RATIONALLY LINEARIZED MLPS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present *Rationally Linearized Multilayer Perceptrons* (**RaLiMPs**), a new approach to efficiently linearizing MLPs using low-degree rational approximations of standard activation functions. Inspired by polynomial-based MLP linearization, we show that rational approximations can provide accurate representations at substantially lower degree. RaLiMPs exploit these approximations to construct kernel representations with narrow bottlenecks, enabling efficient composition across MLP layers. We provide theoretical guarantees for the resulting approximation and evaluate RaLiMPs across applications ranging from implicit 3D models to large language models. Our experiments demonstrate favorable accuracy–efficiency tradeoffs, with RaLiMPs achieving competitive or improved accuracy while substantially reducing inference cost.

1 INTRODUCTION AND RELATED WORK

Nonlinear processing modules are at the heart of modern deep learning and are critical to modeling complex phenomena with machine learning (ML) approaches (Goodfellow et al., 2016; Hinton et al., 2006). Virtually all modern ML systems are built around one or both of the fundamental building blocks: *multi-layer perceptrons* (MLPs) and the *attention layers*. Both are highly nonlinear functions of their inputs, and their combination yields high-capacity models that can approximate complex functions while remaining amenable to robust training.

However that expressiveness comes at the cost: a non-negligible computational footprint which is quadratic in the sequence of the input length for the attention layers and quadratic in the dimensionality of the input for the MLPs. That poses practical challenges and led to fruitful research on efficient attention layers in Transformers(Tay et al., 2023; Elouargui et al., 2023; Choromanski et al., 2021; Kitaev et al., 2020). With recent improvements to efficient attention mechanism, reducing the computational cost of MLPs has become increasingly important. This is in part because the highly optimized attention computation often makes MLPs the new computational bottlenecks for Transformers, in particular when $d^2 \gg L$ for the input dimensionality d and sequence length L. Furthermore, MLPs are ubiquitous in ML and are present also in modern ML architectures that are not attention-based, e.g. implicit 3D-models such as *Neural Radiance Fields* (NeRFs) (Mildenhall et al., 2022), Signed Distance Fields (SDFs) (Ortiz et al., 2022) or in Physics Informed Neural Networks (PINNs) (Raissi et al., 2019).

A single layer of the MLP processing unit for the input vector $\mathbf{x} \in \mathbb{R}^d$, can be seen as a map:

$$\mathbf{x} \rightarrow f(\mathbf{W}\mathbf{x}) \in \mathbb{R}^{d_{\text{out}}}, \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{l \times d_{\text{out}}}$ is a learned matrix of weights and $f : \mathbb{R} \rightarrow \mathbb{R}$ is a fixed non-linear *activation function* applied element-wise. The mapping defined in Eq. 1 is a non-linear function of $\mathbf{x}$ but it can be approximated by a product of two feature maps, $f(\mathbf{W}\mathbf{x}) \approx \Psi(\mathbf{W})\Phi(\mathbf{x})$ where $\Phi(\mathbf{x})$ depends only on the input and $\Psi(\mathbf{W})$ depends only on the weights. More specifically, we consider layers of the form

$$\mathbf{x} \rightarrow \Psi(\mathbf{W})\Phi(\mathbf{x}), \tag{2}$$

where $\Phi, \Psi : \mathbb{R}^d \rightarrow \mathbb{R}^F$ are nonlinear, potentially randomized feature maps, with Ψ applied row-wise to $\mathbf{W}$. We will refer to the map defined above as the (Ψ, Φ)-linearized PL (perceptron layer). It was shown in (Sehanobish et al., 2024) that for large classes of the activation functions f, randomized maps $\Phi = \Phi_f, \Psi = \Psi_f$ can be constructed with the use of *Random Fourier Features* (RFFs) (Rahimi & Recht, 2007; 2008) in such a way that the mechanism proposed in Eq. 2 provides an

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.

unbiased estimation of the map from Eq. 1:

$$f(\mathbf{W}\mathbf{x}) = \mathbb{E}[\Psi_f(\mathbf{W})\Phi_f(\mathbf{x})], \tag{3}$$

with the expectation taken with respect to the randomness in Ψ_f, Φ_f (often one can take: $\Psi_f = \Phi_f$). Note that Eq. 2 is linear in $\Phi(\mathbf{x})$ and that fact itself directly leads to straightforward computational improvements (Sehanobish et al., 2024). The key observation is that even though the map $\mathbf{x} \rightarrow \Psi(\mathbf{W})\Phi(\mathbf{x})$ is still non-linear in $\mathbf{x}$, as a linear transformation of $\Phi(\mathbf{x})$, it **disentangles** the input $\mathbf{x}$ from the weight matrix $\mathbf{W}$, only to entangle them at the end of the entire computational process via simple dot-product calculations.

However achieving an approximation that is not only unbiased, but is also highly accurate, i.e. characterized by low variance, is in general a difficult problem. This challenge directly translates to the accuracy gap that is observed between multi-layer modules using (Ψ, Φ)-linearized PLs and regular MLPs. For particular classes of activation functions, such as polynomials, random features based on tensor products can replace RFFs to provide a simple linearized mechanism with low variance (Kim et al., 2025).

Several other approaches were proposed to improve the computational efficiency of MLPs, yet most of them departure from the low-variance estimation of the regular MLPs. Many of these approaches focus on replacing $\mathbf{W}$ with its structured counterpart (Dao et al., 2019; Sindhwani et al., 2015; Li et al., 2025b), supporting sub-quadratic matrix-vector multiplication. For instance, Monarch-matrices (Dao et al., 2022) are defined as products of block-diagonal and permutation matrices. Their successors, the so-called *Block Tensor-Train* (BTT) matrices (Qiu et al., 2024) (containing Monarch-matrices as special cases), derive from the rich theory on Tensorized Neural Networks (Novikov et al., 2015), leveraging sums over Kronecker products.

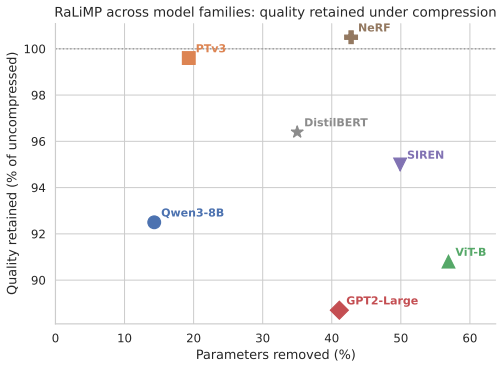

Figure 1: **Compression–quality trade-off across diverse model families.** Each point represents a model family compressed using RaLiMPs.

In this paper, we revisit the (Ψ, Φ)-linearized MLP framework in the context of recent work suggesting that rational functions are particularly effective choices for MLP activations (Tang & Townsend, 2026). We show that (Ψ, Φ)-linearized MLPs can achieve high accuracy while maintaining an attractive computational footprint when approximating low-degree rational activation functions. To the best of our knowledge, this combination has not been explored previously. Low-degree rational functions are also substantially easier to approximate accurately than the higher-degree functions considered in much of the prior work, making them well suited to random feature methods. Our contributions are as follows :

- We present *Rationally Linearized Multilayer Perceptrons* (RaLiMP), a novel (Ψ, Φ) linearized layer and prove results on unbiasedness, variance and approximation errors (Sec. 2).
- We comprehensively test RaLiMP across various tasks like 3D modeling, vision and language modeling against other linearized MLPs and the baselines (Sec. 3).

2 Rationally Linearized Multilayer Perceptrons (RaLiMPs)

2.1 Preliminaries

Let $\boldsymbol{x} \in \mathbb{R}^d$ be an input and $\boldsymbol{W} \in \mathbb{R}^{l \times d}$ a (pretrained) weight matrix with rows $\boldsymbol{w}_i$, and let f be an activation. So the standard feed-forward layer computes $f(\boldsymbol{x} \cdot \boldsymbol{w}_i)$. We do not linearize f directly: we replace it by a rational surrogate $R \approx f$ and linearize R. In particular, we construct feature maps $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^F$ (on the input) and $\Psi : \mathbb{R}^d \rightarrow \mathbb{R}^F$ (on each weight row), where $\boldsymbol{a} \cdot \boldsymbol{b} \stackrel{\text{def}}{=} \boldsymbol{a}^\top \boldsymbol{b}$:

$$\langle \Phi(\boldsymbol{x}),\, \Psi(\boldsymbol{w}) \rangle \;\approx\; R(\boldsymbol{x} \cdot \boldsymbol{w}), \qquad R(t) = P(t) + \sum_j \frac{A_j + B_j(t - p_j)}{(t - p_j)^2 + q_j^2}, \qquad t = \boldsymbol{x} \cdot \boldsymbol{w} \tag{4}$$

with a partial-fraction decomposition into a polynomial part $P(t) = \sum_{n=0}^{K} c_n t^n$ and irreducible-quadratic terms (denominators with no real zero). This decomposition is the governing principle of the construction. We build one separable feature for each part:

- the **polynomial part** P is handled by product features that are separable and unbiased for each monomial $(\boldsymbol{x} \cdot \boldsymbol{w})^n$ (Sec. 2.2);
- each **irreducible-quadratic atom** is handled by writing it as a Laplace integral over frequencies s of $\cos(st)$, and estimating each $\cos(st) = \mathrm{Re}\,(e^{ist})$ with separable Gaussian features (Secs. 2.3 and 2.4).

A concatenation of separable blocks is separable, so their sum is a single map of the form Eq. 4. We first present the centered atom $1/(1+t^2)$ (Sec. 2.3) and give the general shifted term in Sec. 2.4. Because Φ depends only on $\boldsymbol{x}$ and Ψ only on $\boldsymbol{W}$, the full output is a matrix-vector product $\boldsymbol{y} = \Psi(\boldsymbol{W})\Phi(\boldsymbol{x})$, where $\Psi(\boldsymbol{W}) \in \mathbb{R}^{l \times F}$ stacks $\Psi(\boldsymbol{w}_i)$ as rows.

2.2 Handling Polynomial Parts

Eq. 4 requires only that the polynomial part be realized by a separable block with finite variance. Any such mechanism may be used, and the guarantee of App. A.3 treats it as one additive term. A polynomial linearization is the natural choice. We use the EUGen mechanism (Kim et al., 2025): for degree n it draws shared Gaussian projections $\boldsymbol{G}^{(1)}, \ldots, \boldsymbol{G}^{(n)} \in \mathbb{R}^{m \times d}$ and, since $\mathbb{E}[\boldsymbol{G}_r^\top \boldsymbol{G}_r] = \boldsymbol{I}_d$, estimates $(\boldsymbol{x}\cdot\boldsymbol{w})^n$ unbiasedly by averaging $\prod_{i=1}^{n}(\boldsymbol{G}_r^{(i)}\cdot\boldsymbol{x})(\boldsymbol{G}_r^{(i)}\cdot\boldsymbol{w})$ over rows r. This is separable, with features $\phi_n(\boldsymbol{u})_r = \prod_{i=1}^{n}(\boldsymbol{G}_r^{(i)}\cdot\boldsymbol{u})$ for $n \geq 1$ and $\phi_0(\boldsymbol{u}) = 1$, where $\boldsymbol{u} \in \{\boldsymbol{x}, \boldsymbol{w}\}$; the block contributes $c_0 + \sum_{n=1}^{K} c_n \frac{1}{m}\langle \phi_n(\boldsymbol{x}),\, \phi_n(\boldsymbol{w})\rangle \approx P(t)$ and is of width $1 + Km$. Note that this polynomial part may also be approximated by any random feature mechanism with more favorable variance or cost guarantees.

2.3 Handling Irreducible-Quadratic Atoms: centered atom $\frac{1}{1+t^2}$

This atom is bounded and saturating, whereas any Taylor truncation diverges for large t. We obtain separable features for it in two steps: (1) a deterministic frequency quadrature, and then (2) separable softmax-kernel features per frequency, whose real and imaginary parts estimate $\cos(st)$ and $\sin(st)$.

Frequency quadrature. Using the Laplace identity,

$$\frac{1}{1+t^2} = \int_0^\infty e^{-s}\cos(s\,t)\,ds \;\approx\; \sum_{k=1}^{n_{\mathrm{freq}}} \gamma_k\,\cos(s_k\,t), \tag{5}$$

where $\{s_k, \gamma_k\}$ are the Gauss–Laguerre (Abramowitz & Stegun, 1964) nodes and weights (γ_k carry the e^{-s} factor). Nodes above a cutoff $s_{\max}$ are dropped and n_{freq} counts those retained, so the sum discards a fraction $\sum_{s_k > s_{\max}} \gamma_k$ of the unit total weight $\int_0^\infty e^{-s} ds$ in exchange for bounding the feature growth of the next step. Since e^{-s} is a density this integral could instead be sampled; App. C.2 shows why deterministic nodes are preferable. It remains to write each $\cos(s_k\,t)$ separably.

2.3.1 The rational random features

We now unbiasedly estimate each factor $\cos(s_k t)$ (and $\sin(s_k t)$). For this we use the softmax- and Gaussian-kernel approximation mechanism of Performers (Choromanski et al., 2021) rather than regular random Fourier features (Rahimi & Recht, 2007), which linearize shift-invariant kernels and not functions of $\boldsymbol{x}\cdot\boldsymbol{w}$. Fix a node $s \equiv s_k$, draw m i.i.d. Gaussian projections $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$, and set the following real scalars for a vector $\boldsymbol{u}$ ($= \boldsymbol{x}$ or $\boldsymbol{w}$) and a scaling factor τ:

$$Z_r(\boldsymbol{u}) = \sqrt{s/2}\,\big(\boldsymbol{\omega}_r\cdot\boldsymbol{u}/\tau\big), \qquad h_r(\boldsymbol{u}) = \tfrac{s}{2}\|\boldsymbol{u}/\tau\|^2, \qquad r = 1, \ldots, m. \tag{6}$$

The rational random feature maps are the two length-m vectors

$$\boxed{\varphi^{\cos}(\boldsymbol{u})_r = \tfrac{1}{\sqrt{m}}\, e^{Z_r(\boldsymbol{u})}\cos\big(Z_r(\boldsymbol{u}) - h_r(\boldsymbol{u})\big), \qquad \varphi^{\sin}(\boldsymbol{u})_r = \tfrac{1}{\sqrt{m}}\, e^{Z_r(\boldsymbol{u})}\sin\big(Z_r(\boldsymbol{u}) - h_r(\boldsymbol{u})\big).} \tag{7}$$

These are the real and imaginary parts of the softmax-kernel feature $\varphi_\lambda(\boldsymbol{u})_r = \frac{1}{\sqrt{m}} \exp\big(\lambda\, \boldsymbol{\omega}_r \cdot \boldsymbol{u}/\tau - \frac{1}{2}\lambda^2 \|\boldsymbol{u}/\tau\|^2\big)$ with the complex scale $\lambda = \sqrt{s/2}\,(1+i)$ ($\lambda^2 = is$), which satisfies $\mathbb{E}[\varphi_\lambda(\boldsymbol{x})_r \varphi_\lambda(\boldsymbol{w})_r] = \frac{1}{m}\exp(\lambda^2\, \boldsymbol{x}\cdot\boldsymbol{w}/\tau^2) = e^{ist/\tau^2}$ (Choromanski et al., 2021). Since τ rescales both arguments it acts on the scalar variable as $t \mapsto t/\tau^2$. For simplicity, we take $\tau = 1$ below and calibrate it in App. C.1. This partial-fraction, separable-feature view subsumes prior polynomial and Fourier random-feature constructions of MLPs (Kim et al., 2025; Sehanobish et al., 2024) as instances of one kernel representation, which we make precise in App. B. Summing the real (resp. imaginary) part of the product over r gives the unbiased estimators of $\cos(s_k t)$ (resp. $\sin(s_k t)$) as $1/\sqrt{m}$ factor is already inside the formula in Eq. 7:

$$\widehat{C}_k(\boldsymbol{x}, \boldsymbol{w}) = \sum_r \big[\varphi^{\cos}(\boldsymbol{x})_r \varphi^{\cos}(\boldsymbol{w})_r - \varphi^{\sin}(\boldsymbol{x})_r \varphi^{\sin}(\boldsymbol{w})_r\big] \;\to\; \cos(s_k t), \tag{8}$$

$$\widehat{S}_k(\boldsymbol{x}, \boldsymbol{w}) = \sum_r \big[\varphi^{\cos}(\boldsymbol{x})_r \varphi^{\sin}(\boldsymbol{w})_r + \varphi^{\sin}(\boldsymbol{x})_r \varphi^{\cos}(\boldsymbol{w})_r\big] \;\to\; \sin(s_k t). \tag{9}$$

Lemma 1 below verifies these limits. Complete proof is in App. A.1.

Lemma 1 (Unbiasedness). *Let $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ and $t = \boldsymbol{w}\cdot\boldsymbol{x}$. Then for $\tau = 1$,*

$$\mathbb{E}\big[\widehat{C}_k\big] = \cos(s_k t), \qquad \mathbb{E}\big[\widehat{S}_k\big] = \sin(s_k t), \tag{10}$$

for general $\tau > 0$, applying this to $\boldsymbol{x}/\tau$ and $\boldsymbol{w}/\tau$ gives $\mathbb{E}[\widehat{C}_k] = \cos(s_k t/\tau^2)$, $\mathbb{E}[\widehat{S}_k] = \sin(s_k t/\tau^2)$.

