Rational Sparse Autoencoder

Naiyu Yin [1] **Yue Yu** [1]

Abstract

Sparse autoencoders (SAEs) are standard tools for mechanistic interpretability, but current SAE families are constrained by fixed encoder nonlinearities such as ReLU, JumpReLU, and TopK. This hard-codes a particular sparsity mechanism into the model and can distort the reconstruction-versus-sparsity trade-off. We introduce the *Rational Sparse Autoencoder* (RSAE), which replaces the fixed encoder activation with a trainable rational function. Rational activations are flexible enough to uniformly approximate the activation primitives used by existing SAE families on compact domains (for TopK, the thresholded gate obtained after a separating top-k threshold is supplied), while also providing a richer function class for adapting to the observed pre-activation geometry. We realise this idea through a two-stage pipeline: an initialisation procedure that copies the pre-trained baseline SAE weights, plugs in rational coefficients obtained by the relaxed Remez exchange on synthetic data, and calibrates the scale parameters along with the rational coefficients; followed by a fine-tuning step under the standard sparsity-regularised reconstruction objective. Empirically, on residual-stream activations of three open-weight language models and across all three baseline activation families, the RSAE *strictly improves* on it after the fine-tuning step, both on reconstruction-side metrics (MSE, ℓ_0, alive-feature fraction) and on downstream-behaviour metrics (cross-entropy degradation, loss recovered), without sacrificing feature-level interpretability under sparse probing. These gains are consistent across host language models, across baseline activation families, and across the full range of baseline sparsity we tested, while the upgrade itself adds only a handful of scalar parameters per autoencoder and runs in minutes on a single consumer GPU.

[1]Department of Mathematics, Lehigh University, Bethlehem, PA, USA. Correspondence to: Naiyu Yin <nay224@lehigh.edu>, Yue Yu <yuy214@lehigh.edu>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

1. Introduction

Sparse autoencoders (SAEs) have become a central tool for mechanistic interpretability of large language models, decomposing the internal activations of a transformer into a sparse linear combination of an overcomplete dictionary of monosemantic feature directions (Bricken et al., 2023; Huben et al., 2024; Gao et al., 2025; Rajamanoharan et al., 2024a;b). Despite rapid progress on training procedures and evaluation suites, the widely used released baselines considered here pair the same affine encoder with one of three non-smooth activation primitives: ReLU regularised by an ℓ_1 penalty (Bricken et al., 2023; Bloom, 2024), TopK (Gao et al., 2025), or $\mathrm{JumpReLU}$ (Rajamanoharan et al., 2024b). Each of these primitives carries well-documented pathologies. The ℓ_1-regularised ReLU SAE suffers from magnitude shrinkage of active features and a persistent population of dead latents (Taggart, 2024; Rajamanoharan et al., 2024a; Gao et al., 2025). TopK replaces the soft penalty with a hard cardinality constraint that breaks gradient flow through inactive features and relies on auxiliary revival losses to mitigate dead features. $\mathrm{JumpReLU}$ inserts a learnable per-feature threshold but requires a continuous-relaxation surrogate for back-propagation through its indicator gate.

In this work, we consider the shallow encoder architecture used by SAEs: one affine pre-activation layer followed by a sparse activation block. We show that trainable rational activations can efficiently represent the ReLU, JumpReLU, and supplied-threshold TopK gates used by current SAE families. For discontinuous gates, the approximation holds on compact domains separated by a margin δ from the jump; a direct rational gate of polylogarithmic size in $1/\varepsilon$ and the inverse margin suffices, and the same scalar gate also has a constant-width deep rational realization. Conversely, there are $\mathcal{O}(1)$-parameter rational target maps for which any scalar-output single-layer SAE encoder with piecewise-affine ReLU/JumpReLU/supplied-threshold TopK gates needs $\Omega(\varepsilon^{-1/2})$ activated coordinates to reach accuracy ε. While our theoretical analysis in this work focuses on the shallow SAE setting, we point out that a similar efficiency advantage holds for deep networks as well. Specifically, in

the deep setting, constant-width rational networks achieve a depth upper bound of $\mathcal{O}(\log\log(1/\varepsilon) + \log\log(1/\delta))$, whereas piecewise-affine networks obey a parameter lower bound of $\Omega(\log(1/\varepsilon))$. This separation suggests that replacing fixed SAE gates by trainable rational activations can improve reconstruction fidelity at matched sparsity; the deep-layer results are included as a complementary extension beyond the SAE encoder architecture.

We therefore propose the *Rational Sparse Autoencoder* (RSAE), an SAE whose encoder activation is a learnable rational function applied element-wise to the affine pre-activation, with learnable input/output scales $(C_{\text{in}}, C_{\text{out}})$ that map the per-feature pre-activation distribution into a bounded interval. We then propose a two-step RSAE training algorithm. During the initialization procedure, we copy the pre-trained baseline SAE weights verbatim, plug in rational coefficients obtained by the relaxed Remez exchange (Chen et al., 2018) on synthetic data, and calibrate the scale parameters and the coefficients to the baseline's pre-activation distribution. During the fine-tuning procedure, we unfreeze all parameters and minimise the standard ℓ_1-regularised reconstruction objective. Empirically, the rational function is expressive enough to approximate every baseline activation at low degree on synthetic data. At the SAE level, we evaluate the RSAE on residual-stream activations of three open-weight language models spanning a range of model sizes and against all three baseline activation families, supporting our central claim: the RSAE achieves better fidelity at comparable sparsity and strictly improves the baseline across reconstruction-side metrics (MSE, ℓ_0, alive-feature fraction) and downstream-behaviour metrics (cross-entropy degradation, loss recovered), uniformly across host language models and baseline activation families. These gains are consistent across the full range of baseline sparsity we tested and do not come at the cost of feature-level interpretability under sparse probing. All of this is achieved by adding only a handful of scalar parameters per autoencoder and running for minutes on a single consumer GPU.

