonal packing $\mathrm{Im}(A + iB) = B \in \mathbb{R}^{m\times d}$.
1: Parse the input as $\{\mathrm{Enc}(\mathbf{c}_0), ..., \mathrm{Enc}(\mathbf{c}_{d-1})\}$ (either in column or diagonal packing).
2: Let Ans $= \emptyset$.
3: **for** $j \in [d]$ **do**
4: Compute conjugate: $\mathrm{Enc}(\bar{\mathbf{c}}_j)$.
5: Include $(\mathrm{Enc}(\mathbf{c}_j) - \mathrm{Enc}(\bar{\mathbf{c}}_j))/(2i)$ in Ans.
6: **end for**
7: Return Ans.

$\mathbb{R}^{m\times(d/H)}$. It can be verified that

$$\mathbf{c}_j^{(h)\intercal} = \sum_{i\in[m]} \mathrm{Diag}_i(S^{(h)}) \odot \mathrm{Rot}_i(\mathbf{v}_j^{(h)\intercal}) \ ,$$

which requires m HE rotations in HE evaluation. BSGS algorithm can reduce the rotation to $O(\sqrt{m})$ by setting $B, G = \Theta(\sqrt{m}), BG = m$ in the following:

$$\mathbf{c}_j^{(h)\intercal} = \sum_{g\in[G]} \mathrm{Rot}_{gB}\left(\sum_{b\in[B]} \left(\mathrm{Rot}_{-gB}(\mathrm{Diag}_{gB+b}(S^{(h)})) \odot \mathrm{Rot}_b(\mathbf{v}_j^{(h)\intercal})\right)\right).$$

Note that the rotation of diagonals of $S^{(h)}$ can be pre-computed by slightly modifying Algorithm 4 Chen et al. (2024); Yang et al. (2025). Then the total number of HE rotations without using complex slots across $j \in [d]$ and $h \in [H]$ is $H \cdot (G + B) \cdot d/H \approx 2\sqrt{m}d$.

To make use of imaginary parts, we apply Algorithm 3 by writing $V^{(h)} = (V^{(h,0)}||V^{(h,1)})$ and setting $\mathbf{c}_j^{\intercal}$ be the i-th column of $V^{(h,0)} + iV^{(h,1)}$, then the total number of HE rotations becomes $H \cdot (G + B) \cdot d/(2H) \approx \sqrt{m}d$.

The output of Algorithm 7 is converted back to real matrix using a proper order of columns, because the next step is concatenation:

$$X^{\mathrm{Concat}} := (X^{(0)} \parallel X^{(1)} \parallel ... \parallel X^{(H-1)})W_{\mathsf{O}} + \mathbf{1}^{\intercal}\mathbf{b}_{\mathsf{O}} \ ,$$

in which it generates the linear combination of all the columns of $\{X^{(h)}\}_{h\in[H]}$. It is better to separate all the columns in different ciphertexts, instead of packing in complex vectors. Compared with Yang et al. (2025), we use $d/2$ extra conjugations in Algorithm 7.

Remark. To further save a multiplicative level in Algorithm 7, the factor $1/2$ (line 13) and $1/(2i)$ (line 14) can be merged in the plaintext weight matrix W_{O} and the bias $\mathbf{b}_{\mathsf{O}}$. Therefore, Algorithm 7 only consumes 1 level.

Hoisting Halevi & Shoup (2018) introduced a technique to reduce computational overhead when one ciphertext is rotated for many times. Instead of repeatedly decomposing the ciphertext, the decomposed components can be reused across all HE rotations, which reduces the amortized cost of many rotations. Hoisting can accelerate, e.g., the **Baby step** in Algorithm 7.

Algorithm 7 Ciphertext-ciphertext matrix multiplication $S^{(h)}V^{(h)}$ with complex numbers in slots.

Input:

1: Encrypted matrix in rotated diagonal packing: $S^{(h)} \in \mathbb{R}^{m\times m}$, from Algorithm 4:

$$\left\{\text{Enc}(\text{Rot}_{-gB}(\text{Diag}_j(S^{(h)})))\right\}_{j=gB+b\in[m]}.$$

2: Encrypted matrix in column packing: $V^{(h,0)} + iV^{(h,1)} \in \mathbb{C}^{m\times d/(2H)}$.