Proof sketch. Let $\tau = 1$. By angle addition, each $\widehat{C}_k$ summand is $\frac{1}{m}\, e^{\xi}\cos(\xi - H)$ where $\xi := \sqrt{s/2}\,\boldsymbol{\omega}\cdot(\boldsymbol{x}+\boldsymbol{w})$ a centered Gaussian with variance $\frac{s}{2}\|\boldsymbol{x}+\boldsymbol{w}\|^2$ and $H := \frac{s}{2}(\|\boldsymbol{x}\|^2 + \|\boldsymbol{w}\|^2)$. Summing them, $\mathbb{E}[\widehat{C}_k] = \mathbb{E}\, e^{\xi}\cos(\xi - H)$; since $e^{\xi}\cos\xi = \mathrm{Re}\, e^{(1+i)\xi}$ and $(1+i)^2 = 2i$, the Gaussian moment generating function evaluates this to $\cos(st)$ ($\widehat{S}_k$ case identical with $\mathrm{Re} \to \mathrm{Im}$). □

Feature towers. We now present the approximation of the centered atom $\frac{1}{1+t^2}$ (Eq. 5). In that case, we only need $\widehat{C}_k$. To write it as an inner product, split its two terms into two sub-blocks and fold the weight γ_k and the minus sign onto the input side. Length-$2m$ towers, per node k are

$$\phi_k^{\mathrm{rat}}(\boldsymbol{x}) = \big[\ \gamma_k\, \varphi^{\cos}(\boldsymbol{x})\ ,\ -\gamma_k\, \varphi^{\sin}(\boldsymbol{x})\ \big], \qquad \psi_k^{\mathrm{rat}}(\boldsymbol{w}) = \big[\ \varphi^{\cos}(\boldsymbol{w})\ ,\ \varphi^{\sin}(\boldsymbol{w})\ \big]. \tag{11}$$

They satisfy $\phi_k^{\mathrm{rat}}(\boldsymbol{x}) \cdot \psi_k^{\mathrm{rat}}(\boldsymbol{w}) = \gamma_k \widehat{C}_k \to \gamma_k \cos(s_k t)$. Concatenating over the n_{freq} nodes gives $\phi^{\mathrm{rat}}(\boldsymbol{x}), \psi^{\mathrm{rat}}(\boldsymbol{w}) \in \mathbb{R}^{2n_{\mathrm{freq}}m}$ with $\phi^{\mathrm{rat}}(\boldsymbol{x}) \cdot \psi^{\mathrm{rat}}(\boldsymbol{w}) \approx \sum_k \gamma_k \cos(s_k t) \approx 1/(1+t^2)$. Both towers share the same functional form; only the $\boldsymbol{x}$-side carries the coefficients γ_k and signs, so $\psi^{\mathrm{rat}}(\boldsymbol{w})$ depends on $\boldsymbol{w}$ alone. The weights γ_k, initialized at the quadrature values of Eq. 5, and the projections $\boldsymbol{\omega}_r$ can be made trainable. The general shifted atom (Eq. 14) additionally uses $\widehat{S}_k$; its tower is constructed in Sec. 2.4.

Inference Speedups. Since each block separates the weights from the inputs, its computation can be fused into a subsequent linear layer. This eliminates intermediate matrices, reducing the deployed model size and accelerating inference (Sec. C.4).

2.3.2 Variance Analysis

The estimator presented in the previous section is unbiased, but its summand $e^{\xi}\cos(\xi - H)$ is unbounded. Set $\sigma^2 = \frac{s}{2}\|\boldsymbol{x}+\boldsymbol{w}\|^2/\tau^2$ and let $H = \frac{s}{2}(\|\boldsymbol{x}\|^2 + \|\boldsymbol{w}\|^2)$ as in sketch proof of lemma 1, let $Y_r^c = m\big[\varphi^{\cos}(\boldsymbol{x})_r \varphi^{\cos}(\boldsymbol{w})_r - \varphi^{\sin}(\boldsymbol{x})_r \varphi^{\sin}(\boldsymbol{w})_r\big]$ and Y_r^s the analogous $\widehat{S}_k$ term, so that $\widehat{C}_k = \frac{1}{m}\sum_r Y_r^c$ (and likewise for $\widehat{S}_k$). The second moments are exact. Detailed proof is in App. A.2.

Proposition 1 (Second moments). *For each projection, $\mathbb{E}[(Y^c)^2] = \frac{1}{2}\big(e^{2\sigma^2} + \cos(4\sigma^2 - 2H/\tau^2)\big)$ and $\mathbb{E}[(Y^s)^2] = \frac{1}{2}\big(e^{2\sigma^2} - \cos(4\sigma^2 - 2H/\tau^2)\big)$. Consequently, averaging over m i.i.d. projections,*

$$\mathrm{Var}\big(\widehat{C}_k\big) \;\le\; \frac{1}{2m}\Big(e^{2\sigma^2} + 1\Big), \tag{12}$$

$$\mathrm{Var}\big(\widehat{C}_k\big) + \mathrm{Var}\big(\widehat{S}_k\big) \;=\; \frac{1}{m}\Big(e^{2\sigma^2} - 1\Big). \tag{13}$$

The variance depends on the inputs only through $\|\boldsymbol{x} + \boldsymbol{w}\|$ and grows like $e^{2\sigma^2}$, a norm dependence that persists even though the target $\cos(s_k t)$ is bounded. The estimator is thus finite-variance whenever the norms are bounded (e.g. post-LayerNorm on a sphere). The cutoff $s_{\max}$ and the input scale τ are therefore the two quantities that control it. For a *sum* of atoms the variance is moreover governed by the narrowest pole width $q_{\min} = \min_j q_j$ (Cor. 1), not by $s_{\max}$ and τ alone. The scale τ is the practical handle, since the exponent enters as $2\sigma^2$; we calibrate it per layer in App. C.1.

2.4 Handling Irreducible-Quadratic Atoms: General Shifted Atom

The centered atom extends to the general partial-fraction term

$$g(t) = \frac{A + B(t-p)}{(t-p)^2 + q^2}, \qquad q > 0, \tag{14}$$

whose denominator has complex-conjugate roots $p \pm iq$. A rational activation with real coefficients that is finite on $\mathbb{R}$ has no real pole, so its denominator factors into such conjugate pairs and R decomposes as the polynomial P plus one atom per simple pair (repeated pairs in App. C.3). Both parts share the centered atom's Laplace representation, at frequency s/q with the shift p as a phase:

$$\frac{1}{(t-p)^2+q^2} = \frac{1}{q^2}\int_0^\infty e^{-s}\cos\!\big(\tfrac{s}{q}(t-p)\big)ds, \quad \frac{t-p}{(t-p)^2+q^2} = \frac{1}{q}\int_0^\infty e^{-s}\sin\!\big(\tfrac{s}{q}(t-p)\big)ds. \tag{15}$$

Discretizing with the same nodes at rescaled frequency s_k/q, and using $\cos(\frac{s_k}{q}(t-p)) = \cos(\frac{s_k}{q}p)\,\widehat{C}_k + \sin(\frac{s_k}{q}p)\,\widehat{S}_k$ (with analogous identity for $\sin$), both parts are separable via the same estimators (Eqs. 8–9), evaluated at $s_k \to s_k/q$. The shift p mixes the two estimators: the even part (A) and the odd part (B) each combine $\widehat{C}_k$ and $\widehat{S}_k$ with the phase weights above, so the sine estimator that the centered atom leaves unused is needed here. The centered atom is the case $p = 0, q = 1, A = 1, B = 0$. A finite sum of Eq. 14 blocks concatenates into one separable inner product, so with J atoms the total width is $F = 1 + Km + 4Jn_{\text{freq}}m$.

2.5 Approximation Error Guarantees

Combining the three blocks, the layer approximates *any* rational-representable activation with a controlled error on a bounded interval. We present the theorem below with detailed proof in App. A.3.

Theorem 1 (Total error on an interval)**.** *Fix $[-T, T]$ containing the pre-activations of interest. Let $\varepsilon_{\text{rat}} = \sup_{|t|\le T}|R(t) - f(t)|$ be the rational fit error, $\varepsilon_{\text{quad}} = \sup_{|t|\le T}|\widehat{g}(t) - g(t)|$ the quadrature-truncation error (here $g = \sum_j g_j$ is the non-polynomial part of R, each g_j of the form Eq. 14, and $\widehat{g}$ replaces each atom's Laplace integral by the retained nodes $s_k \le s_{\max}$), and $V_{\text{poly}}, V_{\text{rat}}$ the variances of the polynomial and rational blocks (V_{rat} bounded by Cor. 1). Then for every $|t| \le T$,*

$$\Big(\mathbb{E}\big[\big(\langle\Phi(\boldsymbol{x}),\,\Psi(\boldsymbol{w})\rangle - f(t)\big)^2\big]\Big)^{1/2} \;\le\; \varepsilon_{\text{rat}} + \varepsilon_{\text{quad}} + \sqrt{V_{\text{poly}} + V_{\text{rat}}}. \tag{16}$$

The three terms are controlled independently: more poles reduce ε_{rat}, more nodes and a larger cutoff reduce $\varepsilon_{\text{quad}}$, and more projections reduce the variance but raising $s_{\max}$ inflates V_{rat} exponentially through Cor. 1, so the scale τ (App. C.1) keeps that exponent in range. The proof, a refinement of $\varepsilon_{\text{quad}}$, and a concentration tail under bounded projections (Cor. 2) are in App. A.3.

The polynomial and rational blocks differ sharply in how their variance grows as we ask for more accuracy. A degree-n monomial has relative variance at least $3^n - 1$, even when $\boldsymbol{x} \parallel \boldsymbol{w}$ (Prop. 2), so pushing the polynomial block to higher accuracy quickly becomes costly. The rational block behaves differently. Adding poles lowers its approximation error, yet its variance does not grow with the number which is set by the input scale alone. We can therefore sharpen the fit by adding poles without increasing variance. App. A.4 makes the cost precise (Thm. 3).

To summarize, we provided a general (Φ, Ψ)-linearized feedforward layer, combining (1) quadrature acting on the Laplace-transform formulation of the centered atoms and (2) random features for the unbiased estimation of the softmax kernel (Choromanski et al., 2021). We call the resulting module a Rationally Linearized Multilayer Perceptron (RaLiMP). RaLiMPs replace MLPs of deep neural networks with more computationally efficient modules. Taking empty sum in Eq. 4 recovers a

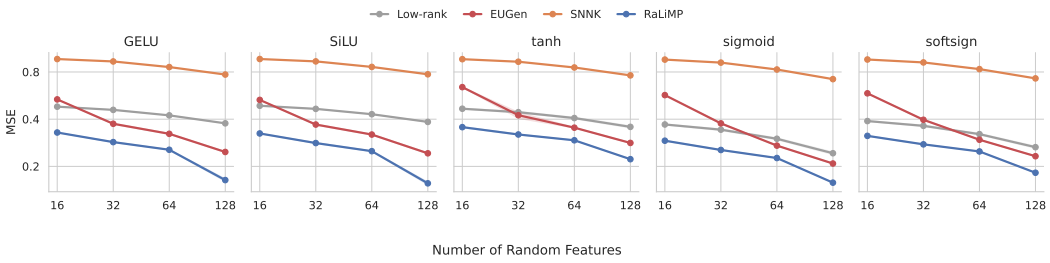

Figure 2: **Synthetic Experiments.** Approximation for various Feedforward layers with different activations. RaLiMPs beat all other published methods. Error for all methods go down as the number of random features increase. All experiments are run using 10 seeds.

polynomial-only layer, so the atoms lower the bias ε_{rat} by construction: at equal polynomial degree they cut it by one to two orders of magnitude on standard activations, and they stay bounded where a polynomial diverges. In the next section, we evaluate the RaLiMP mechanism empirically.

3 Experiments

In this section, we conduct an exhaustive comparison of RaLIMPs with their counterparts and brute-force baselines. We consider the following research questions :

(**RQ1**) How well does RaLiMPs approximate various well-known feed-forward layers?
(**RQ2**) How well does RaLiMPs perform in various Transformer architectures across different modalities like text, 2D vision and 3D modeling?
(**RQ3**) How well does RaLiMPs generalize beyond Transformer architectures?

3.1 Synthetic Experiments

In this experiment, we evaluate the approximation quality of RaLiMP layers. Specifically, we test whether RaLiMP layers can approximate the outputs of a fully-connected feedforward layer (FFL), $\mathbf{Y} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$, where f is an activation function. We compare RaLiMP performance with low-rank approximation, SNNK (Sehanobish et al., 2024) and EUGen (Kim et al., 2025).

We report the results in Fig. 2, where we compute the MSE loss between the FFL output, $\mathbf{Y}$, and RaLiMP prediction, $\hat{\mathbf{Y}}$. We observe that RaLiMP achieve significantly lower MSE loss across all activations, showcasing a superior approximation quality of FFLs (additional details in App. D.1). This provides the answer to (**RQ1**), showing that RaLiMPs are effective in approximating FFLs.

3.2 RaLiMPs in Transformers

Next we focus on replacing feed forward layers in Transformers for various modalities like 3D modeling, image classification, causal language modeling as well as reasoning (**RQ2**).

Vision Tasks: Classification & General Visual Representations We first focus our attention to Vision Transformer (ViT) models (Dosovitskiy et al., 2021). We start with classic benchmarks involving standard classification datasets: CIFAR-100 (Krizhevsky et al., 2009), Places365 (Zhou et al., 2017) and ImageNet-1k (Deng et al., 2009). The results are presented in Fig. 3 (detailed results in Tab. 13). RaLiMP outperforms other efficient MLP substitutes, across different settings (number of layers replaced with the efficient variants). Then, we compare RaLIMP on VTAB-1k (Zhai et al., 2020): a diverse and challenging suite of tasks, designed to evaluate general visual representations (fig. 3 with detailed results in Tab. 14). RaLiMP provides top performance across all datasets and configurations while reducing memory footprint by up to **57**% (Fig. 7). VTAB-1k is the hardest setting for these modules. Each task provides only 1000 training examples which are far too few to adapt these layers to new tasks and overcome the approximation errors introduced. This is why weaker baselines collapse while RaLiMP still maintains performance (details in App. D.2).

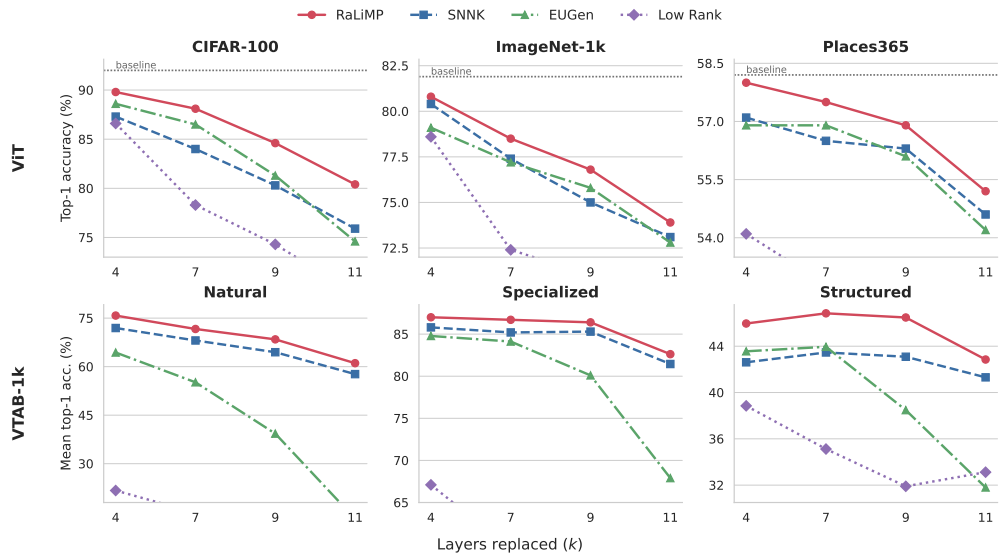

Figure 3: **ViT Results.** *Top Row :* Classification on CiIFAR-100, ImageNet and Places365. *Bottom Row :* Summarized results on VTAB across different image categories. RaLiMP outperforms all variants. Low Rank and EUGen are consistently weak across all methods at high levels of compression. As more layers are replaced, we get up to **57**% reduction in inference parameters.

Table 1: Qwen3-8B accuracy (%) at matched inference budget with the top-k feed-forward layers replaced. For the high-variance benchmarks (AMC23 n=83, AIME24/25 n=30, GPQA n=198) each cell reports **avg@16** with pass@16 beneath it, at t=0.6; the remaining benchmarks are single-sample at a 16k-token thinking-mode budget, and MMLU is the 57-subject accuracy. Within each compression level the best value is in **bold**.

	Competition math (avg@16 / pass@16)			**Reasoning**					
Method	AMC23	AIME24	AIME25	GSM8K	MATH500	Minerva	Olympiad	GPQA	MMLU
k=4 *(last four FFNs replaced)*									
SNNK	76.2 88.0	53.8 68.8	47.5 70.8	92.6	89.9	40.4	66.3	48.2 80.4	71.4
EUGen	79.1 90.6	57.2 72.4	49.4 72.0	93.0	90.2	43.1	65.0	49.8 80.8	72.0
RaLiMP (ours)	**82.2** 96.4	**61.0** 86.0	**52.3** 77.5	**95.5**	**92.8**	**45.3**	**70.4**	**55.6** 85.2	**72.7**
k=8 *(last eight FFNs replaced)*									
SNNK	66.3 84.1	44.8 68.8	39.5 60.8	90.3	87.4	37.8	62.5	42.8 77.6	69.6
EUGen	64.1 82.4	46.3 70.0	41.0 62.8	88.4	86.2	29.8	50.5	35.4 71.3	65.4
RaLiMP (ours)	**78.8** 95.2	**58.1** 79.7	**48.6** 75.3	**95.2**	**94.5**	**43.4**	**69.0**	**51.1** 83.9	**71.9**
Base Qwen3-8B (uncompressed)	83.0 97.6	67.5 90.0	57.7 80.0	93.5	90.0	45.6	69.9	59.3 86.9	73.0

Causal Language Modeling Our next set of experiments involves large language models (LLMs). We start by pretraining GPT-2 Large (770M) on OpenWebText (Gokaslan et al., 2019) on 22B tokens over 30k iterations. We replace the last k layers with RaLiMP and other linearized MLP methods and measure validation perplexity. Fig. 4 shows that RALiMPs provide accuracy improvements over all other methods (detailed results in Tab. 17). Moreover RaLiMP also improves throughput and shrinks memory footprint by up to **50**% (additional training details in App.D.3).

Since Low Rank has been a weak baseline, we drop it in the further expensive experiments.