Contributions. We introduce *RSAE*, a new sparse autoencoder built on a trainable activation. Our model is grounded in approximation theory tailored to the SAE encoder: trainable scalar rational activations can emulate the fixed ReLU, JumpReLU, and supplied-threshold TopK gates used in shallow SAE encoders with polylogarithmic size, while the converse lower bound shows that scalar-output single-layer piecewise-affine encoders may require $\Omega(\varepsilon^{-1/2})$ activated coordinates for some rational targets. To implement this upgrading strategy, we propose a two-step RSAE training algorithm: an initialisation procedure that copies the pre-trained baseline SAE weights, followed by a fine-tuning procedure that unfreezes all parameters under the standard ℓ_1-regularised reconstruction objective. We empirically verify that the RSAE achieves better fidelity at comparable sparsity and improves the baseline across both reconstruction-side metrics (MSE, ℓ_0, alive-feature fraction) and downstream-behaviour metrics (cross-entropy degradation, loss recovered), uniformly across host language models, baseline activation families, and baseline sparsity levels, while preserving feature-level interpretability under sparse probing and adding only negligible parameter and runtime overhead.

2. Preliminaries and Related Work

Sparse Autoencoders (SAEs) decompose a language model's internal activations $\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}$ into a sparse linear combination of an overcomplete dictionary of $d_{\text{sae}} \gg d_{\text{in}}$ feature directions $\boldsymbol{z} \in \mathbb{R}^{d_{\text{sae}}}$. They follow a skeleton with a pair of encoder and decoder functions (f, g) defined by:

$$\begin{aligned} &\text{Encoder: } \boldsymbol{z} = f(\boldsymbol{x}) := \phi(\boldsymbol{W}_{\text{enc}}\,(\boldsymbol{x} - \boldsymbol{b}_{\text{dec}}) + \boldsymbol{b}_{\text{enc}}), \\ &\text{Decoder: } \hat{\boldsymbol{x}} = g(\boldsymbol{z}) := \boldsymbol{W}_{\text{dec}}\,\boldsymbol{z} + \boldsymbol{b}_{\text{dec}}. \end{aligned} \tag{1}$$

We write $\boldsymbol{h} := \boldsymbol{W}_{\text{enc}}(\boldsymbol{x} - \boldsymbol{b}_{\text{dec}}) + \boldsymbol{b}_{\text{enc}}$ for the pre-activation, so that $\boldsymbol{z} = \phi(\boldsymbol{h})$. Here, columns of $\boldsymbol{W}_{\text{dec}}$ represent decoder dictionary directions used to reconstruct $\boldsymbol{x}$ from the sparse code $\boldsymbol{z}$ with unit ℓ_2-norm. The weights in encoder/decoder functions are optimized using a loss function of the form:

$$\begin{aligned} \mathcal{L}(\boldsymbol{W}) =& \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\Big[\, \big\|\boldsymbol{x} - \hat{\boldsymbol{x}}(\boldsymbol{x};\, \boldsymbol{W})\big\|_2^2 \;+\; \lambda\, S\big(\boldsymbol{z}(\boldsymbol{x};\, \boldsymbol{W})\big)\,\Big], \\ \boldsymbol{W} :=& \{\boldsymbol{W}_{\text{enc}}, \boldsymbol{W}_{\text{dec}}, \boldsymbol{b}_{\text{enc}}, \boldsymbol{b}_{\text{dec}}\}, \end{aligned} \tag{2}$$

where S is a function that penalizes non-sparse decompositions with a tunable sparsity coefficient λ.

There are two objectives in the SAE encoder: sparsity, meaning that only a few elements of the dictionary are necessary, and faithfulness, meaning that the reconstructed $\hat{\boldsymbol{x}}$ is close to the original $\boldsymbol{x}$. To achieve a good balance between these two objectives, three major SAE activations were proposed, which differ in the encoder activation ϕ and the sparsity mechanism S imposed on $\boldsymbol{z}$. The *ReLU SAE* (Bricken et al., 2023; Bloom, 2024) sets $\phi = \text{ReLU}$ and imposes sparsity through an explicit ℓ_1 penalty $S(\boldsymbol{z}) := ||\boldsymbol{z}||_1$. In the original *ReLU SAE*, the soft ℓ_1 penalty leads to a magnitude shrinkage of active features and causes loss of reconstruction fidelity (Taggart, 2024; Rajamanoharan et al., 2024a; Gao et al., 2025). *TopK SAE* (Gao et al., 2025) then proposes to replace the soft penalty with a hard top-k selection $\boldsymbol{z} = \text{TopK}_k(\boldsymbol{h})$ that yields exact $\ell_0 = k$, and the *JumpReLU SAE* (Rajamanoharan et al., 2024b) keeps the ℓ_1-style soft sparsity but inserts a learnable per-feature threshold $\theta_j > 0$ in the activation function ϕ by setting $\phi(\boldsymbol{h}) = \boldsymbol{h} \odot H(\boldsymbol{h} - \boldsymbol{\theta})$, where H is the Heaviside function satisfying $H(z) = 0$ if $z \leq 0$ and $H(z) = 1$ elsewhere. Orthogonal to the development in encoder activations, *Matryoshka SAEs* (Bussmann et al., 2025) reorganise the decoder into nested-prefix dictionaries, and *data-free*

SAEs (Laptev et al., 2025) fit dictionaries directly from model weights without streaming activations.

While related variants such as the *Gated SAE* (Rajamanoharan et al., 2024a), ProLU (Taggart, 2024), *BatchTopK SAE* (Bussmann et al., 2024), and end-to-end SAE training (Braun et al., 2024) modify thresholds, gates, batch-level sparsity, or the training objective, our theoretical and empirical comparisons focus on the widely used released baselines considered here—ReLU, JumpReLU, and TopK SAEs—whose encoder nonlinearities are fixed functional forms with sparsity controlled by a penalty coefficient, learned threshold, or cardinality budget rather than by a trainable rational activation. Through a trainable activation architecture supported by approximation theory, our RSAE provides a drop-in modification for pre-trained SAEs (teacher model): while sustaining a similar level of sparsity, it strictly improves model fidelity.

Evaluation benchmarks and pretrained baselines. *SAEBench* (Karvonen et al., 2025) provides matched pretrained ReLU, JumpReLU, and TopK SAEs across model sizes and a unified evaluation suite (covering reconstruction, sparsity, downstream performance, and interpretability metrics); we use the pre-trained ReLU, JumpReLU, and TopK SAEs for our Pythia-160m and Gemma-2-2B baselines, and reuse its sparse-probing harness in §5. For GPT-2 small we additionally use Bloom's *gpt2-small-res-jb* release (Bloom, 2024) as the ReLU baseline and OpenAI's v5 release (Gao et al., 2025) as the TopK baseline.

