# CoQuad: Co-Optimized Quadratic Activations for Efficient Private Transformer Inference

**Anonymous authors**
Paper under double-blind review

## Abstract

Ensuring private inference for Transformer architectures remains challenging, with nonlinear activations serving as a key bottleneck. Quadratic replacement offers inexpensive secure evaluation, but recovering model accuracy requires adaptation. We show that the role of quadratic coefficients depends on the FFN architecture: they can be absorbed into adjacent affine maps in biased standard FFNs, whereas nondegenerate bias-free gated channels retain a coefficient ratio invariant under projection reparameterization. MPCFormer's fixed polynomial prescribes this ratio; CoQuad allows it to adapt by jointly learning layer-wise quadratic coefficients and network weights. DirectFit supplies data-dependent initialization, and Co-Optimize performs teacher-guided joint adaptation while retaining quadratic secure evaluation. On ImageNet-1K, full replacement in ViT-B/16 achieves 79.83% Top-1 accuracy (1.62 percentage points below the 81.45% teacher), exceeding a ViT adaptation of the full MPCFormer method by 1.42 percentage points. Llama experiments extend the evaluation to bias-free SwiGLU: with the same 16 layers replaced and the same 1,500 training steps, CoQuad reaches 72.6% ARC-Easy accuracy, compared with 36.5% for MPCFormer's fixed polynomial. Compared with SHAFT, quadratic GeLU evaluation is estimated to be 47–59$\times$ faster and reduces communication by 62$\times$; with the other operators unchanged, measured end-to-end speedups over SHAFT range from $1.227\times$ to $1.530\times$. These results demonstrate the practical value of joint coefficient and network adaptation across standard and gated Transformer FFNs.

## 1 Introduction

Privacy-preserving Transformer inference must protect both user inputs and model parameters while keeping deployment costs practical. Secure Multi-Party Computation (MPC) and Homomorphic Encryption (HE) support this goal, but nonlinear activations such as GeLU and SiLU remain costly (Li et al., 2023; Lu et al., 2025; Kei & Chow, 2025; Hao et al., 2022). Piecewise approximations preserve activation behavior at the expense of comparison-based evaluation, whereas low-degree global polynomials offer inexpensive arithmetic evaluation but can substantially degrade model accuracy.

**Seeking PPML-friendly activation: challenges** This trade-off gives two challenges: C1, the usability bottleneck of secure evaluation, and C2, polynomial convergence difficulty after replacement. Square activations address C1 (Gilad-Bachrach et al., 2016), but a single quadratic cannot reproduce the asymmetric behavior of GeLU or SiLU. Static polynomial substitution therefore faces a degree–accuracy trade-off (Zimerman et al., 2024); piecewise alternatives retain comparison overhead (Lu et al., 2025; Pang et al., 2024). MPCFormer (Li et al., 2023) established that distillation can recover accuracy after replacing GeLU with a fixed quadratic. This motivates adapting the network to inexpensive activations, while leaving a further question: does weight adaptation alone cover the adjustments available through activation coefficients?

**Why learn quadratic coefficients?** We show that the answer depends on the FFN architecture (Proposition 1). In standard FFNs with biases in both adjacent affine maps, nondegenerate quadratic coefficients can be absorbed into those maps. Explicit coefficients therefore provide an initialization and optimization parameterization within the same function class. In a bias-free gated channel,



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.

however, two quadratics with nonzero coefficients are interchangeable through projection reparameterization if and only if their ratios $ac/b^2$ agree. MPCFormer's fixed polynomial prescribes this ratio as 1, whereas trainable coefficients allow it to change. Consequently, fixed coefficients constrain the available channel-wise quadratic interactions in SwiGLU even when the projections are trainable. This architectural distinction motivates adapting activation coefficients together with network weights.

**Our solution.** Our central premise is that local quadratic fitting alone is insufficient: the network must adapt to the activation it will use. We propose CoQuad, which combines explicit, trainable layer-wise quadratics with teacher-guided network adaptation. Building on MPCFormer's replacement-and-distillation approach, CoQuad permits changes to both the backbone and the coefficient relationships that a fixed quadratic prescribes. This provides direct optimization directions in biased standard FFNs and adjustable channel-wise shape parameters in bias-free gated FFNs, while retaining quadratic activation evaluation.

*DirectFit* initializes each quadratic from calibration inputs collected on the progressively replaced student, accounting for changes introduced upstream. *Co-Optimize* then jointly updates backbone weights and coefficients under teacher supervision, allowing them to depart from the local fits to recover task-level behavior. Adaptation takes place offline; the resulting coefficients are fixed during secure inference. Together, these components target C2 without reintroducing comparison-based activation evaluation.

**Outcomes** We evaluate accuracy preservation in standard and gated Transformer FFNs alongside secure-inference benefits. On ImageNet-1K, full replacement in ViT-B/16 reaches 79.83% Top-1 accuracy, 1.62 percentage points below the reevaluated teacher (81.45%) and 1.42 points above the full MPCFormer method adapted to ViT. A ViT ablation measures a 3.53-point gain from training coefficients alongside the backbone; a separate BERT initialization ablation shows a 1.81-point gain from DirectFit with coefficients frozen in both variants. These results demonstrate practical adaptation benefits in architectures where coefficient absorption is possible.

On Llama-2, we apply MPCFormer and CoQuad under the same setting, replacing the same 16 layers and training for the same 1,500 steps. CoQuad reaches 72.6% ARC-Easy accuracy, whereas MPCFormer with its fixed polynomial reaches 36.5% (Section 6.2). These results extend the evaluation to bias-free SwiGLU.

Compared with SHAFT (Kei & Chow, 2025), quadratic GeLU evaluation is estimated to be $47\times$–$59\times$ faster and reduces communication by $62\times$. With other operators unchanged, measured end-to-end speedups range from $1.227\times$ to $1.530\times$. The operator savings come from the quadratic form; CoQuad supplies the offline adaptation needed to make this inexpensive form useful.

**Contributions.** We characterize quadratic coefficient absorption across FFN architectures, establishing the complete equivalence criterion for nondegenerate bias-free gated channels and identifying the ratio constraint imposed by fixed coefficients. We develop CoQuad to combine layer-wise initialization with joint coefficient–backbone adaptation, preserving the low-degree secure-evaluation structure. Experiments assess accuracy recovery in ViT, BERT, and Llama and secure-inference benefits in SHAFT and BumbleBee. Our code is available at: `https://anonymous.4open.science/r/PolyTransformer-Submission-6C9F`.

## 2 BACKGROUND

**Secure inference costs** Secure MPC and HE enable inference on secret-shared or encrypted data. In arithmetic MPC, additions are local, whereas multiplication of secret values requires interaction, commonly using Beaver triples (Damgård et al., 2012); evaluating GeLU or SiLU introduces additional approximation or comparison costs. In HE schemes such as CKKS (Cheon et al., 2017), low-degree polynomials limit the multiplicative depth needed to approximate activations. These costs motivate our focus on quadratic FFN activations; Appendix F provides the broader protocol context.

**Threat model** We consider two-party private inference between a server holding model weights and a client holding a private input, using a semi-honest secure-inference backend. The model architecture

and quadratic coefficients are public; the underlying protocol must protect the remaining model weights and client input beyond the prescribed output. This is a protocol-level privacy requirement, not a guarantee against model extraction from prediction outputs.

**Target activations** We replace GeLU in ViT FFNs (Dosovitskiy et al., 2020; Hendrycks & Gimpel, 2016) and the SiLU component of Llama's SwiGLU FFNs (Touvron et al., 2023; Shazeer, 2020), retaining the multiplicative gate. Replacement is restricted to the selected layers and does not remove other nonlinear operators, such as attention Softmax. Standard FFN equations and activation definitions are deferred to Appendix F.