RaLiMPs in Large Language Models (LLM) We compress Qwen3-8B (Yang et al., 2025a) by replacing its gated (SwiGLU) feed-forward layers with our rational random feature layer. The rational feature map turns SiLU into an inner product, so the gated product $\mathrm{SiLU}(\mathbf{W}\mathbf{x}) \odot \mathbf{U}\mathbf{x}$ becomes a rank-r Hadamard factorization that never forms the wide intermediate activation (details in App. C.5) We train the replacement layers at matched inference budget to EUGen and SNNK,

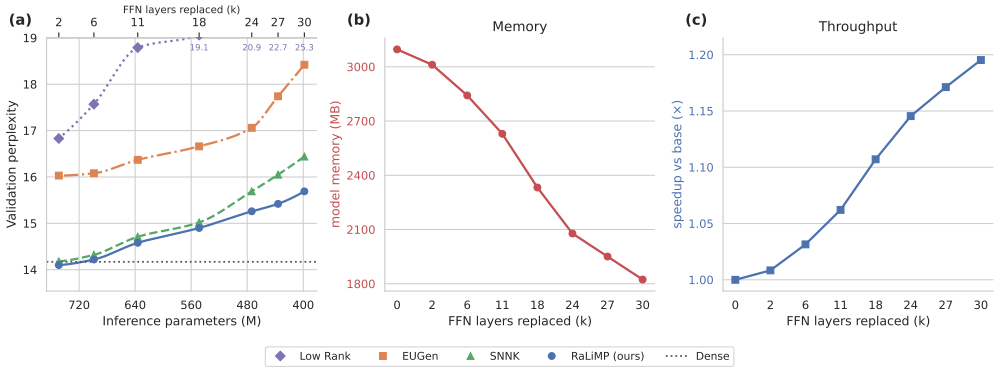

Figure 4: **Casual Language Modeling with GPT2-L (770M) on OpenWebText.** *Left :* Perplexity (Lower is better) RaLiMP shows the best validation perplexity among all the efficient methods considered with matched inference parameters. *Middle :* Memory usage shrinks to almost **50**% as more layers are replaced. *Right :* Throughput increases as well.

and recover the accuracy loss due to compression by on-policy distillation from the uncompressed model (App. D.6). At k=8 the replacement removes 1.17B parameters ($\sim$14% of the model) and decoding throughput rises by 6% over the uncompressed model (Tab. 11). Despite this compression the model stays at near-parity with the base Qwen3-8B across reasoning and knowledge benchmarks: GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021b), Minerva (Lewkowycz et al., 2022), MMLU (Hendrycks et al., 2021a) and OlympiadBench (He et al., 2024) are all within roughly a point with MATH500 and GSM8K slightly exceeding the dense model (Tab. 1). On the hardest competition problems (AIME) (Jia, 2024; Zhang & Math-AI, 2025) the model's pass@16 remains close to the dense model, indicating the underlying capability is largely preserved. Our detailed evaluation shows that RaLiMP can compress a production-scale transformer while retaining its quality. To the best of our knowledge, this is the first time these methods are used for LLMs.

Additional finetuning results on GLUE benchmark (Wang et al., 2018) using DistilBERT (Sanh et al., 2020) shows that our method can be combined with other compression techniques (Fig. 11). RaLiMPs are competitive with baseline while reducing inference parameters by **35**%.

Table 2: Semantic segmentation performance (mIoU) on indoor (ScanNet, and ScanNet200) and outdoor (nuScenes) benchmarks. RaLiMP, SNNK, and EUGen are compared under the same MLP replacement configurations, where k denotes the number of replaced MLP modules in PTv3.

Methods	ScanNet Val			ScanNet200 Val			nuScenes Val		
	MLP modules replaced (k)								
	5	10	22	5	10	22	5	10	22
EUGen	75.65	74.50	72.80	34.62	33.08	30.56	79.14	76.34	74.84
SNNK	76.84	76.63	76.35	35.65	34.69	34.01	80.76	79.92	79.79
RaLiMP (ours)	**78.08**	**77.54**	**77.33**	**36.27**	**35.50**	**34.90**	**80.81**	**80.65**	**80.47**
Baseline (dense, full FT)		77.57			35.26			80.32	

Table 3: Inference efficiency of PTv3 with RaLiMP. k denotes the number of replaced MLP modules. Memory is measured for the full model, while latency is summed over all MLP modules. Measurements use 51,200 points in FP32 on an RTX PRO 6000 Blackwell GPU.

Methods	Model memory (MB) ↓	Peak GPU memory (MB) ↓	MLP latency (ms) ↓
Baseline (dense, full FT)	184.7	849.3	12.31
RaLiMP ($k = 5$)	168.3	832.8	9.23
RaLiMP ($k = 10$)	151.9	817.2	6.52
RaLiMP ($k = 22$)	**149.1**	**814.9**	**6.44**

Semantic Segmentation We focus on the semantic segmentation tasks on ScanNet (Dai et al., 2017), ScanNet200 (Rozenberszki et al., 2022), and nuScenes (Caesar et al., 2020). Using PTv3 (Wu et al., 2024a) as the backbone, we compare RaLiMP with SNNK, EUGen, and the fully fine-tuned dense baseline. The PTv3 backbone contains 22 Transformer blocks, with 14 in the encoder and 8 in the decoder, each containing an MLP module. We consider three MLP replacement configurations, with $k \in \{5, 10, 22\}$ denoting the number of replaced MLP modules. We replace the MLP modules in all Transformer blocks for $k = 22$, and only in encoder and decoder stages with at least 256 channels for $k = 10$. For $k = 5$, we further restrict replacement to the top half of the Transformer layers within each selected stage. As shown in Table 2, RaLiMP consistently outperforms the competing efficient methods across all datasets and replacement configurations. Although segmentation performance generally decreases as more MLP modules are replaced, RaLiMP remains competitive with the dense baseline even at $k = 22$, with gaps of only 0.24 and 0.40 mIoU points on ScanNet and ScanNet200, respectively, while slightly outperforming it on nuScenes. With partial replacement, RaLiMP surpasses the baseline on ScanNet200 by **1.0** mIoU point at $k = 5$ and, at $k = 10$, nearly matches the baseline on ScanNet while outperforming it on ScanNet200 and nuScenes.

Table 3 quantifies the corresponding inference efficiency gains. Compared to the dense baseline, RaLiMP reduces memory by up to **20**%, with up to **2**$\times$ faster MLPs. Notably, $k = 10$ configuration captures most of the efficiency gains: replacing the remaining 12 saves only 2.8 MB of memory and 0.08 ms of MLP latency, while reducing segmentation accuracy across all benchmarks. This demonstrates the favorable accuracy-efficiency tradeoff of targeting MLPs with at least 256 channels.

3.3 Beyond Transformers

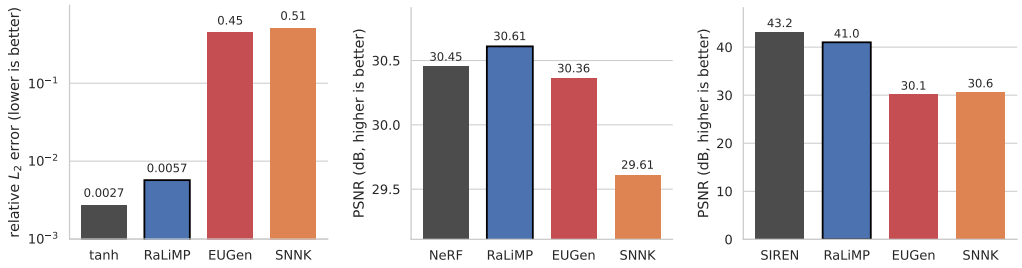

Figure 5: **RaLiMP in PiNNs, NeRF and SIREN.** RaLiMP matches the performance of the baseline PiNN (*Left*) and outperforms NeRF (*Middle*). RaLiMP also matches the SIREN network outperforming the other variants (*Right*). In all cases RaLiMPs reduce model size by **30**% – **50**% while maintaining performance.

We start by targeting NeRF (Mildenhall et al., 2022) architectures that rely on MLPs. We replace regular MLPs with their linearized variations: (1) RaLiMPs, (2) SNNKs (Sehanobish et al., 2024) and (3) EUGens (Kim et al., 2025) following the protocol in (Kim et al., 2025). RALiMPs is the most robust method among all the considered variants, providing top average score across all 3D scenes (Fig. 5 Middle), while reducing model size by **30**% and inference time by **23**%. Detailed NeRF results are presented in Table 12. Moreover we test RaLiMPs in Siren networks (Sitzmann et al., 2020) where we replaced *all* feed forward layers (Fig. 8). RaLiMP match the performance of the dense model while being **50**% smaller (Fig. 5 Right). Finally we apply RaLiMP in Physics Informed Neural Networks (PINNs) (Raissi et al., 2019) for the solution of Burgers equation. RaLiMP close the gap with the dense PINN model while being **42**% smaller. Both EUGen and SNNK struggle to solve this equation (Fig. 5 Left and Fig. 9). For all the experiments, we use the same number of random features for all methods (details in App. D.5. These results affirmatively answer **RQ3**.

4 Conclusion

We presented in this paper Rationally Linearized Multilayer Perceptrons (RaLIMPs) - efficient linearized MLPs capable of approximating various commonly used feed-forward layers. We provide a detailed theoretical analysis, complemented by extensive experiments evaluating RaLiMPs across a broad range of tasks, from implicit 3D modeling to visual representation learning, semantic segmentation, and language modeling. Our results confirm the effectiveness of RaLiMP, as an efficient way of linearizing MLPs to improve their computational footprint with minimal quality loss.

AI USE STATEMENT

In this work, we used generative AI tools for code assistance and running experiments, generating the plots and for grammar correction. We have not used generative AI tools for idea generation, theoretical results, designing the experiments and the general writing of the paper. We developed the theory and the base code. AI was only used to scale and manage some CLM and Qwen experiments. We have manually vetted and reviewed all the code generated by AI as well as made sure that the plots faithfully and accurate describe our results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work presents a method for reducing the computational cost of multilayer perceptrons using kernel approximations. It does not involve human subjects, personal data, or automated decision-making. We are not aware of any specific ethical or societal risks arising directly from the proposed method beyond the general potential for more efficient machine learning systems to be used in applications with societal impacts.

REPRODUCIBILITY STATEMENT

Our mechanism is defined in equation 7. We detail the hardware as well as the hyperparameters for all the experiments in Sec. D. All datasets used are public datasets and are referenced in the paper. We will open source our code upon acceptance of the manuscript.

REFERENCES

Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. US Government printing office, 1964.

AI-MO. Aimo validation amc, 2024. URL https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Nir Ailon and Edo Liberty. An almost optimal unrestricted fast johnson-lindenstrauss transform. *ACM Transactions on Algorithms (TALG)*, 9(3):1–12, 2013.

Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russell Webb. Distillation scaling laws. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 5977–6045. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/busbridge25a.html.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Largekernel3d: Scaling up kernels in 3d sparse cnns. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13488–13498. IEEE, 2023.

Hung-Yueh Chiang, Yen-Liang Lin, Yueh-Cheng Liu, and Winston H Hsu. A unified point-based framework for 3d segmentation. In *2019 International Conference on 3D Vision (3DV)*, pp. 155–163. IEEE, 2019.

Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.

Youngmin Cho and Lawrence K Saul. Analysis and extension of arc-cosine kernels for large margin classification. *arXiv preprint arXiv:1112.3712*, 2011.

Krzysztof M Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. *Advances in neural information processing systems*, 30, 2017.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3075–3084, 2019.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5828–5839, 2017.

Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. Learning fast algorithms for linear transforms using butterfly factorizations. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1517–1527. PMLR, 2019. URL http://proceedings.mlr.press/v97/dao19a.html.

Tri Dao, Beidi Chen, Nimit Sharad Sohoni, Arjun D. Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4690–4721. PMLR, 2022. URL https://proceedings.mlr.press/v162/dao22a.html.

Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse johnson: Lindenstrauss transform. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 341–350, 2010.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Pedro Domingos. Every model learned by gradient descent is approximately a kernel machine. *arXiv preprint arXiv:2012.00152*, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at

scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Yasser Elouargui, Mahmoud Zyate, Abdellatif Sassioui, Meriyem Chergui, Mohamed El Kamili, and Mohammed Ouzzif. A comprehensive survey on efficient transformers. In *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1–6, 2023. doi: 10.1109/WINCOM59760.2023.10322921.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9224–9232, 2018.

Etash Kumar Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Rea Sprague, Ashima Suvarna, Benjamin Feuer, Leon Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah M Pratt, Vivek Ramanujan, Jon Saad-Falcon, Stutee Acharya, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alex Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7xjoTuaNmN.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre.

Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Yuenan Hou, Xinge Zhu, Yuexin Ma, Chen Change Loy, and Yikang Li. Point-to-voxel knowledge distillation for lidar semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

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*, 2022a. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. Efficientnerf-efficient neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12892–12901. IEEE, 2022b.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Maxwell Jia. Aime problem set 2024, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024.

Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. PolySketchFormer: Fast transformers via sketching polynomial kernels. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 22748–22770. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/kacham24a.html.

Purushottam Kar and Harish Karnick. Random feature maps for dot product kernels. In *Artificial intelligence and statistics*, pp. 583–591. PMLR, 2012.

Sang Min Kim, Byeongchan Kim, Arijit Sehanobish, Somnath Basu Roy Chowdhury, Rahul Kidambi, Dongseok Shim, Kumar Avinava Dubey, Snigdha Chaturvedi, Min-hwan Oh, and Krzysztof Marcin Choromanski. Eugens: Efficient, unified and general dense layers. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Lingdong Kong, Youquan Liu, Runnan Chen, Yuexin Ma, Xinge Zhu, Yikang Li, Yuenan Hou, Yu Qiao, and Ziwei Liu. Rethinking range view representation for lidar segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research), 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8500–8509, 2022.

Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 2024.

Lei Li, Yongfeng Zhang, and Li Chen. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pp. 1348–1357, 2023.

Zhuoyuan Li, Yubo Ai, Jiahao Lu, ChuXin Wang, Jiacheng Deng, Hanzhi Chang, Yanzhe Liang, Wenfei Yang, Shifeng Zhang, and Tianzhu Zhang. Pamba: enhancing global interaction in point clouds via state space model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 5092–5100, 2025a.

Zimeng Li, Hongjun Li, Jingyuan Wang, and Ke Tang. Approximation to smooth functions by low-rank swish networks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025b. URL https://proceedings.mlr.press/v267/li25bb.html.

Valerii Likhosherstov, Krzysztof M Choromanski, Kumar Avinava Dubey, Frederick Liu, Tamas Sarlos, and Adrian Weller. Chefs' random tables: Non-trigonometric random features. *Advances in Neural Information Processing Systems*, 35:34559–34573, 2022.

Haojia Lin, Xiawu Zheng, Lijiang Li, Fei Chao, Shanshan Wang, Yan Wang, Yonghong Tian, and Rongrong Ji. Meta architecture for point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17682–17691, 2023.

Jiuming Liu, Ruiji Yu, Yian Wang, Yu Zheng, Tianchen Deng, Weicai Ye, and Hesheng Wang. Point mamba: A novel point cloud backbone based on state space model with octree-based ordering strategy. *arXiv preprint arXiv:2403.06467*, 2024a.

Youquan Liu, Lingdong Kong, Xiaoyang Wu, Runnan Chen, Xin Li, Liang Pan, Ziwei Liu, and Yuexin Ma. Multi-space alignments towards universal lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024b.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: LLM quantization with learned rotations. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ogO6DGE6FZ.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, pp. 23610–23641. PMLR, 2023.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65 (1):99–106, 2022. doi: 10.1145/3503250. URL https://doi.org/10.1145/3503250.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry P. Vetrov. Tensorizing neural networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 442–450, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/6855456e2fe46a9d49d3d3af4f57443d-Abstract.html.

Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotný, Michael Zollhöfer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. In Kris Hauser, Dylan A. Shell, and Shoudong Huang (eds.), *Robotics: Science and Systems XVIII, New York City, NY, USA, June 27 - July 1, 2022*, 2022. doi: 10.15607/RSS.2022.XVIII.012. URL https://doi.org/10.15607/RSS.2022.XVIII.012.

Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16928–16937. IEEE, 2022.

Bohao Peng, Xiaoyang Wu, Li Jiang, Yukang Chen, Hengshuang Zhao, Zhuotao Tian, and Jiaya Jia. Oa-cnns: Omni-adaptive sparse cnns for 3d semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024.

Pointcept Contributors. Pointcept: A codebase for point cloud perception research. https://github.com/Pointcept/Pointcept, 2023.

Andres Potapczynski, Shikai Qiu, Marc Finzi, Christopher Ferri, Zixi Chen, Micah Goldblum, C Bayan Bruss, Christopher De, and Andrew Wilson. Searching for efficient linear layers over a continuous space of structured matrices. *Advances in Neural Information Processing Systems*, 37:3857–3881, 2024.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in neural information processing systems*, 35:23192–23204, 2022.

Shikai Qiu, Andres Potapczynski, Marc Anton Finzi, Micah Goldblum, and Andrew Gordon Wilson. Compute better spent: Replacing dense layers with structured matrices. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 41698–41716. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/qiu24f.html.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis (eds.), *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pp. 1177–1184. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou (eds.), *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 1313–1320. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper/2008/hash/0efe32849d230d7f53049ddc4a4b0c60-Abstract.html.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14315–14325. IEEE, 2021.

Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (ToG)*, 42(4):1–12, 2023.

David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *European conference on computer vision*, pp. 125–141. Springer, 2022.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Arijit Sehanobish, Krzysztof Marcin Choromanski, Yunfan Zhao, Kumar Avinava Dubey, and Valerii Likhosherstov. Scalable neural network kernels. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=4iPw1klFWa.

Vikas Sindhwani, Tara N. Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 3088–3096, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/851300ee84c2b80ed40f51ed26d866fc-Abstract.html.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7462–7473. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf.

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, et al. Llm pruning and distillation in practice: The minitron approach. *arXiv preprint arXiv:2408.11796*, 2024.

Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5449–5459. IEEE, 2022.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8248–8258, 2022.

Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, 2020.

Maosen Tang and Alex Townsend. Rational neural networks have expressivity advantages. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=gPQQyHFbEA.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6):109:1–109:28, 2023. doi: 10.1145/3530811. URL https://doi.org/10.1145/3530811.

Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6411–6420, 2019.

Akiyoshi Tomihari and Issei Sato. Understanding linear probing then fine-tuning language models from ntk perspective. *Advances in Neural Information Processing Systems*, 37:139786–139822, 2024.

Jonas Wacker and Maurizio Filippone. Local random feature approximations of the gaussian kernel. *Procedia Computer Science*, 207:987–996, 2022.

Jonas Wacker, Ruben Ohana, and Maurizio Filippone. Complex-to-real sketches for tensor products with applications to the polynomial kernel. In *International Conference on Artificial Intelligence and Statistics*, pp. 5181–5212. PMLR, 2023.