Rational Neural Networks were built on the key theoretical advantages of rational functions in approximating nonsmooth functions (Newman, 1979; Telgarsky, 2017; Beckermann & Townsend, 2017; Chen et al., 2018). In Boullé et al. (2020), a rational function is employed as a learnable replacement for ReLU or $\tanh$ in feed-forward networks for image classification tasks. A superior performance was also demonstrated in operator learning and PDE surrogates, where the spectral density of rational approximation accelerates convergence on smooth target operators (Trimmel et al., 2022). In rational neural networks, the standard activation function in a feed-forward layer is replaced with a trainable rational function $\frac{P(t)}{Q(t)}$. A naive learnable denominator $Q(t)$ can develop divergent poles during training when Q approaches zero; to prevent this, Molina et al. (2020) introduced the Padé Activation Unit (PAU) by setting $Q(t) := 1 + \sum_{j=1}^{q} b_j\, t^j$. Dunefsky et al. (2024) subsequently proposed the *safe-Padé* parameterisation by parameterizing $Q(t)$ as $1 + |\sum_{j=1}^{q} b_j\, t^j|$, which guarantees pole-free, Lipschitz rational activations.

While prior works have demonstrated the theoretical advantages of rational activation functions against continuous activation functions such as ReLU, GeLU, or $\tanh$ (Boullé et al., 2020; Molina et al., 2020; Delfosse et al., 2021; Trimmel et al., 2022; Tang & Townsend, 2026), little discussion has focused on discontinuous SAE gates such as $\mathrm{JumpReLU}$ and TopK. Moreover, prior rational-network theory is usually stated for deeper feed-forward architectures, whereas the SAE encoder in (1) is shallow: a single affine pre-activation layer followed by coordinatewise gates and a linear decoder. This mismatch motivates a theory whose main separation is stated for the single-layer encoder, with deep rational realizations kept as an additional comparison. In this work we show that trainable rational activations give a more efficient approximation class for the fixed gates used in shallow SAE encoders, and then leverage their approximation power and smooth gradients to obtain SAEs with lower reconstruction error and fewer dead features at matched sparsity.

3. Rational Sparse Autoencoder

Herein, we propose the *Rational Sparse Autoencoder* (RSAE), an overcomplete sparse autoencoder whose encoder activation is a **trainable** rational function. We retain the standard SAE skeleton of (1) and modify *only* the encoder activation $\phi(\cdot)$. Let $\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}$ and write $\boldsymbol{h} = \boldsymbol{W}_{\text{enc}}(\boldsymbol{x} - \boldsymbol{b}_{\text{dec}}) + \boldsymbol{b}_{\text{enc}} \in \mathbb{R}^{d_{\text{sae}}}$ for the pre-activation. The RSAE activation is applied element-wise to the pre-activation $\boldsymbol{h}$ as

$$\begin{aligned} \phi(\boldsymbol{h}) &= C_{\text{out}} \cdot r_{(\boldsymbol{a},\boldsymbol{b})}\Big(\frac{\boldsymbol{h}}{C_{\text{in}}}\Big), \\ r_{(\boldsymbol{a},\boldsymbol{b})}(t) &= \frac{P(t)}{Q(t)} \;=\; \frac{\sum_{i=0}^{p} a_i\, t^i}{\sum_{j=0}^{q} b_j\, t^j},\; t \in [-1, 1]. \end{aligned} \tag{3}$$

Because $r_{(\boldsymbol{a},\boldsymbol{b})}(\cdot)$ admit arbitrarily small uniform error of discontinuous activation functions inside a bounded compact interval $[-1, 1]$, we introduce learnable scaling parameters $C_{\text{in}}, C_{\text{out}} > 0$ with the purpose of mapping pre-activation $\boldsymbol{h}$ into the rational's design interval and the rational's output back to the feature magnitude expected by the decoder.

We now analyze why rational activations are a natural upgrade for the SAE encoder in (1). The main setting is shallow: an affine pre-activation layer followed by a sparse activation block. We prove that ReLU, JumpReLU, and the supplied-threshold TopK gate can each be replaced by trainable rational gates of polylogarithmic size in the target accuracy, with a margin parameter for discontinuous gates. The TopK statement concerns the thresholded gate equivalent to TopK conditional on a supplied threshold, not the full order-statistic operator that computes the k-th threshold. We also state the corresponding constant-width deep rational realizations, connecting the SAE result to the rational-network literature (Boullé et al., 2020; Tang & Townsend, 2026). For the converse direction, we exhibit an $\mathcal{O}(1)$-parameter rational target map that cannot be approximated efficiently by

scalar-output piecewise-affine encoders in the same shallow form: any such single-layer ReLU/JumpReLU/supplied-threshold TopK encoder needs $\Omega(\varepsilon^{-1/2})$ activated coordinates. Together, these results explain why replacing the fixed activation in a pretrained SAE by a trainable rational activation can increase the encoder's approximation power without changing the linear backbone.

Our theoretical result is based on the family of Zolotarev sign functions, which are geometrically convergent rational approximants of $\mathrm{sign}(x)$ on the gap-separated set $E_\delta := [-1,-\delta]\cup[\delta,1]$ for any $\delta \in (0,1)$. We state the quantitative bounds below, and entail all proofs in Appendix:

Lemma 1 (Rational approximation of sign)**.** *For every $\delta \in (0,1)$ and $n \geq 1$ there is a type-$(2n+1, 2n)$ rational $s_{n,\delta}$ such that $\sup_{x\in E_\delta} \left|\mathrm{sign}(x) - s_{n,\delta}(x)\right| \leq 4\exp\big(-\pi^2 n/\log(4/\delta)\big)$.*

Consequently, for every $0 < \varepsilon < 1$, there is a rational function of size $\mathcal{O}\big(\log(1/\varepsilon)\log(1/\delta)\big)$ that approximates sign *on E_δ to uniform error ε. For deep-layer networks, there is a constant-width rational network of depth $\mathcal{O}\big(\log\log(1/\varepsilon) + \log\log(1/\delta)\big)$ that approximates* sign *on E_δ to uniform error ε.*