## 3 CoQuad: Architectural Motivation and Overview

A quadratic offers inexpensive secure evaluation but cannot reproduce the asymmetric tails of GeLU or SiLU. Network adaptation must therefore accommodate the changed activation. The question is whether updating network weights alone covers the adjustments available through the quadratic coefficients. We show that the answer depends on the surrounding FFN architecture, motivating joint adaptation of weights and coefficients.

**Quadratic coefficients and architecture** Let $p(u) = au^2 + bu + c$ act elementwise, with one coefficient triplet shared within each replaced layer. Using column vectors, we compare deterministic FFN maps

$$\begin{aligned} F_p(x) &= W_2 p(W_1 x + \beta_1) + \beta_2, \\ G_p(x) &= W_d\big[p(W_g x) \odot (W_u x)\big]. \end{aligned} \tag{1}$$

The first has biases in both adjacent affine maps, as in ViT/BERT; the second retains the bias-free multiplicative structure of Llama's SwiGLU (Devlin et al., 2019; Dosovitskiy et al., 2020; Touvron et al., 2023; Shazeer, 2020). A gated channel has the form $h_p(x) = p(g^\top x)(v^\top x)$. Its quadratic activation produces cubic, quadratic, and linear interactions in $x$.

**Proposition 1** (Architecture-dependent coefficient absorption)**.** Let $q(u) = a_0u^2 + b_0u + c_0$ be a fixed quadratic. **(i) Biased standard FFNs.** If $a, a_0 \neq 0$, $F_p$ and $F_q$ have the same function class at the same hidden width when both adjacent affine maps may be adjusted. **(ii) Bias-free gated channels.** If all six coefficients are nonzero and $g, v \neq 0$, there exist projection vectors $\widetilde{g}, \widetilde{v}$ such that $h_p(x) = q(\widetilde{g}^\top x)(\widetilde{v}^\top x)$ for all $x$ if and only if

$$\chi(p) := \frac{ac}{b^2} = \frac{a_0c_0}{b_0^2} =: \chi(q). \tag{2}$$

*Proof.* For (i), set $\delta = b/(2a)$, $\kappa = c - b^2/(4a)$, and define $\delta_0, \kappa_0$ analogously for $q$. Completing the square gives, with $\alpha = a/a_0$,

$$p(u) = \alpha q(u + \delta - \delta_0) + \kappa - \alpha\kappa_0. \tag{3}$$

Keep $W_1$ unchanged and set $\widetilde{\beta}_1 = \beta_1 + (\delta - \delta_0)\mathbf{1}$, $\widetilde{W}_2 = \alpha W_2$, and $\widetilde{\beta}_2 = \beta_2 + (\kappa - \alpha\kappa_0)W_2\mathbf{1}$, where $\mathbf{1}$ has the hidden dimension. This realizes $F_p$ using $q$; exchanging $p$ and $q$ gives the converse.

For (ii), equality of the linear terms forces $\widetilde{v} = (c/c_0)v$. Equality of the quadratic terms then forces $\widetilde{g} = (bc_0/(b_0c))g$, by cancellation of the nonzero polynomial $v^\top x$. Finally, equality of the cubic terms requires

$$a = a_0\frac{c}{c_0}\left(\frac{bc_0}{b_0c}\right)^2 \quad \Longleftrightarrow \quad \chi(p) = \chi(q). \tag{4}$$

Conversely, when the ratios agree, these two projection choices match all three homogeneous terms. The scalings are forced by equality, rather than imposed as a restriction on the allowable projections. □

For a nonzero target channel with $c = 0$, a fixed $c_0 \neq 0$ also cannot match it: eliminating the linear term forces $\widetilde{v} = 0$, making the entire fixed channel zero. The channel-wise criterion concerns absorption within a channel; a full FFN additionally permits cross-channel mixing.

**The limitation of fixed coefficients** MPCFormer's polynomial $q(u) = \frac{1}{8}u^2 + \frac{1}{4}u + \frac{1}{2}$ fixes $\chi(q) = 1$ (Li et al., 2023). Proposition 1(ii) shows that adjusting the projections of a nondegenerate gated channel cannot change this ratio. Indeed, for $a, b \neq 0$, setting $z = (a/b)u$ gives $p(u) = (b^2/a)(z^2 + z + \chi(p))$: after absorbing input and output scales, one shape parameter remains. Fixed coefficients prescribe it, whereas CoQuad allows it to adapt separately in each replaced layer. Thus, the coefficient redundancy of biased standard FFNs does not extend to individual bias-free gated channels.

**From the analysis to CoQuad** CoQuad jointly adapts the backbone and explicit quadratic coefficients under teacher supervision. In biased standard FFNs, these coefficients expose direct optimization directions within an equivalent function class; in bias-free gated channels, they also allow adjustment of the ratio in eq. (2). We optimize $(a_\ell, b_\ell, c_\ell)$ directly, rather than explicitly optimizing the ratio, and retain one triplet per replaced layer. After training, the coefficients are fixed for inference: learning them does not increase the polynomial degree or introduce comparison-based activation protocols.

Figure 1 summarizes the procedure. DirectFit (section 5) supplies a data-dependent starting point by progressively fitting each quadratic on the partially replaced student's activation inputs and aligning its output scale. Co-Optimize (section 4) then jointly updates the backbone and coefficients to recover network-level behavior. Local fitting provides initialization; task-level adaptation can subsequently move the coefficients beyond that fit, including changing the gated-channel ratio. Implementation details are provided in section 5.

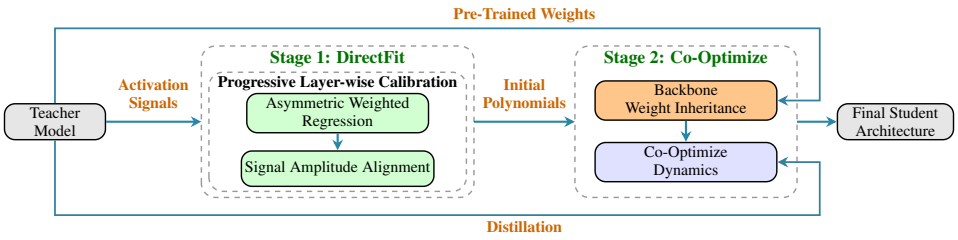



Figure 1: CoQuad pipeline.

## 4 Co-Optimize Distillation

A low-degree polynomial need not reproduce the original activation pointwise if the network can adapt to its behavior. MPCFormer (Li et al., 2023) adapts network weights around a prescribed quadratic; Co-Optimize additionally adapts the quadratic itself. Starting from DirectFit's layer-wise initialization (section 5), we jointly train the backbone and activation coefficients to recover teacher behavior while retaining degree-two secure evaluation.

**Joint coefficient–backbone adaptation** Let $\theta$ denote the student's non-polynomial parameters, including weights and biases, and $\phi = \{(a_\ell, b_\ell, c_\ell)\}_{\ell \in \mathcal{R}}$ the coefficients of replaced layers. Each layer uses $p_\ell(x) = a_\ell x^2 + b_\ell x + c_\ell$, with one triplet shared across its activation coordinates. The student inherits the pretrained backbone; only selected activations are replaced. With a frozen teacher $T$, we optimize

$$\min_{\theta,\phi} \; \mathbb{E}_{\xi \sim \mathcal{D}} \left[ \mathcal{L}_{\text{adapt}} \big( S(\xi; \theta, \phi), T(\xi); \xi \big) \right]. \tag{5}$$

Here $\xi$ includes the input and any training targets. Both parameter sets receive gradients from the same adaptation objective and are updated together. Coefficients are neither frozen nor periodically refitted, and the local DirectFit loss is not retained. Consequently, the learned activation can depart from its initial approximation whenever this improves whole-network behavior.