Output: Encrypted matrix in column packing: $X^{(h)} := S^{(h)}V^{(h)} \in \mathbb{R}^{m\times d/H}$.

3: Let $B := \lceil\sqrt{m}\rceil$, $G := \lceil m/B\rceil$.

4: $\forall j \in [m]$, let $j = b + gB$ with $b \in [B], g \in [G]$.

5: Let $s_j := \text{Enc}(\text{Rot}_{-gB}(\text{Diag}_{gB+b}(S^{(h)})))$ from input.

6: Parse $\text{Enc}_{col}(V^{(h,0)} + iV^{(h,1)})$ as $\{\text{Enc}(\mathbf{v}_0^\intercal), \text{Enc}(\mathbf{v}_1^\intercal), ..., \text{Enc}(\mathbf{v}_{d/2H-1}^\intercal)\}$.

7: **for** $j \in [d/(2H)]$ **do**

8: **Baby step:** For $i \in [B]$, let $rot_i := \text{Rot}_i^{\text{HE}}(\text{Enc}(\mathbf{v}_j^\intercal))$.

9: **Giant step:** Compute the ciphertext of $S^{(h)}\mathbf{v}_j^\intercal$, denoted as ct_j:

$$\text{ct}_j := \sum_{g\in[G]} \text{Rot}_{gB}^{\text{HE}}\left(\sum_{b\in[B]} \text{Mult}(s_j, rot_b)\right).$$

10: Include ct_j in Ans.

11: Extract real and imaginary parts:

12: Compute conjugate of vector encrypted by ct_j, denoted as $\bar{\text{ct}}_j$. It encrypts $\overline{S^{(h)}\mathbf{v}_j^\intercal}$.

13: $\text{ct}_{j,real} = (\text{ct}_j + \bar{\text{ct}}_j)/2$.

14: $\text{ct}_{j,img} = (\text{ct}_j - \bar{\text{ct}}_j)/(2i)$.

15: **end for**

16: Return $\{\text{ct}_{0,real}, \text{ct}_{1,real}, ..., \text{ct}_{d/(2H)-1,real}\} \cup \{\text{ct}_{0,img}, \text{ct}_{1,img}, ..., \text{ct}_{d/(2H)-1,img}\}$.

E.5 CPMM IN FEED FORWARD BLOCK.

Note that the feedforward block can be described as:

$$\text{Layernorm}(X + \text{GELU}(XW_{\text{fc1}} + \mathbf{1}^\intercal\mathbf{b}_{\text{fc1}})W_{\text{fc2}} + \mathbf{1}^\intercal\mathbf{b}_{\text{fc2}}).$$

Similar to Algorithm 3, we can write $W_{\text{fc1}} = (W_{\text{fc1}}^{(0)}||W_{\text{fc1}}^{(1)})$ and define $\tilde{W}_{\text{fc1}} := (W_{\text{fc1}}^{(0)} + iW_{\text{fc1}}^{(1)})$. Then a half of ciphertext-plaintext multiplications are saved. We apply the same method to W_{fc2}. After functional bootstrapping GELU, we extract the real and imaginary parts in slots using $d/2$ extra conjugations.

In summary, we include the comparisons between with/without complex slots in the following tables. The number is counted across **all** H attention heads in **one layer**.

Table 8: Using imaginary parts can reduce the number of ciphertext-plaintext multiplications. d_{fc} is the feedforward hidden dimension s.t. $W_{\text{fc1}} \in \mathbb{R}^{d\times d_{\text{fc}}}$ and $W_{\text{fc2}} \in \mathbb{R}^{d_{\text{fc}}\times d}$.

#ct-pt-mul	$W_{Q,K,V}$	W_{fc1}	W_{fc2}	Total
ARION Yang et al. (2025)	d^2	dd_{fc}	dd_{fc}	$d^2 + 2dd_{\text{fc}}$
Ours	$\frac{d^2}{2}$	$\frac{dd_{\text{fc}}}{2}$	$\frac{dd_{\text{fc}}}{2}$	$\frac{d^2}{2} + dd_{\text{fc}}$

Table 9: Using imaginary parts can reduce the number of ciphertext-ciphertext multiplications.