This lemma enables two complementary rational implementations of the activation gates below: a direct rational gate of the stated size for the shallow SAE encoder, and a constant-width deep rational realization as considered in the previous rational network approximation results for continuous activation functions such as ReLU (Boullé et al., 2020) and GeLU (Tang & Townsend, 2026). To provide analysis for JumpReLU and TopK, we denote:

$$\begin{aligned}
\text{(ReLU)} \quad \boldsymbol{z}_{\mathrm{R}}(\boldsymbol{h}) &= \mathrm{ReLU}(\boldsymbol{h}) = \boldsymbol{h} \odot H(\boldsymbol{h}) \\
&= \boldsymbol{h} \odot \frac{\mathrm{sign}(\boldsymbol{h})+1}{2}, \\
\text{(JumpReLU)} \quad \boldsymbol{z}_{\mathrm{J}}(\boldsymbol{h}) &= \mathrm{JumpReLU}(\boldsymbol{h}) \\
&= \boldsymbol{h} \odot H(\boldsymbol{h}-\boldsymbol{\theta}) \\
&= \boldsymbol{h} \odot \frac{\mathrm{sign}(\boldsymbol{h}-\boldsymbol{\theta})+1}{2}, \\
\text{(TopK gate with threshold)} \quad \boldsymbol{z}_{\mathrm{T}}(\boldsymbol{h};\tau_k) &= \mathrm{TopK}(\boldsymbol{h};\tau_k) \\
&= \boldsymbol{h} \odot \frac{\mathrm{sign}(\boldsymbol{h}-\tau_k)+1}{2},
\end{aligned} \tag{4}$$

where H is the Heaviside function, $\boldsymbol{\theta} \in (\mathbb{R}^+)^{d_{\mathrm{sae}}}$ is the per-feature threshold. For TopK, let $h_{(1)} \geq \cdots \geq h_{(d_{\mathrm{sae}})}$ denote the sorted pre-activations, with $1 \leq k < d_{\mathrm{sae}}$. We use τ_k for a supplied separating threshold satisfying $h_{(k+1)} < \tau_k < h_{(k)}$; under margin δ, this means $h_{(k+1)} + \delta \leq \tau_k \leq h_{(k)} - \delta$. Thus τ_k is not the literal k-th largest entry, which would lie on the discontinuity. Thus $\boldsymbol{z}_{\mathrm{T}}(\boldsymbol{h};\tau_k)$ is the thresholded gate equivalent to TopK conditional on the supplied threshold; our rational approximation result does not include the order-statistic computation that obtains τ_k from $\boldsymbol{h}$.

For the ReLU activation, we use the approximation theorem of Boullé et al. (2020):

Theorem 2 (Rational approximation of ReLU Boullé et al. (2020))**.** *For every $0 < \varepsilon < 1$, there exists a scalar rational function $R_\varepsilon : [-1,1] \to [-1,1]$ of size*

$$\mathcal{O}\Big(\log^2(1/\varepsilon)\Big),$$

such that

$$\sup_{x\in[-1,1]} \left|R_\varepsilon(x) - \mathrm{ReLU}(x)\right| \;\leq\; \varepsilon.$$

Consequently, the ReLU activation block can be replaced in either of two implementations. First, R_ε can be applied coordinatewise as a trainable rational activation, with scalar size $\mathcal{O}\big(\log^2(1/\varepsilon)\big)$. Second, the same scalar map can be realized by a constant-width deep rational network of internal depth

$$M_R \;=\; \mathcal{O}\Big(\log\log(1/\varepsilon)\Big).$$

Under either implementation, the resulting activation block $\mathcal{R}_R : [-1,1]^{d_{\mathrm{sae}}} \to \mathbb{R}^{d_{\mathrm{sae}}}$ satisfies

$$\sup_{\boldsymbol{h}\in[-1,1]^{d_{\mathrm{sae}}}} \left\|\mathcal{R}_R(\boldsymbol{h}) - \boldsymbol{z}_{\mathrm{R}}(\boldsymbol{h})\right\|_\infty \;\leq\; \varepsilon.$$

JumpReLU is discontinuous at $h_i = \theta_i$, so uniform approximation is only meaningful on a domain bounded away from the jump. We therefore fix a *margin* $\delta > 0$ and define $\Omega_\delta := \{\boldsymbol{h} \in [-1,1]^{d_{\mathrm{sae}}} : |h_i - \theta_i| \geq \delta,\ \forall i\}$ [1]. We then have the approximation results on Ω_δ:

Theorem 3 (Rational approximation of JumpReLU)**.** *For every $0 < \varepsilon < 1$, the JumpReLU activation block on Ω_δ can be replaced in either of two implementations. First, each coordinate map can be implemented directly as a trainable scalar rational activation of size*

$$\mathcal{O}\Big(\log(1/\varepsilon)\log(1/\delta)\Big),$$

with constants depending only on the fixed threshold scale. Second, each coordinate map can be realized by a constant-width deep rational network of internal depth

$$M_J \;=\; \mathcal{O}\Big(\log\log(1/\varepsilon) + \log\log(1/\delta)\Big)$$

and per-coordinate size $\mathcal{O}(M_J)$. Under either implementation, the resulting activation block $\mathcal{R}_J : [-1,1]^{d_{\mathrm{sae}}} \to \mathbb{R}^{d_{\mathrm{sae}}}$ satisfies

$$\sup_{\boldsymbol{h}\in\Omega_\delta} \left\|\mathcal{R}_J(\boldsymbol{h}) - \boldsymbol{z}_{\mathrm{J}}(\boldsymbol{h})\right\|_\infty \;\leq\; \varepsilon.$$

[1] This is the standard domain restriction needed for uniform approximation of a discontinuous threshold map: without excluding a δ-neighbourhood of the jump, no continuous or rational approximant can achieve arbitrarily small uniform error. In applications, δ should therefore be interpreted as a lower bound on the threshold margin of the pre-activations under consideration.