Jonas Wacker, Motonobu Kanagawa, and Maurizio Filippone. Improved random features for dot product kernels. *Journal of Machine Learning Research*, 25(235):1–75, 2024.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446/.

Peng-Shuai Wang. Octformer: Octree-based transformers for 3d point clouds. *ACM Transactions on Graphics (TOG)*, 42(4):1–11, 2023.

Alexander Wei, Wei Hu, and Jacob Steinhardt. More than a toy: Random matrix models predict how real-world neural representations generalize. In *International conference on machine learning*, pp. 23549–23588. PMLR, 2022.

Xiuying Wei, Skander Moalla, Razvan Pascanu, and Caglar Gulcehre. Building on efficient foundations: Effective training of llms with structured feedforward layers. *Advances in neural information processing systems*, 37:4689–4717, 2024.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 9621–9630, 2019.

Wenxuan Wu, Li Fuxin, and Qi Shan. Pointconvformer: Revenge of the point-based convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 21802–21813, 2023.

Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35:33330–33342, 2022.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4840–4851. IEEE, 2024a.

Xiaoyang Wu, Zhuotao Tian, Xin Wen, Bohao Peng, Xihui Liu, Kaicheng Yu, and Hengshuang Zhao. Towards large-scale 3d representation learning with multi-dataset point prompt training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024b.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. Grid-guided neural radiance fields for large urban scenes. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8296–8306. IEEE, 2023.

Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5589–5598, 2020.

Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *European conference on computer vision*, 2022.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. *Computational Visual Media*, 11(1):83–101, 2025b.

Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. *Advances in neural information processing systems*, 29, 2016.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. Mammoth2: Scaling instructions from the web. *Advances in Neural Information Processing Systems*, 2024.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. A large-scale study of representation learning with the visual task adaptation benchmark, 2020. URL https://arxiv.org/abs/1910.04867.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 16259–16268, 2021.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209, 2024.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021.

A THEORETICAL RESULTS

In this section we discuss additional theoretical results.

A.1 UNBIASEDNESS OF THE RATIONAL RANDOM FEATURES

We present the complete proof of Lemma 1, which we restate for convenience of the reader.

Lemma 2 (Unbiasedness). *Let $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ and $t = \boldsymbol{w}\cdot\boldsymbol{x}$. Then for $\tau = 1$,*

$$\mathbb{E}\big[\widehat{C}_k\big] = \cos(s_k t), \qquad \mathbb{E}\big[\widehat{S}_k\big] = \sin(s_k t), \tag{17}$$

and for general $\tau > 0$, applying this to $\boldsymbol{x}/\tau$ and $\boldsymbol{w}/\tau$ gives $\mathbb{E}[\widehat{C}_k] = \cos(s_k t/\tau^2)$ and $\mathbb{E}[\widehat{S}_k] = \sin(s_k t/\tau^2)$.

Proof. Drop r and write $Z_x = \sqrt{s/2}\,\boldsymbol{\omega}\cdot\boldsymbol{x}$, $Z_w = \sqrt{s/2}\,\boldsymbol{\omega}\cdot\boldsymbol{w}$, $h_x = \frac{s}{2}\|\boldsymbol{x}\|^2$, $h_w = \frac{s}{2}\|\boldsymbol{w}\|^2$. By angle addition the $\widehat{C}_k$ summand collapses to a single feature,

$$\varphi^{\cos}(\boldsymbol{x})\varphi^{\cos}(\boldsymbol{w}) - \varphi^{\sin}(\boldsymbol{x})\varphi^{\sin}(\boldsymbol{w}) = \tfrac{1}{m}\, e^{Z_x + Z_w}\cos\big((Z_x - h_x) + (Z_w - h_w)\big) = \tfrac{1}{m}\, e^{\xi}\cos(\xi - H), \tag{18}$$

with $\xi := \sqrt{s/2}\,\boldsymbol{\omega}\cdot(\boldsymbol{x}+\boldsymbol{w})$ and $H := h_x + h_w$; ξ is centered Gaussian with $\mathrm{Var}(\xi) = \frac{s}{2}\|\boldsymbol{x}+\boldsymbol{w}\|^2$. Since $e^{\xi}\cos\xi = \mathrm{Re}\, e^{(1+i)\xi}$ and $(1+i)^2 = 2i$, the Gaussian moment generating function gives $\mathbb{E}\, e^{(1+i)\xi} = e^{\frac{1}{2}(2i)\mathrm{Var}(\xi)} = e^{i\varphi}$ with $\varphi := \frac{s}{2}\|\boldsymbol{x}+\boldsymbol{w}\|^2$, hence $\mathbb{E}[e^{\xi}\cos\xi] = \cos\varphi$ and $\mathbb{E}[e^{\xi}\sin\xi] = \sin\varphi$. Therefore $\mathbb{E}[e^{\xi}\cos(\xi - H)] = \cos(H - \varphi)$, and $H - \varphi = \frac{s}{2}(\|\boldsymbol{x}\|^2 + \|\boldsymbol{w}\|^2 - \|\boldsymbol{x}+\boldsymbol{w}\|^2) = -st$, giving $\mathbb{E}\,\widehat{C}_k = \cos(st)$. The $\widehat{S}_k$ case is identical with $\mathrm{Re} \to \mathrm{Im}$. For general τ the argument enters as $\boldsymbol{u}/\tau$ on both sides, so $\boldsymbol{x}, \boldsymbol{w}$ are replaced by $\boldsymbol{x}/\tau, \boldsymbol{w}/\tau$ throughout and the estimators become unbiased for $\cos(st/\tau^2)$ and $\sin(st/\tau^2)$. □

A.2 VARIANCE OF THE RATIONAL RANDOM FEATURES

We compute the second moments of the estimators of Eqs. 8–9 exactly, proving Proposition 1. Throughout, fix a quadrature node s, evaluate the features at $\boldsymbol{u}/\tau$, and set

$$\sigma^2 = \tfrac{s}{2}\|\boldsymbol{x}+\boldsymbol{w}\|^2/\tau^2, \qquad H = \tfrac{s}{2}\big(\|\boldsymbol{x}\|^2 + \|\boldsymbol{w}\|^2\big), \tag{19}$$

as in the main text. Let $Y_r^c = m\big[\varphi^{\cos}(\boldsymbol{x})_r\varphi^{\cos}(\boldsymbol{w})_r - \varphi^{\sin}(\boldsymbol{x})_r\varphi^{\sin}(\boldsymbol{w})_r\big]$ and $Y_r^s = m\big[\varphi^{\cos}(\boldsymbol{x})_r\varphi^{\sin}(\boldsymbol{w})_r + \varphi^{\sin}(\boldsymbol{x})_r\varphi^{\cos}(\boldsymbol{w})_r\big]$, so that $\widehat{C}_k = \frac{1}{m}\sum_r Y_r^c$ and $\widehat{S}_k = \frac{1}{m}\sum_r Y_r^s$ are averages of m i.i.d. terms.

Proposition 1 *For each projection, $\mathbb{E}[(Y^c)^2] = \frac{1}{2}\big(e^{2\sigma^2} + \cos(4\sigma^2 - 2H/\tau^2)\big)$ and $\mathbb{E}[(Y^s)^2] = \frac{1}{2}\big(e^{2\sigma^2} - \cos(4\sigma^2 - 2H/\tau^2)\big)$. Consequently, averaging over m i.i.d. projections,*

$$\mathrm{Var}\big(\widehat{C}_k\big) \;\le\; \frac{1}{2m}\Big(e^{2\sigma^2} + 1\Big), \qquad \mathrm{Var}\big(\widehat{C}_k\big) + \mathrm{Var}\big(\widehat{S}_k\big) \;=\; \frac{1}{m}\Big(e^{2\sigma^2} - 1\Big).$$

Proof. Drop the index r. By the angle addition used in Lemma 1, $Y^c = e^{\xi}\cos(\xi - H/\tau^2)$ and $Y^s = e^{\xi}\sin(\xi - H/\tau^2)$ with $\xi = \sqrt{s/2}\,\boldsymbol{\omega}\cdot(\boldsymbol{x}+\boldsymbol{w})/\tau \sim \mathcal{N}(0, \sigma^2)$; that lemma also gives $\mathbb{E}[Y^c] = \cos(st/\tau^2)$ and $\mathbb{E}[Y^s] = \sin(st/\tau^2)$. For the second moments write $\cos^2\theta = \frac{1}{2}(1 + \cos 2\theta)$ and $\sin^2\theta = \frac{1}{2}(1 - \cos 2\theta)$, so that

$$\mathbb{E}\big[(Y^c)^2\big] = \tfrac{1}{2}\mathbb{E}\big[e^{2\xi}\big] + \tfrac{1}{2}\mathbb{E}\big[e^{2\xi}\cos(2\xi - 2H/\tau^2)\big]. \tag{20}$$

The first term is $e^{2\sigma^2}$ by the Gaussian moment generating function. For the second, note $e^{2\xi}\cos(2\xi - 2H/\tau^2) = \mathrm{Re}\,\big[e^{-2iH/\tau^2}e^{(2+2i)\xi}\big]$ and $(2+2i)^2 = 8i$, hence $\mathbb{E}[e^{(2+2i)\xi}] = e^{\frac{1}{2}(8i)\sigma^2} = e^{4i\sigma^2}$ and the term equals $\cos(4\sigma^2 - 2H/\tau^2)$. The Y^s case is identical with the opposite sign. Subtracting $(\mathbb{E}[Y^c])^2$ and averaging over m i.i.d. projections gives the variance in Eq. 12; the bound follows from $\cos(\cdot) \le 1$ and $\cos^2(\cdot) \ge 0$. For Eqn. 13, add the two second moments, the cosine terms cancel, leaving $e^{2\sigma^2}$ and $\cos^2(\cdot) + \sin^2(\cdot) = 1$. □

The variance depends on the inputs only through $\|\boldsymbol{x}+\boldsymbol{w}\|$ and grows exponentially in $2\sigma^2$ and it is symmetric between the cosine and sine channels, so the general shifted atom of Sec. 2.4 incurs no additional cost per channel.

Corollary 1. *Draw independent projections per quadrature node, and write $\rho = \|\boldsymbol{x}+\boldsymbol{w}\|$.*

(i) For the centered atom estimator $\widehat{R} = \sum_k \gamma_k \widehat{C}_k$,

$$\mathrm{Var}\big(\widehat{R}\big) = \sum_k \gamma_k^2 \,\mathrm{Var}\big(\widehat{C}_k\big) \;\le\; \frac{1}{2m}\Big(e^{s_{\max}\rho^2/\tau^2}+1\Big). \tag{21}$$

(ii) For the general atom of Eq. 14, whose estimator is $\widehat{g} = \sum_k \gamma_k\big(w_{c,k}\widehat{C}_k + w_{s,k}\widehat{S}_k\big)$ at frequencies s_k/q,

$$\mathrm{Var}\big(\widehat{g}\big) \;\le\; \frac{c_A^2+c_B^2}{m}\Big(e^{s_{\max}\rho^2/(q\tau^2)}-1\Big), \qquad c_A = \frac{|A|}{q^2}, \quad c_B = \frac{|B|}{q}. \tag{22}$$

Proof. Both use $\gamma_k > 0$ and $\sum_k \gamma_k = 1$, hence $\sum_k \gamma_k^2 \le 1$, together with independence across nodes. For (ii), the per-node coefficients of Sec. 2.4 are the rotation $w_{c,k} = c_A\cos\phi_k - c_B\sin\phi_k$, $w_{s,k} = c_A\sin\phi_k + c_B\cos\phi_k$ with $\phi_k = s_k p/q$, so $w_{c,k}^2 + w_{s,k}^2 = c_A^2 + c_B^2$ for every k, and Cauchy–Schwarz gives $\mathrm{Var}(\alpha U + \beta V) \le (\alpha^2+\beta^2)(\mathrm{Var}\,U + \mathrm{Var}\,V)$ for any α,β and any U,V; applying this with Eq. 13 at frequency s_k/q, for which $2\sigma_k^2 = s_k\rho^2/(q\tau^2)$, yields the claim. □

The exponent carries q in the denominator, so for a sum of atoms the variance is governed by the *narrowest* pole width $q_{\min} = \min_j q_j$, and not by $s_{\max}$ and τ alone.

A.3 Total Approximation Error

Three errors accumulate: the rational fit, the truncated quadrature, and the finite projection count. Only the last is random, and Prop. 1 controls it. Since P is unbounded on $\mathbb{R}$, we focus our attention on an interval $[-T,T]$ containing the pre-activations of interest.

Theorem 1 gives an upper bound to the total L^2 error. We restate the theorem and prove it below.

Theorem 2 (Total error on an interval)**.** *Fix $T > 0$ and a target activation $f : [-T,T] \to \mathbb{R}$. Let $R = P+g$ with $g = \sum_j g_j$, each g_j of the form Eq. 14, be the rational surrogate, its poles expressed in the rescaled variable of App. C.1 so that f and R are compared in the variable the features are evaluated at, and let $\widehat{g}$ denote the deterministic value of the truncated quadrature, i.e. g with its Laplace integral replaced by the nodes $\{(s_k,\gamma_k) : s_k \le s_{\max}\}$. Put*

$$\varepsilon_{\mathrm{rat}} = \sup_{|t|\le T}\big|f(t)-R(t)\big|, \qquad \varepsilon_{\mathrm{quad}} = \sup_{|t|\le T}\big|g(t)-\widehat{g}(t)\big|. \tag{23}$$

Then for every $\boldsymbol{x},\boldsymbol{w}$ with $t = \boldsymbol{w}\cdot\boldsymbol{x} \in [-T,T]$ the RaLiMP estimator is unbiased for the truncated surrogate, $\mathbb{E}\big[\langle\Phi(\boldsymbol{x}),\,\Psi(\boldsymbol{w})\rangle\big] = P(t) + \widehat{g}(t)$ (stated for $\tau = 1$; general τ replaces t by t/τ^2 inside $\widehat{g}$, cf. App. C.1), and

$$\Big(\mathbb{E}\big[\big(\langle\Phi(\boldsymbol{x}),\,\Psi(\boldsymbol{w})\rangle - f(t)\big)^2\big]\Big)^{1/2} \;\le\; \varepsilon_{\mathrm{rat}} + \varepsilon_{\mathrm{quad}} + \sqrt{V_{\mathrm{poly}} + V_{\mathrm{rat}}}, \tag{24}$$

where V_{rat} is bounded by Cor. 1(ii) and V_{poly} is the variance of the polynomial block. Furthermore $\varepsilon_{\mathrm{quad}} \le \varepsilon_{\mathrm{GL}} + (c_A+c_B)\sum_{s_k > s_{\max}}\gamma_k$, where $\varepsilon_{\mathrm{GL}}$ is the Gauss–Laguerre error at full node count; the same bound holds per atom, and summing over the J atoms gives the truncation error for g.

Proof. Unbiasedness is Lemma 1 per node and channel, plus that of the polynomial block. Decompose

$$\langle\Phi(\boldsymbol{x}),\,\Psi(\boldsymbol{w})\rangle - f(t) = \underbrace{\Big(\langle\Phi(\boldsymbol{x}),\,\Psi(\boldsymbol{w})\rangle - P(t) - \widehat{g}(t)\Big)}_{\text{mean zero}} + \underbrace{\big(\widehat{g}(t)-g(t)\big)}_{|\cdot|\le\varepsilon_{\mathrm{quad}}} + \underbrace{\big(R(t)-f(t)\big)}_{|\cdot|\le\varepsilon_{\mathrm{rat}}}. \tag{25}$$

The last two terms are deterministic, so the L^2 triangle inequality gives Eq. 16, the first term's L^2 norm being the square root of its variance; the two blocks use independent projections, so that variance is $V_{\mathrm{poly}} + V_{\mathrm{rat}}$. For the last claim, the omitted nodes each contribute at most γ_k times the channel coefficients, as $|\cos|, |\sin| \le 1$. □

The trade-off is explicit: more poles cut the first term, more nodes and a larger cutoff the second, more projections the third, but raising $s_{\max}$ inflates V_{rat} exponentially through Cor. 1, so the last two can not be varied independently. The scale τ of App. C.1 keeps that exponent in range.

An exponential tail. The summands of Prop. 1 are lognormal-tailed, with infinite moment generating function, so Hoeffding- and Bernstein-type bounds need the projections to be bounded.

Corollary 2 (Concentration under bounded projections)**.** *Fix a node $s \le s_{\max}$ and suppose $|\boldsymbol{\omega}_r \cdot (\boldsymbol{x}+\boldsymbol{w})| \le \beta\|\boldsymbol{x}+\boldsymbol{w}\|$ for all $r = 1, \dots, m$. Then with $B = \exp\big(\beta\|\boldsymbol{x}+\boldsymbol{w}\|\sqrt{s_{\max}/2}\,/\tau\big)$, for any $\varepsilon > 0$,*

$$\mathbb{P}\Big[\big|\widehat{C}_k - \cos(s_k t/\tau^2)\big| \ge \varepsilon\Big] \le 2\exp\Big(-\frac{m\varepsilon^2}{2B^2}\Big), \tag{26}$$

and likewise for $\widehat{S}_k$.

Proof. The hypothesis gives $|\xi_r| \le \sqrt{s/2}\,\beta\|\boldsymbol{x}+\boldsymbol{w}\|/\tau$, so $|Y_r^c| \le e^{|\xi_r|} \le B$ a.s. The Y_r^c are independent with mean $\cos(s_k t/\tau^2)$ by Lemma 1 and $\widehat{C}_k = \frac{1}{m}\sum_r Y_r^c$, so Hoeffding's inequality applies with range $2B$. □

The hypothesis is not vacuous: $\boldsymbol{\omega}_r \cdot \boldsymbol{v}/\|\boldsymbol{v}\|$ is standard normal, so a union bound gives $\beta = \sqrt{2\log(2m/\delta)}$ with probability $\ge 1-\delta$, whence $B = \exp\big(\|\boldsymbol{x}+\boldsymbol{w}\|\sqrt{s_{\max}\log(2m/\delta)}/\tau\big)$: the failure probability is explicit, and the constant is the same exponential in $\|\boldsymbol{x}+\boldsymbol{w}\|/\tau$ that governs the variance.

A.4 Cost of Accuracy

Sec. 2.2 realizes degree-n monomials as products of n independent linear projections. For any such feature map the required width is exponential in n. The variance computation rests on the following classical fact about Gaussian moments.