#ct-ct-mul	$QK^\intercal$	SV	Total
ARION Yang et al. (2025)	md	$\sqrt{m}d$	$(m + \sqrt{m})d$
Ours	$\frac{md}{2}$	$\frac{\sqrt{m}d}{2}$	$\frac{(m+\sqrt{m})d}{2}$

Table 10: Using imaginary parts can reduce the number of HE rotations Rot^{HE}.

$\#\text{Rot}^{\text{HE}}(\cdot)$	$QK^\intercal$	SV	Conj.	Total
ARION Yang et al. (2025)	$2\sqrt{md}$	$2\sqrt{md}$	0	$4\sqrt{md}$
Ours	$\sqrt{md}$	$\sqrt{md}$	$3 \times \frac{d}{2}$	$2\sqrt{md} + \frac{3}{2}d$

F Additional ablation experiments

We conducted additional ablation experiments in which the distillation and task-specific training pipelines are kept fixed, while XControl and DSF are introduced both separately and jointly. As shown in Table 11, we can see that distillation provides a strong baseline, while our complete design further improves the average score from 80.6 to 80.8 under the same distillation setting. In particular, the full model achieves the best average performance among all evaluated variants. This comparison indicates that the performance of our model cannot be attributed solely to the distillation procedure; rather, XControl and DSF are complementary to distillation when used jointly. Besides, XControl and DSF are primarily designed to address different challenges in converting the original Transformer into an FHE-friendly architecture, rather than to independently maximize plaintext accuracy. Therefore, the most relevant result is that their joint integration enables efficient FHE-compatible computation while fully preserving and slightly improving the accuracy of the distilled model.

Table 11: Ablation experiment on distillation.

Model	**RTE**	**MRPC**	**SST-2**	**Avg.**
BERT + Distill	67.1	81.7	93.1	80.6
BERT + DSF + Distill	66.6	80.7	92.8	80.0
BERT + XControl + Distill	67.6	81.7	92.1	80.5
Powerformer	67.2	80.7	92.8	80.2
Ours	**67.5**	**81.8**	**93.0**	**80.8**

G Scale Experiments on more tasks and models

G.1 Results on more tasks

We evaluated our method on three more tasks, QNLI (Wang et al., 2018), MNLI (Williams et al., 2018), and BoolQ (Clark et al., 2019). As Table 12 shows, our method outperforms BERT-base across these three tasks. We also evaluated our method on the longer-context task MultiRC (Khashabi et al., 2018), using sequence lengths of 256 and 512. As shown in Table 13, our approach remains stable with longer sequence length input and outperforms BERT-base with standard LayerNorm in both settings.

Table 12: Performance comparison with BERT-base on more tasks

Model	QNLI	MNLI	BoolQ
BERT-base	0.908	0.837	0.752
Ours	**0.911**	**0.839**	**0.757**

G.2 Results on long-horizon autoregressive generation

We test robustness on long-horizon autoregressive generation where approximation errors may accumulate. To fully evaluate our proposed method, we first use the GPT-2 and Llama as the backbones to evaluate our proposed model. As shown in Table 14, we use standard sliding-window evaluation on WikiText-2 (Merity et al., 2017), the GPT-2 XControl+DSF obtains a PPL of 23.80 with a context window of 128 tokens, compared with 22.52 for GPT-2. The Llama-3.2-1B XControl+DSF obtains 13.78, compared with 10.46 for its teacher. The results show that our model can also

Table 13: Performance comparison with BERT-base on longer-context tasks

Task	Model	EM↑	F1a↑	Avg
MultiRC (seq_len=256)	BERT-base	0.1679	0.6628	0.4154
	Ours	0.1679	0.6630	**0.4155**
MultiRC (seq_len=512)	BERT-base	0.1867	0.6697	0.4282
	Ours	0.1784	0.6789	**0.4287**

effectively be generalized to GPT-2 and Llama, validating the effectiveness of its generalization capability.