For TopK, fix $1 \le k < d_{\text{sae}}$ and consider a simplified setting in which the pretrained teacher supplies the scalar threshold τ_k that determines the active support. Here τ_k is a separating threshold between the k-th and $(k+1)$-st order statistics, not the k-th activation itself; for example, one may take $\tau_k = (h_{(k)} + h_{(k+1)})/2$ when $h_{(k)} - h_{(k+1)} \ge 2\delta$. The result below therefore approximates the thresholded gate equivalent to TopK conditional on a supplied threshold, not the full TopK operator itself. Similar to the JumpReLU case, we require a margin-separated domain with sorted coordinates $h_{(1)} \ge \cdots \ge h_{(d_{\text{sae}})}$,

$$\Omega_\delta^{\mathrm{T}} := \big\{(\boldsymbol{h}, \tau_k) \in [-1,1]^{d_{\text{sae}}} \times [-1,1] : h_{(k)} - \tau_k \ge \delta,\ \tau_k - h_{(k+1)} \ge \delta\big\},$$

and obtain:

Theorem 4 (Rational approximation of supplied-threshold TopK gate)**.** *Suppose the scalar threshold $\tau_k \in [-1,1]$ is supplied together with each pre-activation vector and satisfies the separating margin condition above. For every $0 < \varepsilon < 1$, the supplied-threshold TopK gate on $\Omega_\delta^{\mathrm{T}}$ can be replaced in either of two implementations. First, each coordinate map can be implemented directly as a trainable scalar rational activation of size*

$$\mathcal{O}\Big(\log(1/\varepsilon)\log(1/\delta)\Big).$$

Second, each coordinate map can be realized by a constant-width deep rational network of internal depth

$$M_T \;=\; \mathcal{O}\Big(\log\log(1/\varepsilon) + \log\log(1/\delta)\Big).$$

Under either implementation, the resulting network $\mathcal{R}_T : [-1,1]^{d_{\text{sae}}+1} \to \mathbb{R}^{d_{\text{sae}}}$ satisfies

$$\sup_{(\boldsymbol{h},\tau_k)\in\Omega_\delta^{\mathrm{T}}} \big\|\mathcal{R}_T(\boldsymbol{h},\tau_k) - \boldsymbol{z}_{\mathrm{T}}(\boldsymbol{h};\tau_k)\big\|_\infty \;\le\; \varepsilon.$$

We now turn to the converse question of approximating rational functions:

Theorem 5 (Lower bound for ReLU/JumpReLU/TopK networks)**.** *Fix $\eta \in (0, 1/2)$ and define the rational target*

$$\mathcal{R}_\eta^\star(x) := \frac{\eta^2}{x^2+\eta^2}, \qquad x \in [-1,1].$$

This target satisfies $\mathcal{R}_\eta^\star : [-1,1] \to [0,1]$ and can be realized with $\mathcal{O}(1)$ rational parameters. Then any scalar map $\mathcal{S} : [-1,1] \to [0,1]$ realized by a ReLU/JumpReLU/supplied-threshold TopK network and satisfying

$$\|\mathcal{S} - \mathcal{R}_\eta^\star\|_{L^\infty([-1,1])} \le \varepsilon$$

must satisfy $P = \Omega(\log(1/\varepsilon))$, where P is the number of trainable parameters. If $\mathcal{S}$ is realized by the scalar-output version of the single-layer encoder architecture in (1) with N activated coordinates, then it must satisfy $N = \Omega(\varepsilon^{-1/2})$.

Together, Lemma 1 and Theorems 2–5 show an expressive asymmetry in the SAE encoder setting: trainable rational activations give compact approximations to the fixed gates used by current SAEs, whereas scalar-output single-layer piecewise-affine encoders can require polynomially many activated coordinates for simple rational targets. This supports the expectation that RSAEs can provide better reconstruction fidelity at matched sparsity.

4. Practical Algorithm

As indicated by our analysis, the rational activation function is guaranteed to provide a better approximation power given a teacher model and supplied threshold. The RSAE is then constructed in two steps: (i) an *initialization procedure* that produces high-quality rational coefficients $(\boldsymbol{a}, \boldsymbol{b})$ and learnable scales $(C_{\text{in}}, C_{\text{out}})$ by first fitting a rational function on a bounded interval of synthetic data and then adapting the rational activation to the teacher SAE's pre-activation distribution; and (ii) a *fine-tuning procedure* that jointly optimises all parameters, including the encoder and decoder weights, under the standard ℓ_1-regularised reconstruction objective.

Step 1: RSAE Initialization Procedure. Let $\phi^{\text{teacher}} \in \{\mathrm{ReLU},\ \mathrm{JumpReLU}_\theta,\ \mathrm{TopK}_k\}$ denote any of the activation primitives used by the baseline SAE families considered here, and let $\{(t_\ell, y_\ell)\}_{\ell=1}^N$ be a uniform dense grid on $[-1,1]$ with $y_\ell = \phi^{\text{teacher}}(t_\ell)$. We first fit a rational function on this bounded interval to obtain coefficients that approximate the teacher activation to high accuracy. To this end, we employ the *relaxed Remez exchange* of Chen et al. (2018), an iterative procedure that alternates a linearised coefficient solve with a node-exchange step until the residual equioscillates, to solve the min–max objective

$$(\boldsymbol{a}^*, \boldsymbol{b}^*) \;=\; \arg\min_{\boldsymbol{a},\boldsymbol{b}} \max_{t\in[-1,1]} \big|\, r_{(\boldsymbol{a},\boldsymbol{b})}(t) \;-\; \phi^{\text{teacher}}(t)\,\big|. \quad (5)$$

The full algorithmic details, including the linearised system (9) solved at each outer iteration, are deferred to Appendix B.2. Remez returns the standard-Padé coefficients used directly in (3); this synthetic fitting is performed once per teacher activation and the resulting coefficients $(\boldsymbol{a}^*, \boldsymbol{b}^*)$ can be tabulated for reuse.