**Why retain trainable coefficients?** Proposition 1 gives two architectural interpretations of this common procedure. In biased standard FFNs, nondegenerate quadratic activations define the same

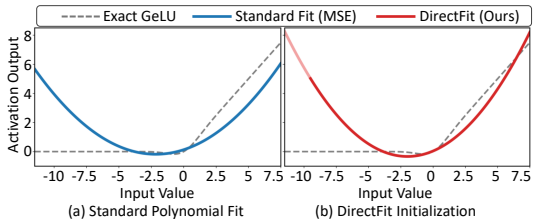



Figure 2: DirectFit better preserves positive signals than standard fitting.

function class: explicit coefficients expose layer-wise adjustment directions without adding representational capacity. Their algebraic absorption into neighboring layers does not make the resulting optimization trajectories identical. In bias-free SwiGLU, replacing only SiLU gives $p_\ell(u) \odot v$, preserving the second input-dependent branch. For nonzero coefficients, the channel-wise ratio $a_\ell c_\ell / b_\ell^2$ cannot be changed by the projection reparameterization characterized in Proposition 1. MPCFormer's fixed polynomial restricts this ratio to one; jointly learning the coefficients permits it to adapt. We train $(a_\ell, b_\ell, c_\ell)$ directly, without computing or regularizing the ratio, so optimization also permits zero coefficients. This uses the additional channel-wise freedom without requiring a claim about the expressivity of the entire multi-channel network.

**Implementation** For ViT, the adaptation objective is logit MSE, $\mathcal{L}_{\text{ViT}} = (BK)^{-1}\|Z^S(\theta,\phi) - Z^T\|_F^2$, on the same batch of $B$ inputs and $K$ classes. No label cross-entropy is added; AdamW updates the backbone and coefficients after gradient accumulation and clipping. For Llama, the same joint parameterization is trained with output-distribution distillation, next-token cross-entropy, and hidden-state MSE. The current Llama-2 configuration uses $\mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{CE}} + 0.1\mathcal{L}_{\text{hidden}}$, where KL is directed from teacher to student and hidden-state MSE is averaged across layers and valid tokens. These task-specific losses instantiate the same joint-adaptation objective.

After training, each layer's coefficients are fixed public constants for secure inference. Learning them adds no activation degree or comparisons: evaluating $a_\ell x^2 + b_\ell x + c_\ell$ still requires one secret–secret multiplication per element. The fixed-coefficient control is evaluated in section 6.2; implementation details are in Appendices C.2 and B.4.

## 5 DirectFit

Learning layer-specific activations also calls for a layer-specific starting point. DirectFit initializes each quadratic from the actual input distribution encountered by the student, before the joint adaptation in section 4. Whereas MPCFormer (Li et al., 2023) uses prescribed coefficients and other approaches fit under Gaussian priors (Peng et al., 2023a), we fit the original activation on calibration inputs from the progressively replaced network. This addresses initialization; it does not require a quadratic to reproduce the original function globally or preserve network accuracy without training.

**Layer-wise fitting based on real data distribution** We visit selected layers in network order. At each layer, we collect scalar activation inputs from a small calibration set, fit a weighted quadratic, align its output scale, and immediately install it. The next layer is therefore calibrated after upstream replacements, rather than on the unchanged teacher's distribution. Each triplet reflects both its layer's inputs and the cumulative changes already introduced into the student. Algorithm 1 in Appendix E.5 gives the procedure.

**Maintaining positive semi-axis signals** An unweighted fit can underestimate positive responses (Figure 2). For sampled inputs $x_{\ell i}$ and original activation $\sigma_\ell$, we instead solve

$$\widehat{\phi}_\ell = \arg\min_{a,b,c} \sum_i \left[w_{\ell i}\big(a x_{\ell i}^2 + b x_{\ell i} + c - \sigma_\ell(x_{\ell i})\big)\right]^2, \tag{6}$$

where $w_{\ell i} = \lambda > 1$ for $x_{\ell i} > 0$ and $w_{\ell i} = 1$ otherwise. These weights multiply residuals, as in our implementation; the positive-side squared errors therefore receive a factor of $\lambda^2$. This prioritizes positive-response reconstruction during initialization without requiring an exact activation match.

Table 1: Latency (s) vs. SHAFT (Poly): estimated GeLU and measured end-to-end (E2E). Softmax is unchanged.

| | GeLU (est.) | | | E2E (meas.) | | |
|---|---|---|---|---|---|---|
| **Network** | **Poly** | **CoQuad** | **Gain** | **Poly** | **CoQuad** | **Gain** |
| 25 Gbps, 0.5 ms | 5.77 | 0.12 | 48.803× | 31.72 | 25.85 | 1.227× |
| 10 Gbps, 0.5 ms | 7.20 | 0.15 | 46.634× | 35.55 | 28.60 | 1.243× |
| 1 Gbps, 0.5 ms | 58.87 | 1.05 | 56.200× | 181.35 | 123.30 | 1.471× |
| 400 Mbps, 4 ms | 146.49 | 2.57 | 56.889× | 432.93 | 288.81 | 1.499× |
| 200 Mbps, 4 ms | 290.89 | 4.91 | 59.260× | 837.17 | 551.74 | 1.517× |
| 100 Mbps, 8 ms | 581.22 | 9.93 | 58.516× | 1656.06 | 1084.39 | 1.527× |
| 50 Mbps, 16 ms | 1161.09 | 19.54 | 59.411× | 3295.59 | 2154.02 | 1.530× |

Table 2: ViT-B/16 ImageNet-1K accuracy.

| Variant | Top-1 (%) |
|---|---|
| Complete models | |
| Teacher | 81.45 |
| Hybrid-6* | 81.14 |
| Full-12 | 79.83 |
| MPCFormer | 78.41 |
| DirectFit, 10 epochs | |
| Frozen coefficients | 76.30 |
| Trainable (Full-12) | 79.83 |

Table 2: *Hybrid-6 uses the original evaluation (teacher: 81.42%); other complete-model results use the common reevaluation. Full-12 is repeated in the ablation group.

After weighted fitting, we align output amplitude by multiplying all coefficients by $\mathrm{std}(Y_{target})/(\mathrm{std}(Y_{pred}) + 10^{-6})$, as specified in algorithm 1. The motivation for this post-fitting rescaling is expanded in Appendix E.4.

**Implementation** Both current ViT and Llama pipelines use progressive DirectFit with $\lambda = 10$ and up to 50,000 sampled scalar inputs per target layer. ViT fits GeLU on a calibration image batch. Llama fits SiLU on the outputs of `gate_proj` using calibration text, excludes padded tokens, and leaves the other SwiGLU branch intact. The same fitting and scale-alignment steps therefore apply to both architectures.

DirectFit chooses the initial coefficients from local distributions; Co-Optimize then updates those coefficients and the backbone using a whole-network objective. Together, they remove the restriction to a prescribed quadratic throughout adaptation while preserving its secure-evaluation cost.

## 6 Experiments

We evaluate two requirements for practical quadratic activation replacement: reducing secure-inference costs and preserving model accuracy. We first quantify operator-level and end-to-end savings, then organize the accuracy results around Proposition 1: adaptation within an unchanged function class for biased standard FFNs, and joint coefficient–backbone adaptation in bias-free SwiGLU, where fixed coefficients constrain the channel-wise ratio $ac/b^2$.

### 6.1 Secure-Inference Efficiency

We evaluate the latency and communication benefits of quadratic GeLU replacement at both the GeLU-operator and end-to-end ViT levels. This separates the direct activation savings from the overall system benefit. Secure-inference hardware specifications are provided in Appendix C.1.