Additionally, we evaluate both teacher-forced position-wise stability and free-running autoregressive generation. Table 15 shows three key observations: (1) from the first to the last 128-token position bucket, the student–over–teacher excess-NLL penalty changes by only 0.0286 nat/token for GPT-2 and 0.0017 nat/token for Llama, corresponding to merely 2.90% and 0.17% changes in their respective PPL ratios; (2) on 512-token greedy continuations, Jensen–Shannon divergence is only 5.05% and 4.81% of its theoretical maximum for GPT-2 and Llama, while the probability-mass overlap remains stable or improves (86.29% to 86.65% and 83.41% to 89.68%); (3) final-bucket top-1 agreement reaches 93.46% and 93.75%, with all tokens and activations finite and bounded. Thus, the fixed-constant approximation has negligible horizon-dependent impact on typical decoding decisions through 512 continuation tokens. These results show that our proposed method is still robust on the long-horizon autoregressive generation.

Table 14: Predictive quality on WikiText-2 (Merity et al., 2017) test.

Backbone / Model	**PPL**	**Backbone / Model**	**PPL**
GPT-2 Teacher	22.5156	Llama Teacher	10.4562
GPT-2 Ours	23.8042	Llama Ours	13.7783

Table 15: Ground-truth drift is the change from the first to the last 128-position bucket in teacher-forced student excess NLL over its teacher. Free-running metrics use eight 128-token test prompts followed by 512 student-greedy continuation tokens. JS is normalized by its theoretical maximum $\ln(2)$. Mass overlap is $1 - TotalVariation$.

Backbone	**Ground-truth horizon drift**	**Rollout KL Mean / Median**	**JS** (%)	**Mass overlap** (%) **First → Last**	**Top-1** (%) **First → Last**
GPT-2	+0.0286 nat/token (+2.90% PPL-ratio change)	$0.1705/\mathbf{0.1013}$	5.05	$86.29 \rightarrow 86.65$	$89.84 \rightarrow 93.46$
Llama-3.2-1B	+0.0017 nat/token (+0.17% PPL-ratio change)	$0.1450/\mathbf{0.1191}$	4.81	$83.41 \rightarrow 89.68$	$82.91 \rightarrow 93.75$

We also performed a free-running CNN/DailyMail summarization evaluation under the same summary-only SFT setup used for the WikiText-2 checkpoints. We independently decoded summaries for 500 fixed test documents. After the same article prompt, each model used only its own generated tokens and maintained its own KV cache. Consequently, there is no teacher forcing, no shared generated prefix, and no token-level re-alignment between the teacher and the student. In Table 16 we can observe that the results show that trajectory divergence does not lead to a rapid collapse in downstream generation quality: the student retains 95.4% and 96.7% of the teacher's ROUGE-Lsum performance for GPT-2-small and Llama-3.2-1B, respectively. The ROUGE-Lsum reductions are modest. BERTScore shows the same overall pattern of a limited, rather than catastrophic, reduction. The results show CtrlFormer largely preserves the teacher's downstream summarization quality under fully independent autoregressive decoding. The results show that trajectory divergence does not cause rapid degradation of downstream generation quality: the student retains 95.4% (GPT-2-small) and 96.7% (Llama-3.2-1B) of the teacher's ROUGE-Lsum performance, with only modest reductions. BERTScore shows a similar, limited decline.

Overall, CtrlFormer largely preserves the teacher's downstream summarization quality under fully independent autoregressive decoding.

Table 16: Text generation quality on the evaluation set.

Backbone	Model	ROUGE-1 / 2 / Lsum ↑	Rescaled BERTScore-F1 ↑	Teacher ROUGE-Lsum retained
GPT-2-small	Teacher	35.87 / 14.80 / 33.18	0.2545	–
GPT-2-small	CtrlFormer	34.13 / 13.86 / 31.67	0.2420	**95.4%**
Llama-3.2-1B	Teacher	41.40 / 18.50 / 38.64	0.3261	–
Llama-3.2-1B	CtrlFormer	40.33 / 17.41 / 37.37	0.3092	**96.7%**

Besides, the following qualitative examples illustrate the specific phenomenon: the student and teacher can diverge while both continue to produce coherent and reference-aligned summaries.

GPT-2-small, CNN/DailyMail test index 10869. The teacher and CtrlFormer obtain ROUGE-Lsum scores of 36.84 and 55.86, respectively.