Given a pre-trained baseline SAE with weights $\{\widetilde{\boldsymbol{W}}_{\text{enc}}, \widetilde{\boldsymbol{b}}_{\text{enc}}, \widetilde{\boldsymbol{W}}_{\text{dec}}, \widetilde{\boldsymbol{b}}_{\text{dec}}\}$ and the rational coefficients $(\boldsymbol{a}^*, \boldsymbol{b}^*)$, we then adapt the rational activation to the teacher's pre-activation distribution by minimising

$$\begin{aligned}&(\widetilde{\boldsymbol{a}}, \widetilde{\boldsymbol{b}}, \widetilde{C}_{\text{in}}, \widetilde{C}_{\text{out}})\\ &= \arg\min_{\boldsymbol{a},\boldsymbol{b},C_{\text{in}},C_{\text{out}}} \big\|\phi(\boldsymbol{h};\, \boldsymbol{a}, \boldsymbol{b}, C_{\text{in}}, C_{\text{out}}) \;-\; \phi^{\text{teacher}}(\boldsymbol{h})\big\|_2^2,\end{aligned} \quad (6)$$

where $\phi(\boldsymbol{h};\, \boldsymbol{a}, \boldsymbol{b}, C_{\text{in}}, C_{\text{out}})$ is computed via (3) and $\boldsymbol{h} = \widetilde{\boldsymbol{W}}_{\text{enc}}\,(\boldsymbol{x} - \widetilde{\boldsymbol{b}}_{\text{dec}}) + \widetilde{\boldsymbol{b}}_{\text{enc}}$ is the teacher's pre-activation. Combined with the inherited encoder and decoder weights, the learned coefficients and scales allow the RSAE to approx-

imately reproduce the teacher's output, up to the Step 1 approximation error.

Step 2: RSAE Fine-Tuning Procedure. We initialise the RSAE encoder and decoder with the teacher weights $\{\widetilde{\boldsymbol{W}}_{\text{enc}}, \widetilde{\boldsymbol{b}}_{\text{enc}}, \widetilde{\boldsymbol{W}}_{\text{dec}}, \widetilde{\boldsymbol{b}}_{\text{dec}}\}$, and the rational activation with the learned coefficients $(\widetilde{\boldsymbol{a}}, \widetilde{\boldsymbol{b}})$ and scales $(\widetilde{C}_{\text{in}}, \widetilde{C}_{\text{out}})$. We then unfreeze all parameters $\Theta := \{\boldsymbol{W}_{\text{enc}}, \boldsymbol{b}_{\text{enc}}, \boldsymbol{W}_{\text{dec}}, \boldsymbol{b}_{\text{dec}}, \boldsymbol{a}, \boldsymbol{b}, C_{\text{in}}, C_{\text{out}}\}$ and minimise the ℓ_1-regularised objective

$$\min_{\Theta} \ \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\Big[\, \big\|\boldsymbol{x} - \hat{\boldsymbol{x}}(\boldsymbol{x};\,\Theta)\big\|_2^2 \ + \ \lambda \,\big\|\boldsymbol{z}(\boldsymbol{x};\,\Theta)\big\|_1 \Big]. \quad (7)$$

5. Empirical Results

5.1. Rational Function Approximation Performance on Synthetic Data

Setup. We evaluate the three rational coefficient fitting procedures of §4 on the activation primitives used by the baseline SAE families considered here. Each fitting procedure is run on a uniform, dense grid of $N = 4001$ points over $[-1, 1]$ with target activation functions ReLU and $\mathrm{JumpReLU}$. We report the mean-squared error (MSE) of the fitted rational functions. In particular, we run the relaxed Remez exchange in both the standard-Padé form (identical to the original formulation) and the safe-Padé form. We additionally fit the safe-Padé coefficients directly under the L^2 and smoothed-L^∞ surrogates as baselines. We deliberately omit TopK from the synthetic study because, conditioned on a given input batch, TopK is pointwise equivalent to a supplied-threshold JumpReLU-type gate with a sample-dependent separating threshold τ_k between the k-th and $(k+1)$-st largest pre-activations, e.g. their midpoint when the TopK gap is positive. Therefore any rational that approximates the $\mathrm{JumpReLU}$ family uniformly over the threshold also approximates TopK on the corresponding batch.

Approximation precision. Figure 1 shows that the Remez procedure fits both ReLU and $\mathrm{JumpReLU}$ to high precision on $[-1, 1]$: even in the immediate neighbourhood of the kink (Figure 1(b)) and the jump (Figure 1(d)), the fits are visually indistinguishable from the teacher. Remez reaches a low MSE of 3.8×10^{-7} on ReLU at type $(15, 14)$ and 2.4×10^{-6} on $\mathrm{JumpReLU}$ with $\theta = 0.1$ at type $(19, 18)$, outperforming the L^2 and L^∞ approaches. While Figure 1(c) shows the fit for JumpReLU with $\theta = 0.1$, we additionally perform ablation studies on JumpReLU with larger discontinuities $\theta \in \{0.2, \ldots, 0.5\}$ (Table 7, Appendix B.3); fitting performance remains consistent across discontinuities.

Choice of degrees. To choose a suitable degree for our algorithm, we perform an ablation over (p, q) for all four procedures; the results are reported in Figure 3 (Appendix B.3), which plots MSE against the numerator degree p. In general, Remez first exhibits the expected near-exponential decay; numerical conditioning of the linearised system (9) then dominates, and the curve flattens or oscillates. Empirically, a single low-degree rational ($(3, 2)$ for ReLU, $(9, 8)$ for JumpReLU and TopK) is expressive enough to reproduce every activation primitive used by current SAE baselines to within numerical precision.

5.2. Rational SAE Performance

Models, SAEs, and Evaluation Metrics. We evaluate the RSAE on residual-stream activations from three open-weight language models of various sizes: **GPT-2 small**, **Pythia-160m-deduped**, and **Gemma-2-2B**. Teacher SAEs are taken from publicly released checkpoints: GPT-2 small from Bloom (Bloom, 2024)'s `gpt2-small-res-jb` (ReLU) and the OpenAI-v5 (Gao et al., 2025) `gpt2-small-resid-post-v5-32k` (TopK); Pythia-160m and Gemma-2-2B from SAEBench (Karvonen et al., 2025) for ReLU, JumpReLU, and TopK. Following the standard protocol, we evaluate our RSAE against SAE baselines in terms of five metrics: (1) reconstruction MSE $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_F^2$, (2) ℓ_0 at the $|z| > 10^{-6}$ threshold, (3) the fraction of alive latents, (4) the cross-entropy degradation when the SAE intercepts the residual stream $\Delta\mathrm{CE} = \mathrm{CE}_{\hat{\boldsymbol{x}}} - \mathrm{CE}_{\boldsymbol{x}}$ (lower is better), (5) the loss-recovered fraction $\mathrm{LR} = (\mathrm{CE}_{\text{zero}} - \mathrm{CE}_{\hat{\boldsymbol{x}}})/(\mathrm{CE}_{\text{zero}} - \mathrm{CE}_{\boldsymbol{x}})$ (higher is better). Please refer to Appendix B.1 for more implementation details.