Lemma 3 (Isserlis' theorem)**.** *Let $(X_1, \dots, X_{2n})$ be a centered jointly Gaussian vector. Then*

$$\mathbb{E}\Big[\prod_{i=1}^{2n} X_i\Big] = \sum_{\textit{pairings}} \prod_{\{i,j\}} \mathbb{E}[X_i X_j], \tag{27}$$

the sum running over all partitions of $\{1, \dots, 2n\}$ into pairs $\{i, j\}$ (moments of odd order vanish). In particular, the fourth moment is

$$\mathbb{E}[X_1X_2X_3X_4] = \mathbb{E}[X_1X_2]\,\mathbb{E}[X_3X_4] + \mathbb{E}[X_1X_3]\,\mathbb{E}[X_2X_4] + \mathbb{E}[X_1X_4]\,\mathbb{E}[X_2X_3]. \tag{28}$$

Proposition 2 (Variance of degree-n product features)**.** *Let $\boldsymbol{G}^{(1)}, \dots, \boldsymbol{G}^{(n)} \in \mathbb{R}^{m\times d}$ have i.i.d. $\mathcal{N}(0,1)$ entries, independent across i, and put $X_r = \prod_{i=1}^n \big(\boldsymbol{G}_r^{(i)} \cdot \boldsymbol{x}\big)\big(\boldsymbol{G}_r^{(i)} \cdot \boldsymbol{w}\big)$, so $\frac{1}{m}\sum_r X_r$ estimates t^n. With $\kappa = t/(\|\boldsymbol{x}\|\|\boldsymbol{w}\|)$,*

$$\mathrm{Var}(X_r) = \big(\|\boldsymbol{x}\|^2\|\boldsymbol{w}\|^2 + 2t^2\big)^n - t^{2n}, \qquad \frac{\mathrm{Var}(X_r)}{t^{2n}} = \Big(\frac{1+2\kappa^2}{\kappa^2}\Big)^n - 1 \;\ge\; 3^n - 1, \tag{29}$$

with equality iff $\boldsymbol{x} \parallel \boldsymbol{w}$. Hence relative accuracy η needs $m \ge (3^n-1)/\eta^2$.

Proof. Independence across i gives $\mathbb{E}[X_r] = t^n$ and $\mathbb{E}[X_r^2] = \prod_i \mathbb{E}[(\boldsymbol{G}_r^{(i)}\cdot\boldsymbol{x})^2(\boldsymbol{G}_r^{(i)}\cdot\boldsymbol{w})^2]$. For one Gaussian row, $(\boldsymbol{g}\cdot\boldsymbol{x}, \boldsymbol{g}\cdot\boldsymbol{w})$ is centered jointly Gaussian with variances $\|\boldsymbol{x}\|^2, \|\boldsymbol{w}\|^2$ and covariance t, so Isserlis' theorem (Lemma 3) gives second moment $\|\boldsymbol{x}\|^2\|\boldsymbol{w}\|^2 + 2t^2$: the like pairing of the two $\boldsymbol{g}\cdot\boldsymbol{x}$ and the two $\boldsymbol{g}\cdot\boldsymbol{w}$ factors contributes $\|\boldsymbol{x}\|^2\|\boldsymbol{w}\|^2$, and the two cross pairings contribute t^2 each. Substituting $\|\boldsymbol{x}\|^2\|\boldsymbol{w}\|^2 = t^2/\kappa^2$ gives the second form; $\kappa^{-2} + 2$ is minimized at $\kappa^2 = 1$, where it equals 3. □

The bound assumes nothing about the geometry and places no bound on the projection entries, which Gaussian rows would violate. The 3^n floor is *attained* at $\boldsymbol{x} \parallel \boldsymbol{w}$, so conditioning on the inputs cannot remove the growth in n.

The contrast with the rational block is: accuracy and cost are controlled by different quantities: the pole count sets the approximation error, while the variance exponent is set by the input scale alone.

Theorem 3 (Accuracy and cost are separate factors)**.** *Let $\|\boldsymbol{x}\| = \|\boldsymbol{w}\|$ with $|\kappa| \geq c_0 > 0$, and let $\tau^2 = c\,\mathrm{RMS}(\boldsymbol{w}\cdot\boldsymbol{x})$ as in App. C.1. Then*

$$\frac{\|\boldsymbol{x}+\boldsymbol{w}\|^2}{\tau^2} \;\leq\; \frac{4}{c\,c_0}, \tag{30}$$

independent of d, so by Theorem. 1 and Cor. 1 an L^2 error ε is attained with $m_{\mathrm{rat}} = O\big(\varepsilon^{-2} e^{4s_{\max}/(q_{\min} c\, c_0)}\big)$, with $q_{\min}$ the narrowest pole width (Cor. 1); the exponent does not depend on ε, on the number of poles, or on d. A degree-K product-projection block instead needs, by Prop. 2,

$$m_{\mathrm{poly}} \;\geq\; (3^K - 1)\left(c_K\, t^K\right)^2/\varepsilon^2, \tag{31}$$

with c_K the leading coefficient and $K = K(\varepsilon)$ the degree required for approximation error $\varepsilon/2$; for non-polynomial f, $K(\varepsilon) \to \infty$.

Proof. For Eq. 30, $\|\boldsymbol{x}\| = \|\boldsymbol{w}\| = r$ gives $\|\boldsymbol{x}+\boldsymbol{w}\|^2 = 2r^2(1+\kappa) \leq 4r^2$, while $|t| = |\kappa| r^2 \geq c_0 r^2$ for every such pair, so $\tau^2 = c\,\mathrm{RMS}(t) \geq c\,c_0 r^2$. Eq. 31 follows by discarding all but the degree-K term of the block variance, at least $c_K^2(3^K-1)t^{2K}/m$ by Prop. 2, and requiring standard deviation at most ε. □

Equation. 31 is only informative if $c_K t^K$ does not vanish as K grows. For $f = \mathrm{silu}$ on $[-T, T]$ we find $|c_K t^K| = \Theta(1)$ at $|t| = T$ across $K = 6, \ldots, 16$, so the 3^K survives and the required width scales as $\varepsilon^{-2-\alpha}$ with $\alpha > 0$ growing with T.

Remark 1. *Consider an atom whose poles lie a distance q from $[-T, T]$. Approximating it to error ε by polynomials needs degree $K \sim \frac{T}{q}\log\frac{1}{\varepsilon}$, so by Eq. 31 the polynomial block uses $\varepsilon^{-\Theta(T/q)}$ features, while the rational block reaches the same ε with $O_q(\varepsilon^{-2})$. A narrow pole (small q) thus enters the rational cost as a constant and the polynomial cost as a growing exponent in ε.*

B Unified Kernel Representation

We relate our construction to prior kernel representations of MLPs (Kim et al., 2025; Sehanobish et al., 2024), expanding on Sec. 2.3.1. All three fit the common representation

$$R(t) = \int_{\Xi} K_\xi(t)\, d\mu(\xi), \qquad K_\xi(\boldsymbol{x}^\top \boldsymbol{w}) = \mathbb{E}_\omega[\phi_{\xi,\omega}(\boldsymbol{x})\psi_{\xi,\omega}(\boldsymbol{w})]\,, \tag{32}$$

with each scalar kernel K_ξ carried by separable features, and differ only in how the spectral measure μ is obtained. For the polynomial block $\Xi = \mathbb{N}$ and $K_n(t) = t^n$, so μ is the signed counting measure carrying the coefficients c_n, at the cost of Prop. 2.

Fourier expansion of a given activation. Universal random features (Sehanobish et al., 2024) read μ off the Fourier transform of an activation f supplied as input, decomposed into nonnegative parts and sampled through proposal densities; the resulting $\exp(\hat{\boldsymbol{x}}^\top \hat{\boldsymbol{w}})$ is then handled by a softmax-kernel linearization of the kind used in Sec. 2.3. We share that device, an oscillatory integrand against a nonnegative measure, and differ in the direction of the construction: there f is given and μ is derived from it, here μ is the parameterized object and f is whatever it produces.

Partial fractions. For an irreducible-quadratic atom the transform is already a single nonnegative exponential density: by Sec. 2.4, μ is $e^{-s}ds$ at frequency s/q for both the cosine and the sine term, with coefficients $c_A = |A|/q^2$ and $c_B = |B|/q$ (Cor. 1), so no decomposition or reweighting is needed, and Gauss–Laguerre is the quadrature matched to it, its weight function being exactly e^{-s}. The measure is differentiable in the pole parameters, q entering only by rescaling the nodes to s_k/q and the weights to γ_k/q^2 and γ_k/q; a gradient therefore reaches μ itself, so pole locations and widths are learned rather than read off a fixed activation. The per-node coefficients are moreover left free, so the block spans $\{\cos(s_k(t-p)/q), \sin(s_k(t-p)/q)\}$ on the node set, with Eq. equation 4 as the initialization and App. A.3 covering the family.

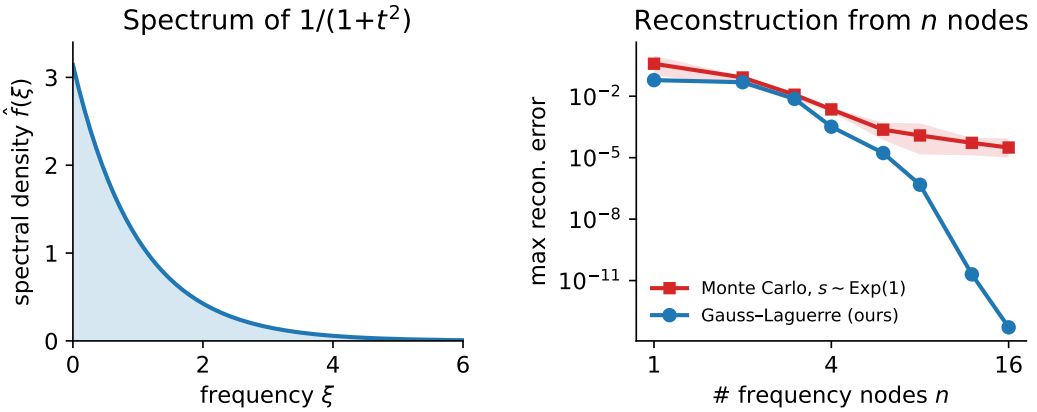

Figure 6: **Left:** the spectral density $\pi e^{-\xi}$ of $1/(1+t^2)$ is continuous, so no finite set of frequencies is exact and the continuum must be discretized. **Right:** max-abs reconstruction error on $|t| \le 1$ from n frequencies, Gauss–Laguerre nodes against $s \sim \mathrm{Exp}(1)$ (mean of 200 draws, shaded 10–90th percentile), both with least-squares-fitted coefficients.

C Practical Considerations and Extended Functionalities of RaLiMP

In this section, we discuss additional properties of the RaLiMP layers and discuss some extensions in context of LLMs and guidance on choosing τ as it controls the variance of the estimator.

C.1 Choosing the Input Scale τ

The rational features of Eq. 7 are evaluated at $\boldsymbol{u}/\tau$ on both sides, so τ acts on the scalar variable as $t \mapsto t/\tau^2$: the block estimates $\cos(s_k t/\tau^2)$ and therefore realizes the *dilated* atom $1/(1+(t/\tau^2)^2)$. A single scale thus controls two quantities at once. It sets the feature exponent $s\|\boldsymbol{x}+\boldsymbol{w}\|^2/\tau^2$ of Sec. 2.3.2, so larger τ means smaller variance; and it sets the effective frequencies s_k/τ^2, so larger τ compresses the Gauss–Laguerre nodes toward the flat center of the atom and wastes them. The useful regime is $t/\tau^2 = O(1)$, which places the nodes on the part of the atom that varies. Concretely we set, *per layer*,

$$\tau^2 \;=\; c\sqrt{\mathbb{E}\big[t^2\big]} \;=\; c\,\mathrm{RMS}(\boldsymbol{w}\cdot\boldsymbol{x}), \tag{33}$$

estimated from a small sample of pre-activations, so that $\mathrm{RMS}(t/\tau^2) = 1/c$ and the constant c directly names the typical argument magnitude. Note that the square is what makes this scale-invariant: taking τ itself proportional to $\mathrm{RMS}(t)$ would leave t/τ^2 *shrinking* as the layer's pre-activation scale grows.

C.2 Deterministic Nodes versus Monte Carlo

The weight e^{-s} in Eq. 5 is a probability density, so the frequency integral equals $\mathbb{E}_{s\sim\mathrm{Exp}(1)}[\cos(st)]$ and could be sampled, as random-feature constructions usually do. Two reasons for fixed Gauss–Laguerre nodes instead. First, accuracy at equal cost: our per-node coefficients are trainable, so the fair comparison grants both node sets least-squares-optimal coefficients and differs only in node placement. Fig. 6 (right) shows the deterministic nodes ahead at every count, by 7.4×10^{-3} against 1.2×10^{-2} at $n_{\mathrm{freq}} = 3$ and two orders of magnitude at $n_{\mathrm{freq}} = 8$; Monte Carlo nodes cluster near $s = 0$, so their near-duplicate features need fitted coefficients of order 10^6 where ours stay below 1. Second, the nodes are known before training, so $s_{\max}$, the discarded weight $\sum_{s_k > s_{\max}} \gamma_k$ and the bound of Cor. 1 are deterministic rather than controlled in expectation, and all randomness in a RaLiMP sits in the projections $\boldsymbol{\omega}$.

C.3 Repeated Conjugate Poles

A conjugate pair of multiplicity n contributes partial-fraction terms with denominators $((t-p)^2+q^2)^j$, $j=1,\dots,n$, which are not of the form of Eq. 14. These require no new feature maps: raising the multiplicity only multiplies the Laplace weight by a polynomial in s. Indeed

$$\int_0^\infty s^k e^{-s} e^{ist}\, ds = \frac{k!}{(1-it)^{k+1}}, \tag{34}$$

whose real and imaginary parts are proper rational functions with denominator $(1+t^2)^{k+1}$. For $n=2$ the four functions obtained from $k=0,1$ have numerators $1+t^2$, $1-t^2$, $t+t^3$ and $2t$ over $(1+t^2)^2$, which span every proper rational function with that denominator; the general case is identical, and the shifted atom follows by the substitution of Sec. 2.4. Consequently weights $\rho_c(s)e^{-s}$ and $\rho_s(s)e^{-s}$ of degree $n-1$ on the cosine and sine channels suffice, and the construction is unchanged apart from replacing the quadrature weights by $\gamma_k\rho_c(s_k)$ and $\gamma_k\rho_s(s_k)$: the kernel e^{is_kt}, the estimators of Eqs. 8–9 and the variance bound of Sec. 2.3.2 all carry over verbatim.

C.4 Compressing Networks by Combining Layers

This is the principal motivation of using kernel methods instead of regular MLP. This was discussed in previous works (Sehanobish et al., 2024; Kim et al., 2025) but we include it for the convenience of the reader.

In several empirical settings, e.g., training NeRF models and uptraining Transformers, we use an RaLiMP layer followed by a linear layer. In this section, we explain how we can achieve improved efficiency by compressing linear layers with RaLiMPs. Recall our RaLiMP layer is given by the following equation (following the PyTorch convention):

$$\mathbf{Y}_1 = \Phi(\mathbf{X})\Psi(\mathbf{W})^\top \tag{35}$$

If the RaLiMP layer is followed by the linear layer with weight $\mathbf{W}'$ and bias $\mathbf{b}'$, then the results is:

$$\begin{aligned}\mathbf{Y}_2 &= \mathbf{Y_1}\mathbf{W'}^\top + \mathbf{b}' \\ &= \Phi(\mathbf{X})\Psi(\mathbf{W})^\top\mathbf{W'}^\top + \mathbf{b}' \\ &= \Phi(\mathbf{X_1})\hat{\mathbf{W}}^\top + \mathbf{b}',\end{aligned}$$

where $\hat{\mathbf{W}} = \mathbf{W}'\Psi(\mathbf{W})$. This simple compression technique allows us to collapse any RaLiMP layer followed by a FFL in our experiments, allowing for more efficient inference.

C.5 Efficient Computations with Hadamard-Product Matrices

Assume that you have a Hadamard-product matrix $\mathbf{M} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$ defined as:

$$\mathbf{M} = \mathbf{Q} \odot \mathbf{R} \tag{36}$$

for dot-product matrices: $\mathbf{Q} = [\mathbf{a}_i^\top \mathbf{b}_j]_{i=1,\dots,N}^{j=1,\dots,N}$ and $\mathbf{R} = [\mathbf{c}_i^\top \mathbf{d}_j]_{i=1,\dots,N}^{j=1,\dots,N}$, and some $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}_i \in \mathbb{R}^d$ for $i = 1, ..., N$. Note that $\mathbf{M}$ can be then re-written as:

$$\mathbf{M} \overset{\mathbb{E}}{=} \mathbf{V}\mathbf{W}^T, \tag{37}$$

for matrices $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{N\times r}$ with rows defined as follows (where $r \ll N$ is a hyperparameter):

$$\begin{aligned}\mathbf{V}[i] &= (\mathbf{G}^1\mathbf{a}_i) \odot (\mathbf{G}^2\mathbf{c}_i) \in \mathbb{R}^r, \\ \mathbf{W}[j] &= (\mathbf{G}^1\mathbf{b}_j) \odot (\mathbf{G}^2\mathbf{d}_j) \in \mathbb{R}^r,\end{aligned} \tag{38}$$

where $\mathbf{G}^1, \mathbf{G}^2 \in \mathbb{R}^{r\times d}$ are independently sampled Gaussian matrices of entries chosen independently from $\mathcal{N}(0,1)$.

Note that Equation 37 provides an unbiased low-rank factorization of the Hadamard-product matrix $\mathbf{M}$, making efficient matrix-vector multiplications with it feasible. Importantly, it does it by avoiding any computations that would require materializing d^2-length vectors.

The above recipe can be trivially extended to non-square matrices.