- **Reference:** Concept artist Alex Brady from Cambridge has created beautiful pictures of what space travel might look like. He imagines a space station of tomorrow similar to Las Vegas, with neon lights and plenty of attractions. He said his designs are intended to be a bit more "preposterous and cheerful" than others. He also explores what the space planes that take humans to orbit in future might look like.
- **Teacher:** Artist Alex Brady created a space station in the future. It would have neon lights and plenty of attractions. It would have similar amenities and services to towns and cities on Earth. Future space settlements may also have artificial gravity by rotating around an axis.
- **CtrlFormer:** Artist Alex Brady created beautiful pictures of space travel. He imagines a space station of tomorrow similar to Las Vegas. He has also given his take on how humans might get into orbit in the future on a variety of space planes.

Llama-3.2-1B, CNN/DailyMail test index 3854. The teacher and CtrlFormer obtain ROUGE-Lsum scores of 51.90 and 57.90, respectively.

- **Reference:** Jenson Button was set to start from the back of the grid after a mechanical failure in qualifying. Despite frantic work from McLaren mechanics, they were unable to get his car onto the grid, leaving Button to watch from the sidelines. It was only the third race of the last 272 staged that he has missed.
- **Teacher:** Jenson Button was set to start from the back of the grid in Bahrain. But he broke down in qualifying and was unable to take part in the race. The British driver was forced to watch from the sidelines. He tweeted updates from his chair in the paddock.
- **CtrlFormer:** Jenson Button's McLaren car broke down in qualifying. The British driver was set to start from the back of the grid. But despite frantic work from his McLaren team, Button was forced to watch from the sidelines. Button took to Twitter to update his fans following yet another car failure.

Here, CtrlFormer preserves the central event sequence: the qualifying failure, the intended back-of-grid start, McLaren's unsuccessful repair effort, and Button watching from the sidelines. Thus, although its surface realization quickly departs from the teacher's, the resulting summary remains coherent and closely aligned with the main content of the reference.

Together, the quantitative and qualitative results show that our method preserves most of the teacher's downstream summarization performance.

G.3 RESULTS ON BERT-LARGE

We evaluated our method on BERT-large (24 attention layers) and more tasks (QNLI, BoolQ). As shown in Table 17, our method consistently outperforms the BERT-large on every task.

Table 17: Performance comparison with BERT-large

Model	RTE	MRPC	SST-2	QNLI	BoolQ	Avg
BERT-large	0.722	0.868	0.925	0.902	0.775	0.838
Ours	**0.733**	0.868	**0.928**	**0.913**	**0.790**	**0.846**

G.4 RESULTS ON GPT-2

Our algorithms can be extended to sequential decoding effectively. We report the breakdown of one transformer layer for sequential decoding in GPT-2 with KV-cache in Table 18.

Table 18: Time breakdown of one transformer layer for sequential decoding in GPT-2 with KV-cache.

Step	**Level Change**	**Time Cost**
Compute Q,K	$6 \to 5$	1.858 s
Compute V	$10 \to 9$	0.036 s
$QK^\intercal$	$5 \to 4$	0.061 s
XControl (functional bootstrapping)	$4 \to 9$	17.14 s
XControl$\times$V	$9 \to 8$	3.443 s
Attention output	$7 \to 6$	2.283 s
DSF	$6 \to 5$	0.195 s
FFN projection 1	$5 \to 4$	1.242 s
GeLU FuncBoot (functional bootstrapping)	$4 \to 9$	33.953 s
FFN projection 2	$9 \to 8$	4.461 s
DSF	$8 \to 7$	0.218 s
1 layer	–	**64.858 s**
GPT-2, 12 layers	–	**778.6 s**

H CALIBRATION PERFORMANCE COMPARISON WITH STANDARD SOFTMAX

We evaluated calibration using Expected Calibration Error (ECE) (Guo et al., 2017) and Negative Log Likelihood (NLL) (Hastie et al., 2009). Table 19 indicates that our method achieves lower ECE and NLL than BERT-base with Standard Softmax, demonstrating superior calibration.

Table 19: Calibration Performance Comparisons

Task	Model	ACC↑	AVG_CONF	ECE↓	NLL↓
RTE	Softmax (BERT-base)	0.675	0.966	0.297	1.538
	Ours	**0.715**	0.923	**0.233**	**1.091**
MRPC	Softmax (BERT-base)	0.851	0.962	**0.112**	0.623
	Ours	**0.853**	0.960	0.115	**0.588**
SST2	Softmax (BERT-base)	0.920	0.961	0.043	0.249
	Ours	**0.924**	0.934	**0.018**	**0.247**