We compare against SHAFT's standard polynomial implementation (`poly`) using ViT-B/16 (`google/vit-base-patch16-224`) with $224 \times 224$ RGB inputs. Both implementations use identical hardware and the seven network settings in Table 1, ranging from 25 Gbps/0.5 ms to 50 Mbps/16 ms. We report online latency, communication bytes exchanged between parties, and synchronization rounds, excluding offline preprocessing. GeLU costs cover all GeLU layers in one full-model inference.

The GeLU latency estimates in Table 1 follow SHAFT's network-aware model, $T_{\text{total}} = T_{\text{comp}} + 2\,\text{Comm}/\text{Bandwidth} + \text{Rounds} \cdot \text{Latency}$. The end-to-end results in the table and below are measured wall-clock times.

The GeLU columns of Table 1 isolate the direct cost of GeLU evaluation. Across all network settings, CoQuad achieves an estimated $47\times$–$59\times$ GeLU latency speedup over SHAFT's standard `poly` implementation. Aggregated over all GeLU layers in one ViT-B/16 inference, communication volume is reduced by approximately $62\times$, while synchronization rounds fall from 240 to 12 ($20\times$). These are GeLU-only costs, not whole-model communication reductions. The savings arise from the global

quadratic form, whose secure evaluation requires one secret–secret multiplication per activation element; CoQuad's contribution is making this form accurate enough to use.

With Softmax unchanged, the measured end-to-end speedup over SHAFT ranges from $1.227\times$ to $1.530\times$ (Table 1). The smaller gains relative to GeLU evaluation reflect the remaining attention, linear-layer, and other inference costs. The larger gains under more constrained network conditions are consistent with the reduced communication volume and synchronization rounds.

In a separate end-to-end ViT-B/16 MPC experiment, we combine quadratic GeLU replacement with MPCFormer's 2Quad Softmax approximation (Li et al., 2023). Within this separate experiment, the baseline and modified model use identical hardware and network conditions, without imposed bandwidth throttling. The combination reduces measured online latency from 25.110 s to 9.805 s ($2.56\times$ speedup) and total communication from 19.272 GB to 2.407 GB (an $87.5\%$ reduction), relative to the baseline using standard secure GeLU and the original Softmax implementation. These gains reflect the combined GeLU and Softmax replacements.

**Compatibility with BumbleBee** To evaluate compatibility with an HE-based secure-inference backend, we test our method in BumbleBee. In BumbleBee's HE-based ViT-B16 pipeline, replacing GeLU with CoQuad yields $5.80\times$–$10.78\times$ GeLU speedups and up to $1.22\times$ end-to-end speedup; the complete runtime and communication results are in Appendix B.6.

### 6.2 Model Accuracy Preservation

Proposition 1 distinguishes two roles for learnable coefficients. In biased standard FFNs, they change the training parameterization without enlarging the function class. In bias-free SwiGLU, they also allow the channel-wise coefficient ratio to change. We examine these settings separately using the existing adaptation and ablation results.

**Standard FFNs: ViT** On ImageNet-1K, we initialize ViT-B/16 students from the pretrained teacher and replace either the first six GeLU layers (Hybrid-6) or all twelve (Full-12). Table 2 shows that Full-12 reaches 79.83% Top-1 after 10 epochs, compared with 81.45% for the teacher and 78.41% for the complete MPCFormer adaptation (Li et al., 2023). This is a 1.62-point drop from the teacher and a 1.42-point improvement over MPCFormer. The methods use their respective training recipes and a common reevaluation procedure; this comparison measures complete-method performance. Hybrid-6 retains accuracy within 0.28 points of its original-evaluation teacher (81.14% versus 81.42%). Training settings and offline costs are in Appendices C.2 and D.1, with configurations in our anonymized code release.

**Adaptation and explicit coefficients** Local approximation alone does not preserve network behavior: DirectFit without subsequent training yields only 0.27% accuracy with three replaced layers and 0.08% with full replacement (Appendix B.1). After adaptation, freezing DirectFit coefficients gives 76.30%, while jointly training them with the backbone gives 79.83%, a 3.53-point improvement (Table 2). Both variants use 10 epochs and logit-MSE distillation. Thus, even where coefficients are algebraically absorbable, keeping them explicit and trainable is useful under the evaluated training budget. This is an optimization benefit, not evidence of additional representational capacity. The comparison includes the effect of exempting coefficients from weight decay; detailed settings are in Appendix B.2.

**Initialization: BERT evidence** On BERT-Base/RTE, using DirectFit in place of MPCFormer's coefficient initialization improves accuracy from 71.84% to 73.65% under its training recipe. Both variants keep coefficients frozen, isolating the initialization change in this experiment (Appendix B.3). Together, the standard-FFN results show that initialization and training parameterization matter even when the underlying quadratic function class is unchanged.

Transfer to CIFAR-10 is reported in Appendix B.7.

**Bias-free SwiGLU: Llama** We next evaluate Llama-2 (Touvron et al., 2023) on ARC-Easy using log-likelihood accuracy. Here, Proposition 1 identifies a coefficient ratio that cannot be absorbed within a single bias-free gated channel: MPCFormer's fixed quadratic imposes $ac/b^2 = 1$, whereas

Table 3: Llama-2 ARC-Easy accuracy. All runs are trained for 1,500 steps; drops are relative to the unreplaced baseline (75.6%).

| Method | Replaced layers | Baseline (%) | Acc. (%) | Drop (pp) |
|---|---|---|---|---|
| MPCFormer | 8/32 (25%) | 75.6 | 53.7 | 21.9 |
| MPCFormer | 16/32 (50%) | 75.6 | 36.5 | 39.1 |
| CoQuad | 16/32 (50%) | 75.6 | **72.6** | **3.0** |

CoQuad can adapt this ratio. We compare the MPCFormer method, which keeps its prescribed polynomial $0.125x^2 + 0.25x + 0.5$ fixed throughout training, with CoQuad, initialized by DirectFit and jointly trained with Co-Optimize. Both methods are evaluated under the same setting: SiLU is replaced in the same 16 alternating layers, the multiplicative branch is preserved, and both are trained for the same 1,500 steps.

Table 3 shows substantially better accuracy recovery with CoQuad. Under the identical setting of 16 replaced layers and 1,500 training steps, MPCFormer reaches 36.5%, whereas CoQuad reaches 72.6%. Even when MPCFormer replaces only eight layers, its accuracy drop remains 21.9 points, compared with 3.0 points for CoQuad at twice the replacement count. These results support joint coefficient–backbone adaptation beyond biased standard FFNs and are consistent with the channel-wise ratio constraint identified in Proposition 1: with the same layers replaced and the same training budget, the difference between the two methods lies in whether the quadratic coefficients are fixed or learned. Additional checkpoints, normalized accuracies, and Llama-3 results are in Appendix B.4; Llama activation-cost estimates are in Appendix B.5.

## 7 Discussion

The role of coefficient learning depends on architecture. In biased standard FFNs, Proposition 1 establishes representational equivalence, while the ViT and BERT results show the practical value of training parameterization and initialization. In bias-free SwiGLU, the proposition identifies a channel-wise coefficient ratio that fixed-polynomial projection reparameterization cannot change. The Llama results support joint adaptation in this setting: with the same replaced layers and the same training budget, CoQuad recovers substantially more accuracy than the MPCFormer method with its fixed polynomial.

Our experiments support full GeLU replacement in ViT and partial SiLU replacement in Llama. They do not establish that every layer or Transformer can be replaced without loss. Remaining nonlinear operators limit end-to-end acceleration, even when the replaced activations become inexpensive. Choosing the replacement schedule therefore remains a model- and task-dependent trade-off.