The experiments are organised around two claims: **(C1)** the rational activation, under the proposed initialization procedure in Algorithm 1, approximately reproduces the baseline SAE teachers' behaviour at initialisation; **(C2)** after a joint fine-tune, the RSAE improves on the teacher across the great majority of reconstruction- and downstream-behaviour metrics, uniformly across host language models and teacher activation families.

(C1) Approximate reproduction at initialisation. Table 1 establishes **(C1)**: for every (model, teacher) pair, the *RSAE init* row closely tracks the teacher row, especially on reconstruction MSE, while small ℓ_0 and alive-feature differences remain in some cases. This is the expected outcome of the RSAE initialization procedure: the rational activation approximately reproduces the teacher's pre-activation $\boldsymbol{h}$ to activation $\boldsymbol{z}$ map, yielding similar SAE evaluations before fine-tuning.

(C2) Strict improvement after fine-tuning. After 2K Adam steps, the *RSAE* row beats the teacher across the great majority of cells in Tables 1 and 2: $22/24$ reconstruction-axis cells in Table 1 strictly improve over the teacher (the only exceptions a 1.6-token regression in ℓ_0 on ReLU/GPT-2 small and an alive tie at 99.9% on TopK/Gemma-2-2B), and $13/16$ downstream-axis cells in Table 2 strictly improve (the only exceptions a marginal regression on ReLU/Pythia-

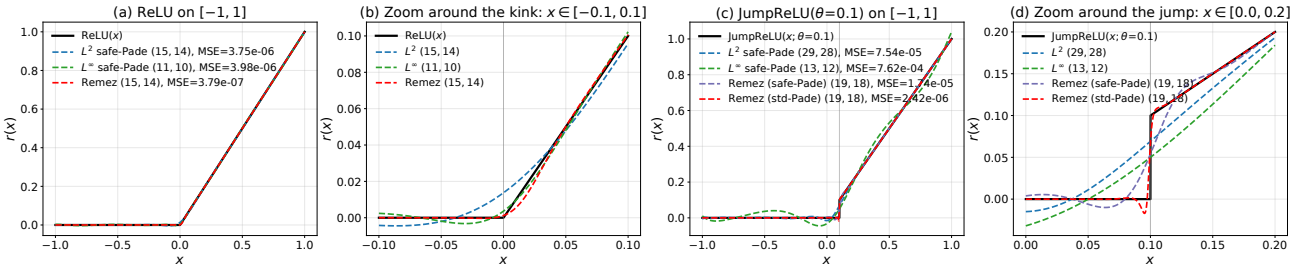

Figure 1. **Rational approximation of SAE activation primitives (ReLU and JumpReLU) on** $[-1, 1]$**.** Best-MSE rational fits of ReLU (figure (a) and figure (b)) and JumpReLU with $\theta = 0.1$ (figure (c) and figure (d)) under three procedures: the relaxed Remez exchange (red for standard-Padé and purple for safe-Padé), the L^2 fit (blue), and the smoothed L^∞ fit (green). Figure (b) and figure (d) zoom into the kink and the jump, respectively. Each curve uses the optimal (p, q) for its procedure. Across figure (a) - (d), the Remez fit is visually indistinguishable from the teacher activations, validating the universal-approximation claim.

Table 1. **Main results:** teacher SAE, RSAE after initialization (RSAE init), and RSAE after fine-tuning on residual-stream activations of GPT-2 small, Pythia-160m, and Gemma-2-2B.

SAEs	GPT-2 small (layer 6)			Pythia-160m (layer 8)			Gemma-2-2B (layer 12)		
	$\lVert \boldsymbol{x}-\hat{\boldsymbol{x}} \rVert_F^2 \downarrow$	$\ell_0 \downarrow$	alive$\uparrow$	$\lVert \boldsymbol{x}-\hat{\boldsymbol{x}} \rVert_F^2 \downarrow$	$\ell_0 \downarrow$	alive$\uparrow$	$\lVert \boldsymbol{x}-\hat{\boldsymbol{x}} \rVert_F^2 \downarrow$	$\ell_0 \downarrow$	alive$\uparrow$
ReLU SAE	5.97×10^{-1}	51.4	89.5%	5.78×10^{-2}	157.7	72.9%	1.9394	135.6	79.0%
RSAE init (ReLU as teacher)	5.97×10^{-1}	52.6	91.0%	5.79×10^{-2}	161.2	72.9%	1.9394	135.6	79.0%
RSAE	$\mathbf{5.30\times10^{-1}}$	53.0	**91.6%**	$\mathbf{5.24\times10^{-2}}$	**149.0**	**74.5%**	**1.8955**	**127.4**	**79.1%**
JumpReLU SAE	no public release			2.68×10^{-1}	361.4	38.9%	3.8397	212.3	99.6%
RSAE init (JumpReLU as teacher)	–	–	–	2.68×10^{-1}	361.2	38.9%	3.8230	260.4	99.7%
RSAE	–	–	–	$\mathbf{3.20\times10^{-2}}$	**160.8**	**39.7%**	**1.7887**	**211.7**	**99.7%**
TopK SAE	6.71×10^{-2}	32.0	71.3%	2.99×10^{-2}	160.0	78.1%	1.4278	160.0	99.9%
RSAE init (TopK as teacher)	6.73×10^{-2}	33.1	71.3%	3.16×10^{-2}	152.7	78.1%	1.3731	220.4	99.9%
RSAE	$\mathbf{5.96\times10^{-2}}$	**30.7**	**71.9%**	$\mathbf{2.73\times10^{-2}}$	**151.3**	**80.8%**	**1.3990**	**158.9**	99.9%

160m and a tied ΔCE of 0.093 on TopK/Gemma-2-2B). The wins hold uniformly across the three host language models, all three teacher activation families, and both reconstruction- and downstream-axis metrics, so the improvement is not an artefact of any single architecture, host model, or evaluation axis. Together, the (C1) approximate match at initialisation and the (C2) wins after fine-tuning are consistent with the shallow-encoder rational-vs-piecewise expressivity asymmetry of §3: the rational activation contains every fixed-form teacher within a single low-degree family, and is then free to deviate from any of them in whatever direction lowers the regularised reconstruction loss on the host model's actual activation distribution.