Implementation of RaLiMP in LLMs LLMs use a *gated* (SwiGLU) feed-forward block, $\mathrm{SiLU}(\mathbf{G}x) \odot (\mathbf{U}x)$ followed by a down-projection $\mathbf{W}_d$, which is exactly the Hadamard-product form of Section C.5 with $\mathbf{Q} = [\mathrm{SiLU}(\mathbf{g}_j^\top \mathbf{x}_i)]$ and $\mathbf{R} = [\mathbf{u}_j^\top \mathbf{x}_i]$ ($\mathbf{g}_j, \mathbf{u}_j$ the rows of $\mathbf{G}, \mathbf{U}$). $\mathbf{R}$ is already a dot-product matrix; the only extra step is to write $\mathbf{Q}$ as one, which we do by linearizing the scalar SiLU with the rational random-feature map of Section 2.1, $\mathrm{SiLU}(\mathbf{g}_j^\top \mathbf{x}_i) \approx \langle \Phi(\mathbf{x}_i), \Psi(\mathbf{g}_j) \rangle$.

Section C.5 then applies verbatim and yields the unbiased rank-r factorization $\mathbf{M} \overset{\mathbb{E}}{=} \mathbf{V}\mathbf{W}^\top$. The weight side $\mathbf{W}_d\mathbf{W}$ depends only on $\mathbf{G}, \mathbf{U}, \mathbf{W}_d$ and folds once, offline, into a single $d \times r$ matrix, so at inference each token costs one feature map plus two $O(rd)$ matrix products and never materializes the wide intermediate.

D Experimental Details

In this section we provide experimental details for all the experiments considered in our work. All experiments on ViT, synthetic and Implicit Neural Representations (INR) are run on 1 24 GB A10 GPU. The CLM and the Qwen experiments are run on 1 GPU inside a 8 x 96 GB Blackwell GPU machine with 2048 GB RAM. The PTv3 experiments are run on 4 x 84 GB NVIDIA H100 GPUs in a machine with approximately 2048 GB of RAM. Each subsection states its data and optimizer and then lists the hyperparameters of the replaced layers following notation of Sec. 2.4. A rational block draws n_{freq} Gauss–Laguerre frequencies (Eq. 5) and keeps those with node $s_k \leq s_{\max}$; where an experiment lists no $s_{\max}$, all n_{freq} frequencies are kept. The nodes spread further from the origin as n_{freq} grows, so a fixed $s_{\max}$ retains only the lowest-frequency ones: raising n_{freq} refines the quadrature inside $[0, s_{\max}]$ rather than reaching higher frequencies. If s_{max} is not mentioned then the given experiment keeps all n_{freq} nodes. On the pretrained backbones the input scale is $\tau = c\,\mathrm{RMS}(\boldsymbol{w} \cdot \boldsymbol{x})$ (Eq. 33) with $c = 4$ unless stated otherwise.

D.1 Synthetic Experiments

The synthetic experiment benchmark fits one layer $\boldsymbol{Y} = f(\boldsymbol{X}\boldsymbol{W}^\top)$ with a fixed $\boldsymbol{W} \in \mathbb{R}^{256\times256}$ and $\boldsymbol{X} \in \mathbb{R}^{8192\times256}$, every entry i.i.d. $\mathrm{Unif}[0, 1]$. We take $f \in \{\mathrm{GELU}, \mathrm{SiLU}, \tanh, \mathrm{sigmoid}, \mathrm{softsign}\}$ and train for 10 seeds using Adam optimizer. The models are trained for 3000 steps with learning rate 10^{-2} and cosine decay along with clipping at 10. We report the fitting error as train MSE loss. All four methods linear map, the EUGen, SNNK, and ours share the feature budget of 32 random features/rank, RaLimP spends it on the polynomial of degree $K = 2$ with 8 random features and the rest of the 24 features in the rational block is split among $n_{\mathrm{freq}} = 3$ frequencies with $m = 4$ features each, one centered summand of Eq. 4, and $\tau = 1$. The companion panel sweeps the budget over various number of random features for our layer.

D.2 ViT Experiments

We use ViT-B/16 pretrained on ImageNet-21k (`google/vit-base-patch16-224-in21k`): 12 layers, dimension =768, MLP width 3072 and GELU non-linearity. We replace the MLP of the last k blocks (i.e. blocks $12 - k, \ldots, 11$) and fine-tune the entire network with cross-entropy, AdamW, a one-cycle schedule with 10% linear warmup, gradient-norm clipping at 1.0, fp32, and seed 0. We test across $k = 4, 7, 9, 11$. We run two regimes: full-dataset fine-tuning on CIFAR-100, Places365 and ImageNet-1k, and VTAB-1k (Zhai et al., 2020) transfer over its 19 tasks (each 1000 training images, the official 800+200 split used at first for hyperparameter tuning). The reported numbers are from the full test set.

The Visual Task Adaptation Benchmark measures how well a pretrained vision model transfers to new image-classification tasks under a tight data budget. It comprises 19 tasks grouped into three categories: *Natural* (everyday photographic images, e.g. pets, flowers), *Specialized* (domain images such as medical and satellite imagery), and *Structured* (tasks requiring geometric reasoning, e.g. counting objects or estimating distances). The "1k" denotes the protocol: each task provides only 1,000 labeled training examples, with evaluation on the full test set. This small training budget is what makes the benchmark demanding as there is little data from which to recover the accuracy lost when introducing our replacement layers.

Table 4 gives the per-dataset settings; for VTAB-1k, Table 5 lists the values swept around the default recipe and each method's reported accuracy is its best over the sweep.

Table 4: ViT-B/16 fine-tuning settings. VTAB-1k uses random-resized crop only when the source resolution exceeds 100 pixels.

Dataset	train / eval	epochs	batch	lr / wd	augmentation
CIFAR-100	50k / 10k	12–20	256	5×10^{-5} / 0.05	resize, h-flip
Places365	1.71M / 90k	15	128	5×10^{-5} / 0.05	random-resized crop, h-flip
ImageNet-1k	1.28M / 50k	25–35	128–256	5×10^{-5} / 0.05	random-resized crop, h-flip
VTAB-1k (19 tasks)	1k / full test	100	64	2.5×10^{-4} / 10^{-3}	resize, random-resized crop, h-flip

Layer hyperparameters. The random-feature projections are frozen at their random initialization in all experiments. Coefficients are initialized by a ridge-regularized least-squares fit of GELU on a per-layer range set from the pretrained pre-activations (2000 points), with the quadratic term fixed to EUGen's and the coefficient A fit to the residual. The per-layer scale is $\tau = 4\,\mathrm{RMS}(\boldsymbol{w}\cdot\boldsymbol{x})$, measured once from a minibatch of pretrained pre-activations and then held fixed. This value of τ for some datasets (in VTAB) can be large so we additionally clamp this value to a fixed number. On VTAB-1k the rational branch is introduced gradually, it is scaled by a factor that increases linearly from 0 to 1 over the first 30% of steps. We use 64 random features for the larger image classification experiments and 32 random features for VTAB. RaLiMP splits its budget into 48, resp. 24 (for VTAB) features for the polynomial part and the rational block with one centered atom using 16 resp. 8 (for VTAB) random features, $n_{\mathrm{freq}} = 3$, and $s_{\max} = 4$.

Table 5: Hyperparameters swept on VTAB-1k, around the default recipe of Table 4. Deep layers of a pretrained ViT have much larger pre-activations than early ones, so capping τ is the one lever specific to our layer.

Hyperparameter	Values
learning rate	1.9, 2.1, 2.5 ($\times10^{-4}$); up to 10^{-2} on the hardest Structured tasks
weight decay	10^{-3}, 5×10^{-2}
scale multiplier c in τ	4, 8, 10, 12
cap on τ	none, 10, 12, 16, 20, 21
label smoothing / dropout	0, 0.05, 0.1, 0.15 / 0, 0.1

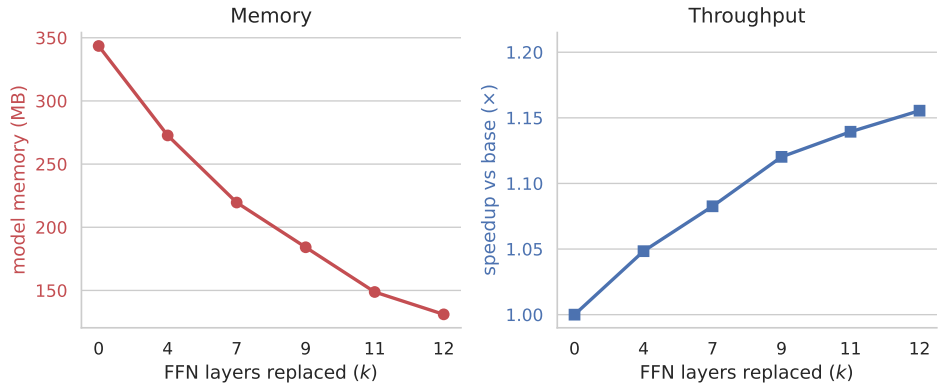

Figure 7: RaLiMPs in ViT. Replacing more layers by RaLiMPs reduce memory overhead and speed up inference.

D.3 Causal Language Modeling

We use `https://github.com/karpathy/nanogpt` implementation of GPT-2. For our experiment, GPT-2 Large (36 blocks, $d = 1280$, 20 heads, context 1024) is trained from scratch on

OpenWebText with the GPT-2 tokenizer and 10M held-out tokens for validation. We replace the MLP of the last k blocks and run each experiment identically: AdamW ($\beta = (0.9, 0.95)$, weight decay 0.1), lr $6{\times}10^{-4}$ decaying by cosine to $6{\times}10^{-5}$ after 500 warmup iterations, batch 24×30 accumulated steps of 1024 tokens (737k tokens per iteration), bf16, and seed 1337. Validation perplexity is reported at a fixed budget of 30k iterations ($\approx$22B tokens, generally following the Chinchilla scaling law (Hoffmann et al., 2022)), averaged over 200 batches. Both EUGen and RaLiMP use degree 2 polynomials. Both SNNK and EUGen uses 128 random features, while RaLiMP uses 128 random features split between the polynomial block with 120 features and rational block with $n_{\text{freq}} = 2$ frequencies and $m = 4$ projections each with trainable A, B, p, q of Eq. 14 (i.e. 8 random features), and $\tau = 3$.

D.4 PTv3 Experiments

We evaluate RaLiMP on three indoor and outdoor 3D semantic segmentation benchmarks:

- **ScanNet v2** (Dai et al., 2017): A large-scale indoor RGB-D dataset with 20 semantic categories. We use the standard split of 1,201 training and 312 validation scans.
- **ScanNet200** (Rozenberszki et al., 2022): An extension of ScanNet v2 with 200 fine-grained categories, designed to evaluate robustness under long-tailed class distributions.
- **nuScenes** (Caesar et al., 2020): A large-scale outdoor autonomous driving dataset with 1,000 urban scenes. We use it to evaluate RaLiMP on sparse LiDAR point clouds across 16 classes.

We use the official PTv3 implementation provided by Pointcept (Pointcept Contributors, 2023) and replace selected MLP modules with RaLiMP. The backbone uses serialized softmax attention with FlashAttention, and the patch sizes of both the encoder and decoder are set to 1024. The serialization patterns Z, TZ, H, and TH denote Z-order, transposed Z-order, Hilbert order, and transposed Hilbert order, respectively. We compare EUGen, SNNK, and RaLiMP using the same PTv3 backbone and MLP replacement configurations. For each $k \in \{5, 10, 22\}$, all three methods replace the MLP modules in the same Transformer blocks, ensuring that both the number and locations of replaced modules are matched across methods. The backbone and RaLiMP configurations, together with the MLP replacement allocations, are summarized in Tab. 7.

We report mean intersection over union (mIoU), mean class accuracy (mAcc), and overall accuracy (allAcc). mIoU and mAcc average the intersection over union and classification accuracy across semantic classes, respectively, while allAcc measures the fraction of correctly classified points. The results for all three metrics are reported in Tab. 6.

Table 6: Semantic segmentation performance on indoor (ScanNet and ScanNet200) and outdoor (nuScenes) benchmarks. k denotes the number of replaced MLP modules in PTv3.

k	Methods	ScanNet Val			ScanNet200 Val			nuScenes Val		
		mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
5	EUGen	75.65	83.98	91.54	34.62	45.47	83.37	79.14	86.36	94.56
	SNNK	76.84	84.33	91.89	35.65	46.37	83.78	80.76	87.38	94.79
	RaLiMP (ours)	78.08	85.43	92.37	36.27	46.54	83.65	80.81	87.20	94.77
10	EUGen	74.50	82.77	90.87	33.08	43.65	82.76	76.34	85.56	94.38
	SNNK	76.63	84.19	91.69	34.69	44.33	83.40	79.92	86.82	94.68
	RaLiMP (ours)	77.54	85.24	92.09	35.50	45.36	83.24	80.65	87.19	94.75
22	EUGen	72.80	81.49	90.22	30.56	41.96	81.16	74.84	84.11	94.21
	SNNK	76.35	84.15	91.66	34.01	43.99	83.26	79.79	87.40	94.64
	RaLiMP (ours)	77.33	84.89	91.93	34.90	45.62	83.41	80.47	87.13	94.65
Baseline (dense, full FT)		77.57	84.97	92.04	35.26	46.05	83.38	80.32	87.25	94.63

Table 7: PTv3 backbone and RaLiMP configurations for semantic segmentation. C denotes the channel dimension of the corresponding block. τ_0 and $\lambda_{\mathrm{gate},0}$ denote the initial values of the learnable rational input scale and gate, respectively. k denotes the number of replaced MLP modules, with replacement starting from the top MLP modules within each selected stage.

Configuration	**Value**
PTv3 backbone	
Attention	Serialized softmax attention (FlashAttention)
Serialization patterns	Z + TZ + H + TH
Patch interaction	Shift Order + Shuffle Order
Positional encoding	xCPE
Encoder depths	$[2, 2, 2, 6, 2]$
Encoder channels	$[32, 64, 128, 256, 512]$
Encoder heads	$[2, 4, 8, 16, 32]$
Encoder patch sizes	$[1024, 1024, 1024, 1024, 1024]$
Decoder depths	$[2, 2, 2, 2]$
Decoder channels	$[64, 64, 128, 256]$
Decoder heads	$[4, 4, 8, 16]$
Decoder patch sizes	$[1024, 1024, 1024, 1024]$
Downsampling strides	$[\times 2, \times 2, \times 2, \times 2]$
MLP ratio	4
QKV bias	True
Pre-normalization	True
Drop path rate	0.3
Backbone output channels	64
RaLiMP configuration	
Random feature dimension	8
Random projections (base and rational blocks)	Frozen Gaussian
Rational atom type	General shifted
Number of rational atoms J	2
Rational quadrature frequencies n_{freq} ($C < 256$ / $C \geq 256$)	$2/3$
Random projections per frequency m ($C < 256$ / $C \geq 256$)	$4/8$
Rational input normalization	Disabled
Readout weights and rational coefficients	Learnable
Rational input scale initialization τ_0	3.0
Gate initialization per block $\lambda_{\mathrm{gate},0}$	$\{0.5, 1.0, 2.0\}$
MLP replacement configurations	
Allocation for $k = 5$	Enc. $[0, 0, 0, 3, 1]$, Dec. $[0, 0, 0, 1]$
Allocation for $k = 10$	Enc. $[0, 0, 0, 6, 2]$, Dec. $[0, 0, 0, 2]$
Allocation for $k = 22$	Enc. $[2, 2, 2, 6, 2]$, Dec. $[2, 2, 2, 2]$

D.5 COORDINATE NETWORKS

Next we discuss the experimental details for NeRF, SIREN and PINN networks.

D.5.1 SIREN

Image fitting on Cameraman (256×256) with 3 hidden layers of width 256 and $\omega_0 = 30$, trained by Adam on the full-image MSE for 500 steps, seed 0, each arm's learning rate tuned over 10^{-4} to 10^{-2}; we report $\mathrm{PSNR} = 10 \log_{10}(1/\mathrm{MSE})$. The first sine layer is kept for all the methods and all of hidden layers are replaced by linear variants. A periodic target uses $\cos / \sin$ pair at $n_{freq} = 1$, $\tau = 1$. Ours uses 128 projections, hence 256 features $[\cos, \sin]$ and 99k deployed parameters against 198k for SIREN. Both SNNK and EUGen use 256 random features to match the same inference parameters as RaLiMP.

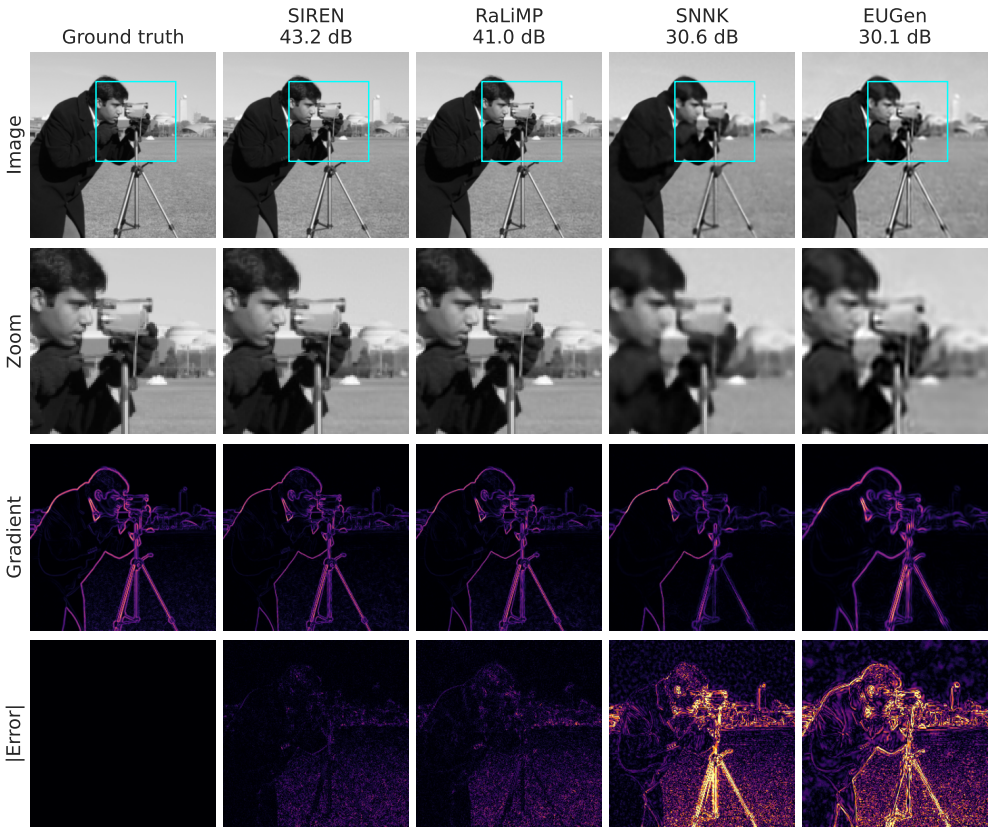

Figure 8: **SIREN Networks.** RaLiMP matches the dense baseline while replacing all layers and outperform all variants.