Higher-degree polynomial replacements for LLMs remain a direction for future work; Appendix D.2 discusses the potential trade-off between expressiveness, multiplication cost, and communication rounds.

## 8 Related Work

Polynomialization offers a backend-friendly path to resolve the complexity of communication C1 (Usability Bottleneck) by transforming the network into an arithmetic computation graph via polynomial surrogates. Polynomial evaluation avoids comparison-based interval selection, but its multiplication depth still determines interaction costs in MPC and contributes to the depth budget in HE. Recovering accuracy after replacement remains a challenge (C2); local approximation quality alone does not ensure successful network adaptation. *CryptoNets* (Gilad-Bachrach et al., 2016) pioneered using $z^2$ to address C1 in shallow networks. NAS-based methods such as *Delphi* (Mishra et al., 2020), *CryptoNAS* (Ghodsi et al., 2020), *DeepReduce* (Jha et al., 2021), and *PASNet* (Peng et al., 2023b) attempted to balance these constraints by selectively linearizing layers during optimization.

SMART-PAF (Tong et al., 2024) similarly uses data-aware coefficient tuning and model adaptation to recover accuracy after polynomial replacement. It targets ReLU and MaxPooling in CNNs under

FHE, whereas CoQuad focuses on quadratic activation replacement in Transformers for private inference.

MPCFormer (Li et al., 2023) replaces GeLU with a global quadratic and approximates Softmax using normalized squared scores, avoiding comparison-based interval selection in these approximations. However, its fixed GeLU coefficients are neither tailored to layer-specific activation distributions nor updated during distillation. We instead initialize a separate quadratic for each layer from calibration data and jointly optimize its coefficients with the backbone weights. Proposition 1 clarifies this distinction: explicit coefficients change the training parameterization in biased standard FFNs and permit adaptation of a channel-wise invariant ratio in bias-free SwiGLU. To further compress latency, *BOLT* (Pang et al., 2024) leverages optimized fitting to utilize medium-degree polynomials, yet it remains sensitive to the degree-latency trade-off. Conversely, Zimerman et al. (2024) prioritize accuracy via high-degree polynomials (up to degree-27), trading additional polynomial-evaluation cost for approximation fidelity.

Recently, *AutoReP* (Peng et al., 2023a) introduced the DaPa (Distribution-aware Polynomial Approximation) method to fit coefficients based on Gaussian priors. In our evaluated Transformer models, we observe substantial deviations from Gaussian activation distributions. Under our initialization protocol, Gaussian-based fitting yields lower activation variance than the original network, suggesting a mismatch between the assumed distribution and the observed activations.

BumbleBee (Lu et al., 2025) instead optimizes secure protocols for Transformer operators. Such backend improvements are complementary to model adaptation: our BumbleBee experiment evaluates the benefit of replacing GeLU within its inference pipeline. CoQuad focuses on recovering model accuracy with inexpensive quadratic activations rather than introducing a new secure protocol.

## 9 Conclusion

CoQuad adapts Transformers to inexpensive quadratic activations through distribution-aware initialization and joint coefficient–backbone training. Our analysis distinguishes absorbable coefficients in biased standard FFNs from a channel-wise invariant ratio in bias-free SwiGLU. This motivates learning activation shapes alongside network weights rather than prescribing fixed coefficients throughout adaptation. DirectFit supplies layer-specific starting points, and Co-Optimize recovers whole-network behavior. The experiments demonstrate useful accuracy recovery across these architectural settings and secure-inference savings while retaining degree-two activation evaluation.

**Reproducibility statement** All models are initialized from public pretrained checkpoints (ViT-B/16, BERT-Base, Llama-2-7B, Llama-3-8B). Evaluation covers ImageNet-1K, CIFAR-10, RTE, and ARC-Easy. Full experimental settings, network configurations, and ablation protocols are described in section 6 and the appendix. Our anonymized code is available at `https://anonymous.4open.science/r/PolyTransformer-Submission-6C9F`.

**AI use statement** In this work, we used generative AI tools to assist in implementing the experimental code framework and for retrieval and discovery of related work. We have not used generative AI tools for generating synthetic datasets, for proving mathematical claims, or for drafting sections of the paper. Additionally, we used generative AI tools to edit parts of the paper to improve readability. All AI-assisted code was reviewed and tested for correctness by the authors, and all AI-assisted text was reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

## References

Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, et al. Openfhe: Open-source fully homomorphic encryption library. In *proceedings of the 10th workshop on encrypted computing & applied homomorphic cryptography*, pp. 53–63, 2022.

Alon Brutzkus, Ran Gilad-Bachrach, and Oren Elisha. Low latency privacy preserving inference. In *International Conference on Machine Learning*, pp. 812–821. PMLR, 2019.

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.

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

Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Theory of Cryptography Conference*, pp. 285–304. Springer, 2006.

Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pp. 643–662. Springer, 2012.

Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY: A framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

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.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pp. 201–210. PMLR, 2016.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. Sigma: Secure gpt inference with function secret sharing. *Cryptology ePrint Archive*, 2023.

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

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

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

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

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

Andes YL Kei and Sherman SM Chow. Shaft: Secure, handy, accurate, and fast transformer inference. In *Network and Distributed System Security Symposium, NDSS*, volume 2025, 2025.

Dongwoo Kim and Cyril Guyot. Optimized privacy-preserving cnn inference with fully homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 18:2175–2187, 2023.

Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34:4961–4973, 2021.

Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. {SWIFT}: Super-fast and robust {Privacy-Preserving} machine learning. In *30th USENIX security symposium (USENIX security 21)*, pp. 2651–2668, 2021.

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

Seewoo Lee, Garam Lee, Jung Woo Kim, Junbum Shin, and Mun-Kyu Lee. Hetal: Efficient privacy-preserving transfer learning with homomorphic encryption. In *International conference on machine learning*, pp. 19010–19035. PMLR, 2023.

Dacheng Li, Rulin Shao, Hongyi Wang, Han Guo, Eric P Xing, and Hao Zhang. MPCFormer: Fast, performant and private transformer inference with MPC. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=CWmvjOEhgH-`.

Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 619–631, 2017.

Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and Wenguang Chen. Bumblebee: Secure two-party inference framework for large transformers. In *32nd Annual Network and Distributed System Security Symposium, NDSS*, 2025.

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

Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 35–52, 2018.

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.

Hongwu Peng, Shaoyi Huang, Tong Zhou, Yukui Luo, Chenghong Wang, Zigeng Wang, Jiahui Zhao, Xi Xie, Ang Li, Tony Geng, et al. Autorep: Automatic relu replacement for fast private network inference. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5178–5188, 2023a.

Hongwu Peng, Shanglin Zhou, Yukui Luo, Nuo Xu, Shijin Duan, Ran Ran, Jiahui Zhao, Chenghong Wang, Tong Geng, Wujie Wen, et al. Pasnet: polynomial architecture search framework for two-party computation-based secure neural network deployment. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2023b.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

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

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Jianming Tong, Jingtian Dang, Anupam Golder, Arijit Raychowdhury, Cong Hao, and Tushar Krishna. Accurate low-degree polynomial approximation of non-polynomial operators for fast private inference in homomorphic encryption. *Proceedings of Machine Learning and Systems*, 6: 210–223, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *arXiv preprint arXiv:2004.02229*, 2020.

Andrew C Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pp. 160–164. IEEE, 1982.

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*, 2024.