D.5.2 NERF

On the 8 synthetic Blender scenes at half resolution we follow the EUGen protocol in replacing the feed-forward layers with our layers including : depth 8, width 256, positional encoding of order 10 (and 4 for view directions), 1024 rays per step, 64 coarse and 128 fine samples, Adam decaying by a factor 10 every 500k steps, for 200k iterations, per-scene learning rate 5×10^{-4} for chair, drums, lego, materials and mic and 10^{-4} for ficus, hotdog and ship. We report the mean PSNR over the test views. All the linear methods use matched 256 random features. Moreover RaLiMP uses $n_{freq} = 1, \tau = 1$.

D.5.3 PHYSICS-INFORMED NETWORK

The viscous Burgers equation $u_t + uu_x = \nu u_{xx}$ with $\nu = 0.01/\pi$ on $x \in [-1, 1]$, $t \in [0, 1]$, $u(0, x) = -\sin(\pi x)$ and $u(t, \pm 1) = 0$. The network is 4 hidden layers of width 128 with $\tanh$ activations, trained by Adam (lr 10^{-3}, cosine decay) for 20k steps on 10k collocation points and 200 initial/boundary points with an unweighted sum of the residual, initial and boundary losses; seed 0. We report relative L_2 error against a finite-difference reference on a 101×256 grid. We replace all hidden layers by our variants and the replaced layers are constrained to fewer deployed parameters than the $\tanh$ network (66.6k): ours 38.5k ($K = 3$, 8 polynomial projections, $n_{freq} = 3$ nodes, 4 projections per node, $\tau = 3$, $s_{max} = 4$), EUGen 38.5k (12 projections per degree) and SNNK 38.4k (36 random features).

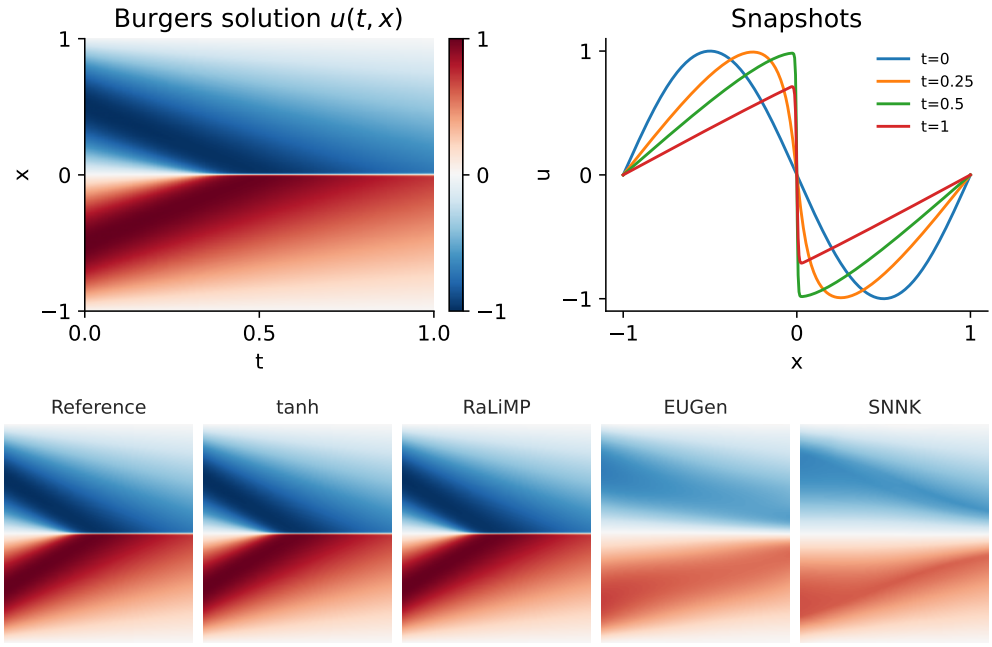

Figure 9: **Top Row : Reference solution of the viscous Burgers' equation.** *Left:* the ground-truth field $u(t, x)$ over $(t, x) \in [0, 1] \times [-1, 1]$. *Right:* snapshots at $t \in \{0, 0.25, 0.5, 1\}$, showing the initial sinusoidal profile steepening into a sharp shock at x=0. **Bottom Row : Predicted Burgers' fields.** Solution $u(t, x)$ recovered by each model against the reference (leftmost). The dense tanh network and RaLiMP sharply reproduce the shock at x=0, whereas EUGen and SNNK smear it and lose the discontinuity, at matched inference budget. Tanh refer to the baseline PiNN model with tanh activation.

D.6 QWEN3-8B EXPERIMENTAL DETAILS

In this section we describe in detail the Qwen experiments.

D.6.1 DATASETS

First we list all the datasets that are used for this experiment.

Training.

- **OpenThoughts3-1.2M** (Guha et al., 2026) : 1.2M long chain-of-thought reasoning traces; used for the supervised fine-tuning of the replacement layers (Section D.6.2).
- **DeepScaleR** (Luo et al., 2025) : 40,027 competition-mathematics problems with verifiable answers; the prompt pool from which the compressed model generates its own solutions

for the Stage-2 on-policy rollouts (Section D.6.3). After sampling and filtering to correct, self-terminating solutions, the retained Stage-3 distillation set contains $87{,}000$ (problem, solution) pairs for the $k = 8$ ($20k$ for the $k = 4$ model). The data consists of math problems with short prompts (median 63 tokens, 10th/90th percentiles 30/129) and long chain-of-thought solutions (median $6{,}538$ tokens, 10th/90th percentiles $2{,}271$/$13{,}042$), of which 0.9% reach the $16{,}384$-token cap. We additionally pool a *science-focused* data with the mathematics pool above. The science prompts come from **WebInstruct-verified** (Yue et al., 2024), restricted to the questions tagged physics, chemistry, or biology whose answers are numeric or short symbolic expressions (answer types *Float*, *Integer*, or *Expression*); in practice these are chiefly physics and chemistry. This yields $2{,}287$ traces over $2{,}256$ distinct problems, with similarly short prompts (median 58 tokens) but somewhat shorter solutions (median $4{,}098$ tokens, 10th/90th percentiles $1{,}522$/$7{,}478$).

Evaluation. (n denotes the number of problems.)

- **AMC23** (n=83) (AI-MO, 2024) : problems from the 2023 American Mathematics Competitions (AMC 10/12); avg@16 and pass@16.[1]
- **AIME24** (n=30) (Jia, 2024) : the 2024 American Invitational Mathematics Examination; avg@16 and pass@16.
- **AIME25** (n=30) (Zhang & Math-AI, 2025) : the 2025 American Invitational Mathematics Examination; avg@16 and pass@16.
- **GSM8K** (n=1,319 test) (Cobbe et al., 2021) : grade-school arithmetic word problems.
- **MATH500** (n=500) (Hendrycks et al., 2021b) : a 500-problem subset of the MATH competition-mathematics benchmark (algebra, geometry, number theory, precalculus).
- **Minerva Math** (n=272) (Lewkowycz et al., 2022) : undergraduate-level quantitative problems in physics, chemistry, and related sciences.
- **OlympiadBench** (n=674) (He et al., 2024) : olympiad-level mathematics and physics problems.
- **GPQA-Diamond** (n=198) (Rein et al., 2024) : graduate-level, multiple-choice questions in biology, physics, and chemistry, avg@16 and pass@16.
- **MMLU** (n=14,042 test) (Hendrycks et al., 2021a) : multiple-choice questions across 57 subjects spanning STEM, humanities, and social sciences.

We compress Qwen3-8B by replacing the feed-forward network in the top k transformer blocks with our random-feature layer, and then recover the lost accuracy in two training stages: (i) a supervised fine-tuning stage that fits the replacement layers on a large reasoning corpus, and (ii) an on-policy distillation stage in which the compressed model learns to reproduce the original model's next-token distribution on the compressed model's own generated solutions. We run the identical pipeline at two compression levels, $k = 8$ and $k = 4$; the only differences between them are which blocks are replaced (Table 8). Everything else, i.e. optimizer, schedule, and the two-stage procedure is shared. The k = 4 model requires less data in both stages to recover its performance.

Qwen3-8B has 36 transformer blocks. For $k = 8$ we replace the feed-forward network in blocks 28–35 (the last eight); for $k = 4$ we replace blocks 32–35 (the last four). In both cases the attention layers, the embedding, and the output head are left unchanged, and the original feed-forward weights of the replaced blocks are discarded. During all training the original (non-replaced) weights are kept frozen and adapted only through low-rank adapters (LoRA) (Hu et al., 2022a); the replacement layers themselves are trained fully.

The replacement layer is exactly the construction of Section C.5, applied to the gated (SwiGLU) feed-forward network $\mathrm{SiLU}(\mathbf{W}\mathbf{x}) \odot (\mathbf{U}\mathbf{x})$: the SiLU gate is expressed through the random-feature map (linear and squared-dot terms plus the rational term), and the gated product is collapsed to a rank-r factorization. Table 8 lists all the settings used in our experiments.

The feature map has width 672: a linear group of 192; a squared-dot group of 192 (two projections $\mathbf{G}, \mathbf{H}$ combined by an element-wise product, which does not double the width); and a rational group

[1] We use the 83-problem `AI-MO/aimo-validation-amc` release.

of $2 \times 3 \times 48 = 288$ (3 retained quadrature frequencies, 48 features each, a cosine and sine tower per frequency).

We use a single pole $1/(1+t^2)$, with $\tau = 1$, and normalize the pre-activation onto the unit sphere before the rational projection, and as a further safeguard the exponent of the rational random features (the argument of the $e^{(\cdot)}$ factor in Eq. equation 7) is clamped from above at 10. The gated product is collapsed to Hadamard rank $r = 512$ for $k = 8$; for $k = 4$ we report both $r = 256$ and $r = 512$. The SiLU coefficients are held at their fixed values, and all random projections together with the rational weight and node coefficients are trained.

Table 8: Architecture of the replaced feed-forward layers. Both settings replace the top k blocks of Qwen3-8B and share all other hyperparameters.

	$k = 8$	$k = 4$
Blocks replaced	28–35	32–35
Linear (random-projection) group dimension	192	192
Quadratic (EUGen squared-dot) group dimension	192	192
Rational quadrature frequencies	6	6
Rational random features per frequency	48	48
Rational group width ($2 \times 3 \times 48$)	288	288
Total feature width	672	672
Linear coefficient (fixed)	0.5	0.5
Quadratic coefficient (fixed)	0.25	0.25
Rational weight (learnable, init.)	0.0	0.0
τ	1	1
Cap on rational-feature exponent (Eq. 7)	10.0	10.0
Hadamard rank r	512	512

D.6.2 Stage 1: supervised fine-tuning of the replacement layers

We first fit the freshly-initialized replacement layers to the base model's behavior by supervised fine-tuning on OpenThoughts3-1.2M (Guha et al., 2026). We stream the corpus shard by shard and tokenize on the fly, so that so no example repeats within a pass, with a fixed seed for the shard order and sampling. Each training example is a (problem, solution) pair; the loss is the standard next-token cross-entropy on the solution tokens. We choose the best checkpoint based on validation accuracy on a held out portion of OpenThoughts to mimic the distribution of the downstream tasks. Note that fine-tuning base Qwen model on this dataset using LoRA does not improve the base model.

Hyperparameters.

- Optimizer: AdamW, learning rate 1×10^{-4}, cosine decay to 10% of the peak, 100 warmup steps.
- Weight decay: 0.
- Sequence length: 16,384 tokens; examples longer than this are dropped so that only complete (end-of-sequence terminated) traces are trained on.
- Batch size: 1 sequence per step, with gradient accumulation over 16 micro-batches (effective batch size 16 sequences).
- Number of optimization steps: 64,000 for k = 8 and 40,000 for k = 4.
- Frozen backbone with low-rank adapters on all linear layers (attention and feed-forward projections): LoRA rank 32, LoRA scaling factor 64. The replacement layers are trained fully.
- Attention implementation: scaled dot-product attention. Activation checkpointing enabled.
- Precision: bfloat16.

This stage produces the compressed model we refer to as the *pre-distillation* model in the main text.

D.6.3 STAGE 2: SAMPLING THE COMPRESSED MODEL'S OWN SOLUTIONS

On-policy distillation requires solutions drawn from the *compressed* model itself so that the teacher is later scored on states the compressed model actually visits. We therefore generate a pool of solutions from the Stage-1 model on Competition-mathematics problems from DeepScaleR. Each problem is presented with the instruction "Please reason step by step, and put your final answer within `\boxed{}`." For each problem we draw 8 independent samples with temperature 0.8, nucleus sampling $p = 0.95$, top-k = 20, and a maximum generation length of 16,384 tokens.

Filtering. We keep only solutions that (i) terminate on their own (emit the end-of-sequence token before the length cap) and (ii) are correct, verified by extracting the final `\boxed{}` answer and checking it against the ground-truth answer. The kept (problem, solution) pairs form the on-policy distillation set. This pool contains roughly 87,000 solutions from math datasets and about 2700 from science. Notably for the k = 4 model, we use about 18,000 traces from the math dataset as that model was already close to base after fine-tuning.

D.6.4 STAGE 3: ON-POLICY DISTILLATION

Starting from the Stage-1 compressed model, we distill the base Qwen3-8B into the compressed model on the solutions collected in Stage 2. Both the compressed ("student") model and the frozen original ("teacher") model process the same solution; the training loss is the forward KL divergence $\mathrm{KL}(p_{\text{teacher}} \,\|\, p_{\text{student}})$ between the teacher's and student's raw next-token distributions (temperature = 1), averaged over the solution tokens. It is on-policy because these solutions are the student's own Stage-2 generations, so the teacher is scored on the states the student actually visits.

Hyperparameters.

- Initialization: the Stage-1 compressed model (Section D.6.2).
- Distillation data: the on-policy solution pool from Stage 2 (Section D.6.3).
- Optimizer: AdamW, learning rate 2×10^{-5}, cosine decay to 10% of the peak, 10 warmup steps.
- Weight decay: 0.
- Forward KL divergence as the only loss term (weight 1.0).
- The forward KL is computed over the completion tokens in chunks of 2,048 tokens to bound memory, then summed over chunks and averaged over tokens.
- Sequence length: 12,288 tokens.
- Batch size: 1 sequence per step, with gradient accumulation over 24 micro-batches (effective batch size 24 sequences).
- Frozen backbone with low-rank adapters on all linear layers: LoRA rank 32, LoRA scaling factor 64. The replacement layers are trained fully.
- Attention implementation: scaled dot-product attention. Activation checkpointing enabled.
- Precision: bfloat16. The frozen teacher is held on a separate device from the student.
- Number of optimization steps: the distillation set is trained for two epochs. We select the final checkpoint by validation accuracy on a small 150 sample in Numina-Math train (Li et al., 2024).

The $k = 4$ model is produced by the identical Stage-3 procedure, initialized from the $k = 4$ Stage-1 model, with the only architectural differences being those in Table 8.

D.6.5 EVALUATION PROTOCOL

All models are evaluated with temperature 0.6, nucleus sampling $p = 0.95$, top-k 20, and a maximum generation length of 16,384 tokens, in the model's default reasoning ("thinking") mode.

Since AMC, AIME and GPQA are small datasets, we therefore draw 16 samples per problem and report the mean per-problem accuracy (the average over the 16 samples), which we denote avg@16.

We also report pass@16 (a problem counts as solved if any of the 16 samples is correct) as a capability ceiling. For the larger benchmarks (GSM8K, MATH500, Minerva, OlympiadBench) we report a single sample per problem at the same decoding settings.

Mathematical answers are extracted from the last `\boxed{}` in the generation and scored against the ground truth by `math_verify` for symbolic/numeric equivalence, with a boxed-string match and a numeric-tolerance check as fallbacks when parsing fails; the same check gates the Stage-2 filter. The single-sample benchmarks decode with a fixed seed, while the 16 avg@16 draws are independent. The multiple-choice answers are extracted as the boxed letter. MMLU is evaluated via log probabilities via (Gao et al., 2024).

D.6.6 Effect of the distillation stage

Table 9 isolates Stage 3 (on-policy distillation) at $k = 8$, with benchmarks ordered by base generation length L. The Stage-1 supervised model loses accuracy that grows with L. The length-compounding pattern of Eq. 39 and distillation recovers most of it, with the largest gains on the longest-generation benchmarks. After it, GSM8K and MATH500 exceed the uncompressed model and Minerva and OlympiadBench are within a point.

Table 9: Effect of the on-policy distillation stage (accuracy, %), benchmarks ordered by base median generation length L (tokens). For each compression level, "pre" is the Stage-1 supervised model and "+dist" is the same model after Stage 3; "Base" is uncompressed Qwen3-8B. AMC/AIME are avg@16, the rest single-sample; maximum generation length 16,384 tokens.

		k=4		k=8		
Benchmark	L	pre	+dist	pre	+dist	Base
GSM8K	1,685	94.5	95.5	92.3	95.2	93.5
MATH500	3,528	87.4	92.8	85.2	94.5	90.0
Minerva	4,892	35.3	45.3	28.7	43.4	45.6
AMC23	7,820	75.2	82.2	71.8	78.8	83.0
OlympiadBench	8,244	65.4	70.4	63.8	69.0	69.9
AIME24	12,298	51.0	61.0	42.3	58.1	67.5
GPQA	12,803	41.4	52.4	28.8	51.1	56.1
AIME25	15,942	43.3	52.3	34.2	48.6	57.7

D.6.7 Length-compounding of the kernel approximation error

A finite-feature kernel block incurs a bounded per-token approximation error that perturbs the output logits once per decoded token, so its effect on end-to-end accuracy compounds with the generation length L. Modeling this as a constant per-token derailment rate δ gives a multiplicative law

$$\text{acc} \;\approx\; \text{acc}_{\text{base}} \exp(-\delta L), \qquad \text{i.e.} \quad \text{deficit} \;=\; 1 - \frac{\text{acc}}{\text{acc}_{\text{base}}} \;\approx\; 1 - e^{-\delta L}. \tag{39}$$

Figure 10 evaluates this for the pre-distillation model of Table 9. The deficit is monotone in L, from 3% at $L \approx 1.5$k (GSM8K) to 41% at $L \approx 16$k (AIME25). The implied rate δ is stable at $\sim 2\text{–}4 \times 10^{-5}$ per token wherever the base model is unsaturated where the δ is dominated by ceiling effects. Two orderings are robust: δ grows with the swapped-layer count ($\delta_{k=8} > \delta_{k=4}$) and shrinks with feature rank at fixed k (r=512 $<$ r=256), so δ tracks the kernel's approximation error rather than layer count alone. Long-chain benchmarks are thus *predictably* harder for a fixed kernel budget, and the distillation stage above preferentially removes this length-dependent error.

D.6.8 Effect of the Hadamard rank r

Table 10 compares the two k=4 variants, r=256 and r=512. Raising the rank lowers the kernel's factorization error and improves the competition-math benchmarks. The main-body table reports the r=512 variant.

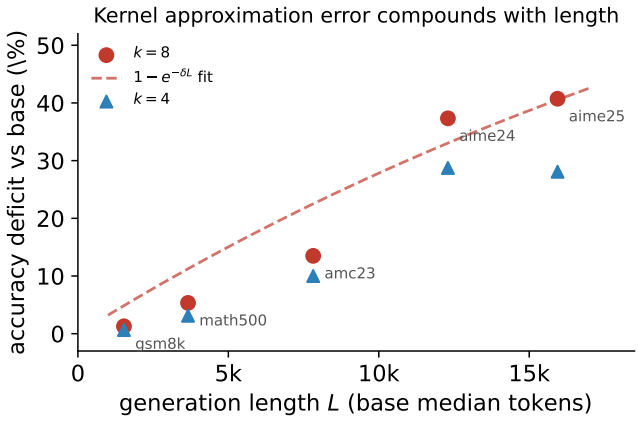

Figure 10: **Accuracy deficit vs. generation length L at matched decoding.** Points are the benchmarks; the dashed curve is the one-parameter fit $1 - e^{-\delta L}$ (δ=3.2×10^{-5}, fit on the unsaturated benchmarks). Swapping more layers (k=8 vs. k=4) raises the deficit at fixed L.

Table 10: k=4 at Hadamard rank r=256 vs. r=512 (accuracy after stage 1 training, %; AMC/AIME are avg@16).

r	AMC23	AIME24	AIME25	GSM8K	MATH500	Minerva	GPQA
256	74.7	48.1	41.5	92.9	87.2	37.9	44.9
512	75.2	51.0	43.3	94.5	87.4	35.3	41.4

D.6.9 Memory footprint and throughput

Because the replacement layer folds its weight side into a single $[d \times r]$ matrix at inference, each replaced feed-forward network is far smaller than the original gated network, and the saving grows linearly with the number of blocks replaced. Table 11 reports the parameter reduction relative to the full 8.19B-parameter model (the feed-forward networks account for about two-thirds of the model's parameters). At $k = 8$ the replacement removes 1.17B parameters (14% of the whole model, 21% of the feed-forward parameters); at $k = 4$ it removes 0.59B (7% of the model).

Table 11: Efficiency of replacing the top k feed-forward networks of Qwen3-8B (36 blocks, 8.19B parameters). Parameter reduction is relative to the full model / to the feed-forward parameters (which are ∼two-thirds of the model). Throughput is compiled decoding (bfloat16; 4096-token prompt, 1024 generated tokens, 16 concurrent sequences, single GPU).

Model	Params removed	% of model	% of FFN	Throughput (tok/s)	Speedup vs. base
Uncompressed	—	—	—	763.3	—
$k = 4$	0.59B	7.2%	10.8%	785.6	+2.9%
$k = 8$	1.17B	14.3%	21.5%	809.6	+6.1%

E Detailed Results

In this section we present detailed results for NeRF, ViT, PTv3 and Causal Language Modeling.

Table 12: Synthetic-NeRF held-out test PSNR (dB). RaLiMP beats all other methods, at *matched deployed parameters* and even beating the dense NeRF baseline.

Method	Chair	Drums	Ficus	Hotdog	Lego	Mater.	Mic	Ship	Avg
NeRF	33.70	25.46	26.40	35.81	31.31	29.00	33.11	28.79	30.45
EUGen	33.09	25.18	28.40	35.47	30.60	28.88	32.63	28.66	30.36
SNNK	30.66	24.76	28.21	35.72	30.00	28.64	30.44	28.41	29.61
RaLiMP (ours)	33.19	25.26	28.89	35.73	30.95	29.02	32.86	28.95	30.61

Table 13: ViT results on CIFAR-100, ImageNet, and Places365 (Top-1 Accuracy). RaLiMPs beat all other methods at *matched deployed inference budget*. Baseline is the dense ViT-B/16 full fine-tune.

Dataset	Method	k (layers replaced) 4	7	9	11
CIFAR-100	Low Rank	86.6	78.3	74.3	69.3
	SNNK	87.3	84.0	80.3	75.9
	EUGen	88.6	86.5	81.3	74.6
	RaLiMP (ours)	**89.8**	**88.1**	**84.6**	**80.4**
	Baseline		92.0		
Places365	Low Rank	54.1	52.8	49.7	45.9
	SNNK	57.1	56.5	56.3	54.6
	EUGen	56.9	56.9	56.1	54.2
	RaLiMP (ours)	**58.0**	**57.5**	**56.9**	**55.2**
	Baseline		58.2		
ImageNet-1k	Low Rank	78.6	72.4	71.0	62.8
	SNNK	80.4	77.4	75.0	73.1
	EUGen	79.1	77.2	75.8	72.8
	RaLiMP (ours)	**80.8**	**78.5**	**76.8**	**73.9**
	Baseline		81.9		

Table 14: VTAB-1k top-1 accuracy (%), matched inference budget, per k (layers replaced). **R**aLiMP (ours), **S**NNK, **E**UGen, **L**owrank. **Bold** = best method at each (task, k).

	k=4				k=7				k=9				k=11			
Task	R	S	E	L	R	S	E	L	R	S	E	L	R	S	E	L
Natural																
CIFAR-100	**58.2**	52.9	38.2	4.8	**57.5**	43.3	26.7	3.3	**51.0**	36.0	13.8	2.5	**40.3**	29.1	3.6	2.8
Caltech101	**85.2**	82.1	74.3	25.6	**84.4**	79.0	63.6	4.2	**76.0**	75.9	49.1	9.0	**69.4**	69.2	7.0	3.8
DTD	**69.0**	66.2	59.9	18.2	**67.9**	63.5	53.4	8.5	**59.1**	59.0	44.2	10.5	**50.0**	49.7	22.8	9.7
Flowers102	**95.6**	95.4	90.3	10.2	**94.3**	94.0	76.9	6.0	**92.3**	91.8	73.6	4.3	85.2	**86.1**	31.2	6.7
Pets	**85.7**	83.8	78.2	22.0	**85.0**	79.7	65.3	10.7	**75.9**	74.3	52.4	5.2	**66.7**	64.0	2.8	8.7
SVHN	**89.3**	88.0	87.3	66.7	**89.2**	88.2	84.9	68.0	**89.7**	89.5	31.4	22.1	**85.8**	85.7	19.6	34.5
Sun397	**46.5**	35.2	22.4	4.5	**44.8**	28.9	15.3	3.0	**35.2**	24.7	10.6	3.4	**30.1**	20.0	3.8	2.5
Specialized																
PatchCamelyon	**88.7**	87.5	87.1	83.3	**88.5**	87.1	87.4	82.1	**88.4**	88.2	82.8	84.7	**86.0**	85.8	78.6	77.1
EuroSAT	**95.3**	94.7	94.5	91.7	**95.7**	95.4	94.3	71.7	**95.6**	95.4	91.6	48.9	**93.4**	93.4	82.1	70.6
Resisc45	**85.8**	84.3	81.5	16.5	**85.2**	84.8	78.2	8.2	**84.0**	82.6	72.2	17.8	**75.6**	75.3	37.3	19.6
Retinopathy	**78.2**	76.7	76.0	76.9	**76.8**	76.5	76.6	76.0	**77.6**	76.7	73.8	76.3	**75.5**	75.3	73.6	73.6
Structured																
Clevr-count	**67.8**	51.8	56.0	66.5	**70.2**	56.9	63.9	68.5	**68.3**	53.4	67.2	58.6	**57.8**	53.2	53.9	48.7
Clevr-dist	54.7	52.0	54.1	**55.1**	**58.0**	52.9	56.4	54.6	**57.6**	55.5	55.9	55.7	**56.5**	55.4	56.2	54.2
DMLab	**51.5**	49.4	51.5	40.5	**50.9**	50.3	47.0	38.3	**49.4**	47.3	39.4	38.9	**45.6**	45.3	35.7	35.1
dSprites-loc	**44.1**	41.4	43.9	37.1	**44.9**	42.1	43.9	37.4	**43.5**	40.0	42.1	8.8	**45.3**	41.4	19.7	35.2
dSprites-ori	**29.8**	29.7	29.4	22.9	**30.7**	29.8	27.4	13.2	**28.8**	28.2	13.6	7.6	**27.4**	27.2	7.6	10.7
KITTI-dist	**52.5**	51.6	51.0	41.6	**53.7**	51.9	51.9	46.4	54.8	**54.9**	43.3	49.0	**52.0**	50.1	47.7	49.5
sNORB-azim	**26.3**	24.4	25.4	10.1	**26.2**	25.8	21.0	7.1	**27.9**	26.9	5.6	6.5	**26.2**	26.1	7.8	8.5
sNORB-elev	**41.0**	40.6	37.2	37.0	**40.4**	38.0	40.1	15.5	**41.5**	38.5	40.7	30.1	**32.0**	31.8	25.8	23.1

Table 15: Semantic segmentation results on ScanNet v2. Val denotes validation mIoU. Bold indicates the best result in the table.

Method	Val
MLP/CNN	
PointNet++ (Qi et al., 2017)	53.5
SparseConvNet (Graham et al., 2018)	69.3
PointConv (Wu et al., 2019)	61.0
JointPointBased (Chiang et al., 2019)	69.2
KPConv (Thomas et al., 2019)	69.2
PointASNL (Yan et al., 2020)	63.5
PointNeXt (Qian et al., 2022)	71.5
LargeKernel3D (Chen et al., 2023)	73.5
PointMetaBase (Lin et al., 2023)	72.8
MinkowskiNet (Choy et al., 2019)	72.2
Transformer	
Fast Point Transformer (Park et al., 2022)	72.1
Stratified Transformer (Lai et al., 2022)	74.3
PointConvFormer (Wu et al., 2023)	74.5
OctFormer (Wang, 2023)	75.7
Swin3D (Yang et al., 2025b)	76.4
PTv1 (Zhao et al., 2021)	70.6
PTv2 (Wu et al., 2022)	75.4
PTv3 (Wu et al., 2024a)	77.6
Mamba	
Point Mamba (Liu et al., 2024a)	74.6
Pamba (Li et al., 2025a)	77.6
PTv3 variants (our experiments)	
MLP (dense baseline)	77.6
RaLiMP ($k = 5$)	**78.1**
RaLiMP ($k = 10$)	77.6
RaLiMP ($k = 22$)	77.3

Table 16: Semantic segmentation results on nuScenes. Val denotes validation mIoU. Bold indicates the best result in the table.

Method	Val
MLP/CNN/SparseConv	
SPVNAS (Tang et al., 2020)	77.4
Cylinder3D (Zhu et al., 2021)	76.1
PVKD (Hou et al., 2022)	76.0
2DPASS (Yan et al., 2022)	79.4
MinkUNet (Choy et al., 2019)	73.3
+ M3Net (Liu et al., 2024b)	79.0
+ PPT (Wu et al., 2024b)	78.6
OA-CNNs (Peng et al., 2024)	78.9
Transformer	
SphereFormer (Lai et al., 2023)	78.4
RangeFormer (Kong et al., 2023)	78.1
PTv2 (Wu et al., 2022)	80.2
PTv3 (Wu et al., 2024a)	80.4
Mamba	
Pamba (Li et al., 2025a)	80.4
PTv3 variants (our experiments)	
MLP (dense baseline)	80.4
RaLiMP ($k = 5$)	**80.8**
RaLiMP ($k = 10$)	80.7
RaLiMP ($k = 22$)	80.5

Table 17: GPT-2 Large (774M, OpenWebText) validation perplexity (lower is better) at a fixed 30k-iteration budget. RaLiMP improves on the EUGen and SNNK baselines at every k.

	FFN layers replaced (k)						
Method	2	6	11	18	24	27	30
Low Rank	16.83	17.57	18.79	19.06	20.91	22.66	25.31
EUGen	16.03	16.08	16.37	16.66	17.06	17.74	18.42
SNNK	14.18	14.32	14.71	15.02	15.69	16.05	16.44
RaLiMP (ours)	**14.10**	**14.22**	**14.58**	**14.90**	**15.26**	**15.42**	**15.69**
Dense (no replacement)				14.17			

F Additional Experiments

In this section, we show that RaLiMP provides further compression over already compressed models. To illustrate this point, we finetune DistilBERT Sanh et al. (2020) which is distilled from BERT (Devlin et al., 2019) on the GLUE benchmark (Wang et al., 2018). RaLiMP beats both SNNK and EUGen and further reduces DistilBERT by **35**% while remaining competitive. This shows that our method is complementary to various compression methods and can be combined with them.

In all our experiments we use 32 random features. RaLimP spends that budget on the polynomial of degree $K = 2$ with 8 random features and the rest of the 24 features in the rational block is split among $n_{\text{freq}} = 3$ frequencies with $m = 4$ features each, one centered summand of Eq. 4, and $\tau = 4$. We report the MCC score for CoLA, F1 score for MRPC and QQP, Spearman correlation for STSB, and accuracy scores for the other tasks. We use the AdamW optimizer with a weight decay of 10^{-5}

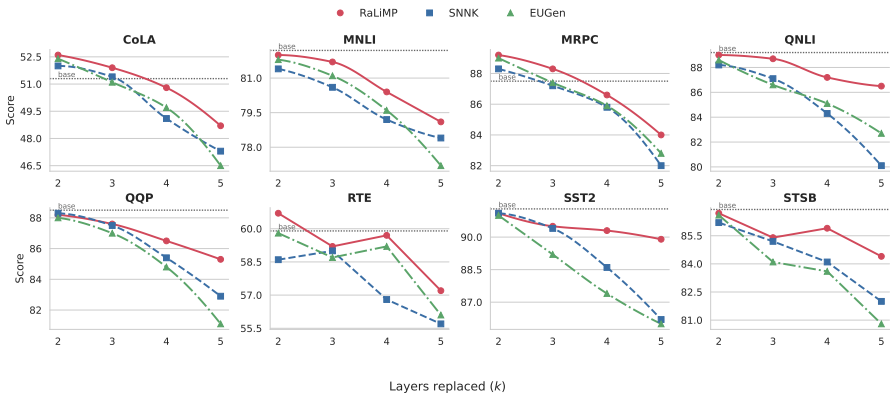

Figure 11: **GLUE results.** MCC score is reported for CoLA, F1 score is reported for MRPC and QQP, Spearman correlation is reported for STSB, and accuracy scores are reported for the other tasks. RaLiMP is competitive with baseline while being almost 35% smaller. Results are an average of 5 seeds.

and a learning rate of 3×10^{-5}, and a constant scheduler with warmup steps 6% of the total number of training steps with a batch size of 16.

G Additional Related Work

In this section we present additional related work to complement the works already discussed in Sec. 1.

The central theme of our work is to reduce the time and space complexity of neural networks. A substantial body of prior work has focused on reducing these computational bottlenecks via dimensionality reduction using the Johnson-Lindenstrauss transform (JLT) (Dasgupta & Gupta, 2003; Dasgupta et al., 2010; Ailon & Liberty, 2013) or via kernel methods with random features (RF) (Rahimi & Recht, 2007). Following the influential work of Rahimi & Recht 2007, considerable effort has been devoted to improving the accuracy of RF-based methods through the use of entangled projections (Yu et al., 2016; Choromanski et al., 2017). Kernels that are widely studied are of the form $\mathbf{K}_f(\mathbf{x}, \mathbf{y}) := f(\mathbf{x}^\top \mathbf{y})$, where $f := \exp$ (Choromanski et al., 2021; Likhosherstov et al., 2022) or f is a polynomial or power series with positive coefficients (Kar & Karnick, 2012; Wacker & Filippone, 2022; Wacker et al., 2023; Kacham et al., 2024; Wacker et al., 2024). However, these approaches are restricted to positive definite kernels, with the recent exception of (Kim et al., 2025). Our work extends this linearization to special classes of rational functions f.

Kernels have also been used to linearize two-layer neural networks (Cho & Saul, 2009; 2011), as well as to analyze more general neural networks through Neural Tangent Kernels (NTKs) (Jacot et al., 2018) and Path Kernels (Domingos, 2020). While NTK-based methods provide useful insights into training dynamics, particularly in the fine-tuning regime where weights remain close to their pre-trained values (Wei et al., 2022; Malladi et al., 2023; Tomihari & Sato, 2024), they do not directly reduce the computational cost of MLPs trained from scratch. A different line of work uses kernel approximations to directly linearize the computation of MLP layers. Sehanobish et al. 2024 introduced Universal Random Features (URF), which separates the dependence on weights and inputs in certain MLP layers. However, its reliance on Fourier transforms of activation functions limits its direct applicability to unbounded functions such as ReLU. More recently, Kim et al. 2025 avoids this requirement by using random features to provide unbiased approximations for MLPs with polynomial activations, while supporting continuous activations through polynomial approximation on bounded intervals.

Other well known methods include pruning (Sun et al., 2024; Ma et al., 2023), quantization (Xiao et al., 2023; Liu et al., 2025; Zhao et al., 2024), knowledge distillation (Li et al., 2023; Sreenivas et al., 2024; Busbridge et al., 2025), and the use of structured matrices in place of dense ones (Potapczynski et al., 2024; Wei et al., 2024) as common strategies for reducing the compu-

tational cost of models with dense layers, including Transformers. For NeRFs, specialized techniques have been developed to improve efficiency (Tancik et al., 2022; Xu et al., 2023; Sun et al., 2022; Reiser et al., 2021; Müller et al., 2022; Hu et al., 2022b; Reiser et al., 2023), which does not generalize beyond that domain.