# A Supplementary Illustrations

## A.1 Transformer Architecture

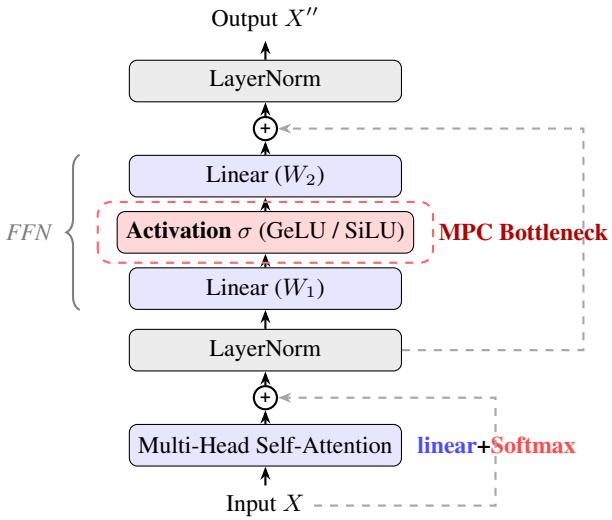



Figure 3: Transformer encoder structure. Blue box: linear operations (MPC-friendly); Red box: non-linear operations (MPC-expensive). Dashed lines: residual connections. Dashed red box: activation (the target of our replacement).

# B Supplementary Experiments

## B.1 Static Replacement Sweep

Table 4: Ablation on distillation: Performance collapse under static replacement.

| **Poly Layers** | 0 (Baseline) | 3 | 6 | 9 | 12 |
|---|---|---|---|---|---|
| *Top-1 Acc (%)* | 81.42 | 0.27 | 0.11 | 0.10 | *0.08* |

## B.2 Polynomial Coefficient Trainability

The table below summarizes the fixed-coefficient control and Co-Optimize results discussed in section 6.2. Both use DirectFit initialization, the same logit-MSE objective, and matched 10-epoch optimization settings. Backbone weight decay is 0.05; trainable polynomial coefficients are exempt from weight decay.

Table 5: *Ablation on the impact of polynomial coefficient trainability (ImageNet-1K).* Both variants update backbone weights under identical optimization protocols; only the trainability of polynomial coefficients differs.

| **Metric** | **Fixed Coeffs** | **Co-Optimize** |
|---|---|---|
| Init. Strategy | DirectFit | DirectFit |
| Backbone Train | ✓ | ✓ |
| Coeff. Train | × | ✓ |
| Top-1 Acc (%) | 76.3 | *79.83* |
| Gain (percentage points) | – | *+3.53* |

## B.3 DirectFit Initialization on BERT

Table 6 reports the BERT-Base/RTE initialization ablation. Both polynomial variants use frozen GeLU coefficients and retain Exact Softmax; the second replaces MPCFormer's coefficient initialization with DirectFit.

Table 6: BERT-Base accuracy on RTE. Polynomial coefficients remain frozen during training in both MPCFormer variants.

| Method | Accuracy (%) |
|---|---|
| Original BERT | 70.76 |
| MPCFormer | 71.84 |
| MPCFormer + DirectFit | 73.65 |

## B.4 Additional MPCFormer Results on Llama-2

Both MPCFormer and CoQuad replace SiLU at zero-based layer indices $1, 3, \ldots, 31$ in the 16-layer setting, preserving the other activations and the SwiGLU multiplicative branch, and are trained for the same number of steps. Training combines output-distribution distillation, next-token cross-entropy, and hidden-state MSE, with an additional cosine-distance term for Llama-3. MPCFormer keeps its prescribed polynomial $0.125x^2 + 0.25x + 0.5$ fixed throughout training, whereas CoQuad learns the coefficients jointly with the backbone. The unreplaced Llama-2 model scores 75.6% ARC-Easy accuracy. The 16-layer, 1,500-step MPCFormer run in Table 3 obtains 36.5% accuracy and 34.2% length-normalized accuracy. The 8-layer MPCFormer run obtains 51.9% accuracy and 46.7% normalized accuracy at 1,000 steps, and 53.7% and 47.5%, respectively, at 1,500 steps.

As an additional architecture evaluation, the trainable-quadratic Llama-3 model (Grattafiori et al., 2024) replaces 16 of 32 layers and reaches 73.0% ARC-Easy accuracy against its 80.8% baseline.

### B.5 LLAMA2 ACTIVATION-COST ESTIMATES

We further evaluate the scalability of CoQuad SiLU in a 2PC secure-inference pipeline based on additive secret sharing. Due to GPU memory constraints, we cannot execute the full 32-layer Llama2 secure-inference configuration in a single run. We therefore report a composed 32-layer latency estimate by combining the measured latency of 16 SiLU-replaced layers with the latency of the remaining 16 layers evaluated using the Fourier approximation baseline. This 50% replacement ratio is also accuracy-driven: replacing all SiLU layers causes substantial accuracy degradation on Llama2, so we conservatively evaluate CoQuad under partial replacement.

We evaluate three network settings: 10 Gbps / 0.5 ms RTT, 1 Gbps / 0.5 ms RTT, and 400 Mbps / 4 ms RTT. The 1 Gbps / 0.5 ms RTT setting matches BumbleBee's LAN configuration, enabling a contextual comparison under the same network setup and comparable hardware conditions. However, the comparison is not a strict apples-to-apples reproduction because BumbleBee uses an HE-based implementation, whereas CoQuad is evaluated in an additive-secret-sharing-based 2PC pipeline. In addition, BumbleBee's public release does not provide a ready-to-run Llama2 pipeline. Therefore, we use BumbleBee's paper-reported Llama-scale result as a reference point to contextualize large-model practicality.

Table 7 reports the composed 32-layer estimate mainly to characterize the activation-level cost of CoQuad SiLU under different network conditions. The replaced SiLU latency is 2.136 s under 10 Gbps / 0.5 ms RTT, 12.056 s under 1 Gbps / 0.5 ms RTT, and 29.485 s under 400 Mbps / 4 ms RTT. Across all network settings, the replaced SiLU operators contribute only 1.260 GB of communication. These results show that CoQuad can be integrated into an existing 2PC system and effectively reduces nonlinear activation overhead.

As shown in Table 8, under the matched LAN setting of 1 Gbps bandwidth and 0.5 ms RTT, BumbleBee reports 17.99 s and 1.92 GB communication for the SiLU activation component in its paper-reported Llama-scale experiment. This corresponds to a $1.49\times$ SiLU speedup and a $1.52\times$ communication reduction, even under our conservative partial-replacement regime.

Table 7: Extrapolated 32-layer results with 50% SiLU replaced by CoQuad (Llama2).

| **Network** | **SiLU Time (s)** | **SiLU Comm. (GB)** |
|---|---|---|
| 10Gbps, 0.5ms | 2.136 | 1.260 |
| 1Gbps, 0.5ms | 12.056 | 1.260 |
| 400Mbps, 4ms | 29.485 | 1.260 |

Table 8: Activation-level comparison under the LAN setting.

| **Metric** | **BumbleBee (Lu et al., 2025)** | **CoQuad** | **Gain** |
|---|---|---|---|
| SiLU Time (s) | 17.990 | 12.056 | $1.49\times$ |
| SiLU Comm. (GB) | 1.920 | 1.260 | $1.52\times$ |

*Note.* Both results are under LAN setting of 1Gbps bandwidth and 0.5ms RTT. BumbleBee values are directly taken from the paper-reported HE-based Llama implementation, while CoQuad reports a composed Llama2 result under 50% SiLU replacement.

### B.6 COMPATIBILITY WITH HE-BASED SYSTEM

We evaluate CoQuad in BumbleBee by replacing its original `seg3_gelu` implementation. The goal of this experiment is not to outperform highly optimized ASS-MPC systems such as SHAFT. Instead, we use BumbleBee to test whether CoQuad can be integrated into an existing HE-based secure inference pipeline and substantially reduce the overhead of nonlinear activation within that system. We run BumbleBee's ViT-B16 inference pipeline on a single $224 \times 224$ RGB input image under four network settings: 25 Gbps / 0.5 ms RTT, 10 Gbps / 0.5 ms RTT, 1 Gbps / 0.5 ms RTT, and 400 Mbps / 4 ms RTT. Since ViT-B16 contains 12 Transformer blocks, each inference invokes GeLU 12 times.

Table 9: Runtime comparison between BumbleBee's original GeLU and CoQuad GeLU (ViT-B16).

| Metric | 25 Gbps 0.5 ms | 10 Gbps 0.5 ms | 1 Gbps 0.5 ms | 400 Mbps 4 ms |
|---|---|---|---|---|
| CoQuad E2E (s) | 185.982 | 187.140 | 258.613 | 400.495 |
| Original E2E (s) | 201.885 | 204.627 | 299.660 | 487.973 |
| E2E Speedup | 1.09× | 1.09× | 1.16× | 1.22× |
| CoQuad GeLU (s) | 1.802 | 2.054 | 7.703 | 18.051 |
| Original GeLU (s) | 19.421 | 19.978 | 50.158 | 104.724 |
| GeLU Speedup | 10.78× | 9.73× | 6.51× | 5.80× |

Table 10: Communication comparison between BumbleBee's original GeLU and CoQuad GeLU (ViT-B16).

| Metric | Original | CoQuad | Reduction |
|---|---|---|---|
| GeLU comm. (GB) | 4.47025 | 0.80355 | 5.56× |
| Total comm. (GB) | 12.68311 | 9.01693 | 1.41× |

Tables 9 and 10 show that CoQuad consistently reduces both GeLU latency and communication. Across all network settings, GeLU communication drops from 4.47025 GB to 0.80355 GB, corresponding to a stable $5.56\times$ reduction. This communication saving translates into substantial GeLU runtime speedups, ranging from $10.78\times$ in the default local setting to $5.80\times$ under 400 Mbps / 4 ms RTT. At the end-to-end level, CoQuad achieves up to $1.22\times$ speedup. These results show that CoQuad GeLU can run stably inside BumbleBee and substantially reduce the cost of nonlinear activation. More broadly, this experiment suggests that our design is not only effective in BumbleBee's secure-inference pipeline, but also well aligned with HE-based systems, where additions and multiplications are the preferred computation primitives.

### B.7 Generality on Downstream Tasks

Beyond the native ImageNet domain, we further evaluate the transfer-learning capability of CoQuad on CIFAR-10. To ensure a strictly fair comparison, both the *FP32 baseline* and the CoQuad model are fine-tuned under an identical transfer protocol. Concretely, we retain the pre-trained 1000-class ImageNet classification head and map the 10 CIFAR-10 target categories to the first 10 logit indices of the output layer. Importantly, the semantic classes associated with these first 10 indices do not correspond to CIFAR-10 categories in the original ImageNet head. This index-level mismatch requires the model to re-adapt the mapping between deep representations and output nodes during fine-tuning.

Table 11: Classification Accuracy on CIFAR-10 under polynomial replacement (ViT-Base).

| Metric | Baseline | Hybrid-6 | Full-12 |
|---|---|---|---|
| Poly. Layers | 0/12 | 6/12 | 12/12 |
| Degree | – | 2 | 2 |
| Accuracy (%) | 98.53 | **98.80** | 97.53 |
| Gap (%) | – | **+0.27** | -1.00 |
| Activation | GeLU | Mixed | Polynomial |

All models, including the baseline and polynomial variants, are trained with full-parameter fine-tuning throughout. Input images are resized to $224 \times 224$ via bilinear interpolation. We follow a standard supervised fine-tuning setup in our experiments: models are trained for *10* epochs with batch size *32* and a constant learning rate of $5 \times 10^{-5}$. To prevent overfitting, we apply standard data augmentation, including *RandAugment*, random horizontal flipping, and random cropping with padding. Evaluation is conducted on the standard CIFAR-10 test set without test-time augmentation.

The results demonstrate strong downstream adaptability of polynomial networks in practice. As shown in Table 11, under this transfer setting, the *Hybrid-6* configuration achieves 98.80% accuracy, comparable to the fine-tuned FP32 baseline (98.53%). This indicates that CoQuad remains stable even under index-mismatched transfer settings. Moreover, the fully polynomial *Full-12* configuration reaches 97.53% accuracy, confirming that even without standard transcendental activations, a purely arithmetic network can successfully adapt to downstream tasks through full-parameter fine-tuning while maintaining competitive performance.

# C Implementation and Evaluation Details

## C.1 Secure-Inference Hardware

Secure-inference experiments were conducted on a Linux server with two AMD EPYC 9124 (Zen4) processors and two NVIDIA A6000 GPUs. The machine provides 64 vCPUs ($16 \times 2$ physical cores with SMT) with a maximum turbo frequency of 3.6 GHz and 256 GB of RAM. The processor supports AVX-512 instructions, which we leverage to accelerate our polynomial and matrix kernels.

## C.2 ImageNet ViT Architecture, Training, and Evaluation

For completeness, the ViT logit-MSE objective in section 4 is written elementwise below. Here $B$ is the batch size, $K$ is the number of classes, and the teacher is frozen.

$$\mathcal{L}_{\mathrm{ViT}} = \frac{1}{BK} \sum_{i=1}^{B} \sum_{k=1}^{K} \left( z_{ik}^{S}(\theta, \phi) - \mathrm{stopgrad}(z_{ik}^{T}) \right)^2 , \tag{7}$$

**ViT Architecture** The ViT-Base backbone consists of 12 Transformer encoder layers with hidden dimension 768, 12 attention heads, and an MLP dimension of 3072. The model operates on $224 \times 224$ inputs with a patch size of $16 \times 16$, totaling approximately 86M parameters.

**Training Implementation** In our implementation, we first execute the operator-replacement procedure to explicitly construct `PolyGeLU` modules within the computational graph. We then proceed to the Co-Optimize fine-tuning stage, using the AdamW optimizer with a learning rate of $6 \times 10^{-5}$, together with a one-epoch linear warmup followed by cosine decay. We adopt a parameter-grouped optimization scheme: weight decay is selectively removed for polynomial coefficients, while backbone parameters retain a regularization strength of $0.05$. To ensure high numerical throughput and align with modern production-level compute settings, ViT training is conducted on NVIDIA H200 using BFloat16 mixed precision and graph compilation via `torch.compile`. Standard preprocessing is applied, resizing images to $256$ pixels followed by a $224 \times 224$ center crop, and no test-time augmentation (TTA) is used.

**Evaluation and Checkpoint Loading** We evaluate on the *ILSVRC-2012* validation set. Hybrid-6 retains its original evaluation result, as indicated in Table 2. Image preprocessing follows the conventional pipeline: inputs are resized to *256* pixels and center-cropped to $224 \times 224$, with no test-time augmentation (TTA). During inference, to ensure that model behavior strictly follows polynomial arithmetic semantics, we adopt a *structure-first, parameter-second* loading procedure. Specifically, prior to loading checkpoints, we instantiate the *ViT* backbone and execute the operator-replacement strategy to explicitly construct *PolyGeLU* modules in the computation graph. This loading order guarantees that all subsequent numerical evaluations are performed on the structurally transformed graph, ensuring that inference behavior is fully consistent with the intended polynomial operators.

# D Supplementary Discussion

## D.1 Offline Adaptation Cost and Early-Epoch Accuracy

On our setup, the full 10-epoch ViT Co-Optimize run takes approximately two hours. The recorded Top-1 accuracy reaches 51.81% after the first epoch (approximately 12 minutes) and 73.02% after the second. These observations characterize the offline adaptation cost of this run; they do not establish a

convergence-rate advantage over another training recipe. The resulting adapted model can then be reused for online inference.

### D.2 Higher-Degree Polynomial Replacements: Future Directions

To explore greater replacement coverage in Llama, future work will explore higher-degree polynomial approximations for large language models. We hypothesize that the complex gating behavior in SiLU may be more effectively captured by higher-order functions than by simple quadratics. From a protocol perspective, increasing polynomial degree introduces a subtle trade-off: while it increases the number of multiplications (and thus Beaver triple bandwidth consumption), it can substantially reduce communication rounds compared to comparison-based functions. For example, under standard arithmetic secret sharing, a fourth-degree polynomial can be evaluated in as few as two communication rounds. If higher-degree polynomials provide sufficient expressiveness to enable full replacement in LLMs, the dramatic reduction in communication rounds under wide-area network settings may yield a net latency benefit that offsets the additional bandwidth cost introduced by extra multiplications. More broadly, this points to a context-dependent design space in which polynomial degree, replacement schedule, model family, task objective, and MPC protocol characteristics must be jointly optimized.

## E Extended Method Discussion

### E.1 Model Capacity and Distillation Motivation

We use the pretrained teacher to supervise adaptation after quadratic replacement. The student inherits the backbone weights and updates them jointly with the activation coefficients to recover teacher predictions. The accuracy recovery in our ViT experiments motivates this choice; it does not establish that parameter redundancy guarantees recovery.

### E.2 Optimization Stability Rationale

In the ViT implementation, gradient clipping with max_norm=1.0 limits the gradient norm before each optimizer update. AdamW applies weight decay of 0.05 to backbone parameters and no weight decay to polynomial coefficients. This leaves the shared activation coefficients to adapt under the logit-MSE distillation objective without direct shrinkage from weight decay. These are implementation choices, not a claim that the coefficients enlarge the FFN function class or that their gradients concentrate in a particular activation region. The present ablation evaluates the resulting training parameterization; separating the effects of regularization and parameterization requires additional controls.

### E.3 Discussion of Full Parameter Updates

We use full-parameter fine-tuning so that all backbone weights can adapt to the changed activations. We do not evaluate parameter-efficient alternatives such as LoRA (Hu et al., 2022), and do not claim that low-rank adaptation is insufficient for this task.

### E.4 Motivation for Output-Amplitude Alignment

After weighted fitting, we rescale all polynomial coefficients by the ratio of target to predicted output standard deviation, with $10^{-6}$ added to the denominator. This calibration step approximately aligns output variation on the sampled inputs; it does not preserve the weighted least-squares optimum or establish a mechanism for accuracy recovery. Algorithm 1 specifies the procedure.

### E.5 Progressive Layer-wise DirectFit Procedure

Algorithm 1 details the initialization procedure described in section 5. Weights multiply residuals before squaring, as in Equation 6.

**Algorithm 1** Progressive Layer-wise DirectFit

**Require:** Pretrained model $M$, calibration data $B$, selected layers $\mathcal{R}$, sample budget $N$, residual weight $\lambda$, degree $d = 2$
**Ensure:** Model $M$ with initialized quadratic activations at layers $\mathcal{R}$
1: Set $M$ to evaluation mode
2: **for** each selected layer $l \in \mathcal{R}$ in network order **do**
3: Run forward pass with $B$ and capture inputs $X_l$ to current activation $\sigma_l$
4: $\hat{X} \leftarrow$ Flatten valid inputs (excluding padding) and sample up to $N$ points
5: $Y_{target} \leftarrow \sigma_l(\hat{X})$ {Original activation on captured inputs}
6: Assign weights $W$: $w_i = \lambda$ if $x_i > 0$, else $w_i = 1$
7: $c_l \leftarrow \text{WeightedLeastSquares}(\hat{X}, Y_{target}, W, \text{degree} = d)$
8: $Y_{pred} \leftarrow P(\hat{X}; c_l)$ {Initial polynomial response}
9: $s \leftarrow \text{std}(Y_{target})/(\text{std}(Y_{pred}) + 10^{-6})$ {Calculate standard-deviation-based scale factor}
10: $c_l \leftarrow c_l \cdot s$ {Global amplitude rescaling}
11: Replace $\sigma_l$ with polynomial $P(x; c_l)$ in $M$
12: Update computation graph for subsequent layers
13: **end for**

## F Additional Background

### F.1 Secure-Inference Protocol Context

Privacy-preserving computing uses Secure MPC and HE to enable inference over secret-shared or encrypted data among mutually distrustful parties. A central challenge is the cost gap between arithmetic and nonlinear operations. In MPC, additions are local, and secure multiplications can be implemented by Beaver triples (Damgård et al., 2012). In contrast, nonlinear operators such as ReLU, GeLU, and Softmax typically require more expensive protocols, including garbled circuits (Yao, 1982), bit-decomposition-based comparisons (Damgård et al., 2006; Demmler et al., 2015), or approximation-based methods. Prior secure-inference systems address this overhead through hybrid protocols (Liu et al., 2017; Juvekar et al., 2018; Mishra et al., 2020; Mohassel & Rindal, 2018; Demmler et al., 2015) and optimized MPC frameworks (Rathee et al., 2020; Huang et al., 2022; Knott et al., 2021; Hao et al., 2022; Dong et al., 2025; Gupta et al., 2023), but nonlinear evaluation remains a major source of latency and communication. In HE, especially CKKS (Cheon et al., 2017), computation is naturally expressed through additions and multiplications, making polynomial functions efficient while non-polynomial functions require approximations that increase multiplicative depth and may require bootstrapping (Chillotti et al., 2020; Al Badawi et al., 2022; Brutzkus et al., 2019; Lee et al., 2023; 2022; Kim & Guyot, 2023). Thus, our work focuses on the FFN-side activation component of this bottleneck, namely GeLU and SiLU in Transformer architectures.

Our evaluation assumes semi-honest security. Malicious-security mechanisms (Koti et al., 2021; Wagh et al., 2020) require additional protocol choices and are outside the scope of this evaluation.

### F.2 Transformer FFNs and Activation Definitions

As illustrated in Figure 3, each Transformer block (Vaswani et al., 2017) comprises Multi-Head Self-Attention (MHSA) and a Feed-Forward Network (FFN), connected by residual links and layer normalization. The standard FFN is a position-wise two-layer MLP:

$$\text{FFN}(X) = \sigma(XW_1 + b_1)W_2 + b_2, \tag{8}$$

where $\sigma$ is the activation function. GeLU (Hendrycks & Gimpel, 2016), commonly used in BERT (Devlin et al., 2019) and ViT (Dosovitskiy et al., 2020), is defined as $\text{GeLU}(x) = x\Phi(x)$, where $\Phi$ is the standard normal cumulative distribution function. SiLU (Ramachandran et al., 2017) is defined as

SiLU$(x) = x/(1 + e^{-x})$; Llama (Touvron et al., 2023) uses this activation within SwiGLU (Shazeer, 2020), as described in section 6.2.

### F.3 Quadratic Reparameterization in ViT

Proposition 1(i) gives the general absorption construction for two nondegenerate quadratics in a biased standard FFN. Choosing the reference activation $q(u) = u^2$ recovers the square-activation specialization for ViT. Both affine biases are required by this construction; bias shifts are broadcast over tokens and examples. The equivalence concerns the function class, not optimizer trajectories or regularization. For bias-free gated channels, Proposition 1(ii) instead characterizes equivalence through the coefficient ratio. The experimental interpretations in section 6.2 follow this distinction.