Attribution of the gains. Because each teacher SAE is trained to convergence on the same activation buffer and at the same sparsity level used for RSAE fine-tuning, continuing to fine-tune the teacher with its *fixed* activation under the same objective is a stationary point and leaves it unchanged up to optimization noise; the teacher row therefore already serves as the converged fixed-activation control. Since RSAE inherits the teacher's weights and—per the *RSAE init* row—reproduces the teacher before fine-tuning, the only added degrees of freedom are the rational activation's scalar parameters. The improvements in Tables 1–2 are thus attributable to the trainable activation itself, rather than to additional optimization steps or any data/distribution difference.

Runtime and scalability. Table 3 reports the wall-clock cost of the RSAE training procedures and shows two properties of the method: **(i) Cheap approximate teacher reproduction.** The Remez fit is a one-shot off-line procedure depending only on the target activation, so it can be tabulated once and reused across (model, teacher) pairs. The remaining model-specific 500-step adaptation to the teacher pre-activation spaces completes in 25 to 60 s, including Gemma-2-2B. Combined with verbatim weight inheritance, this lets RSAE approximately reproduce ReLU, JumpReLU, and TopK teachers under one framework. **(ii) Lightweight fine-tune overhead.** Fine-tuning adds at most $(p+1)+q+2$ scalar parameters per RSAE, negligible relative to the $\mathcal{O}(d_{\text{in}}d_{\text{sae}})$ encoder/decoder parameters. The corresponding 2K-step fine-tune costs 80 to 295 s on the small models and $\sim$5 min on Gemma-2-2B, so upgrading a released teacher costs minutes, not hours, on a single RTX 5090.

Ablation studies on sparsity. We perform ablations along two complementary sparsity axes: the RSAE's own ℓ_1 coefficient λ (Figure 2) and the teacher SAE's training sparsity

Table 2. **Downstream behavior:** cross-entropy degradation and loss recovered when the SAE intercepts the residual stream of GPT-2 small, Pythia-160m, and Gemma-2-2B.

SAEs	GPT-2 small (layer 6)		Pythia-160m (layer 8)		Gemma-2-2B (layer 12)	
	ΔCE ↓	LR ↑	ΔCE ↓	LR ↑	ΔCE ↓	LR ↑
ReLU SAE	0.180	97.17%	0.259	95.71%	0.283	97.30%
RSAE	**0.123**	**98.07%**	0.279	95.41%	**0.273**	**97.39%**
JumpReLU SAE	no public release		0.682	88.86%	0.127	98.79%
RSAE	–	–	**0.118**	**98.10%**	**0.125**	**98.82%**
TopK SAE	0.136	97.89%	0.232	95.94%	0.093	99.11%
RSAE	**0.092**	**98.55%**	**0.097**	**98.31%**	0.093	**99.76%**

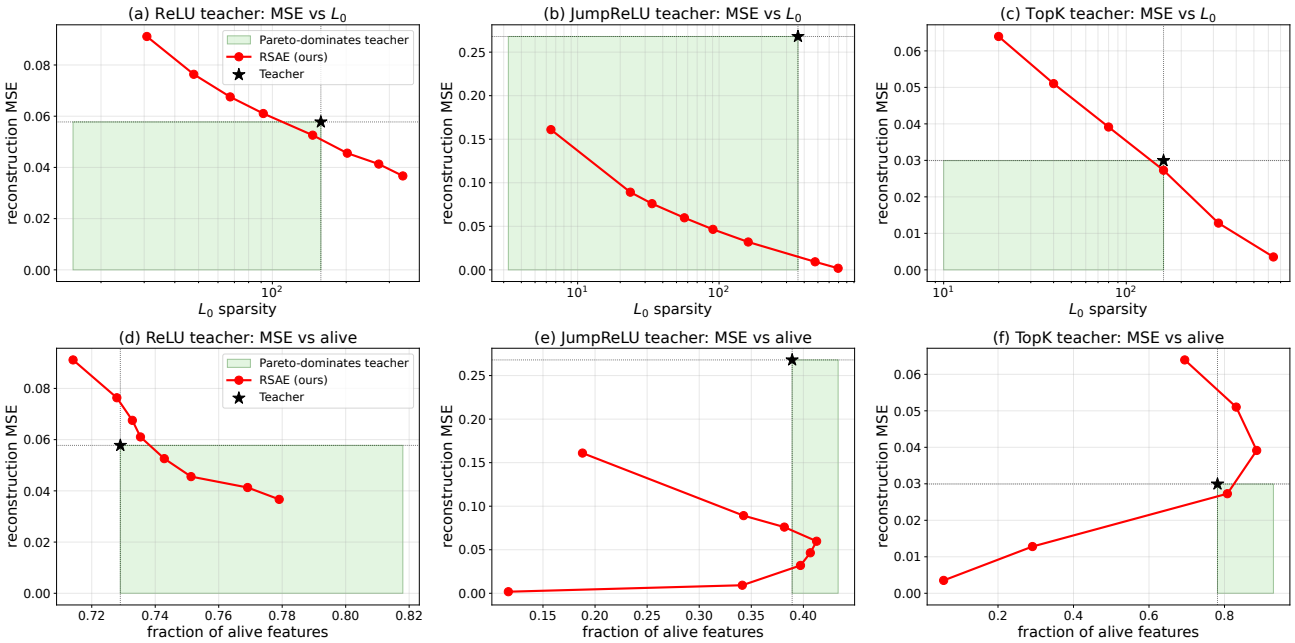

Figure 2. **Pareto fronts on Pythia-160m for all three baseline activation families.** Subfigures (a), (b), (c) plot MSE vs. ℓ_0, and subfigures (d), (e), (f) plot MSE vs. alive. Subfigures (a) and (d) use ReLU as the teacher, (b) and (e) use JumpReLU, and (c) and (f) use TopK. The black star is the teacher SAE; the red curve traces the RSAE Pareto front under a λ sweep. The green sweet-zone marks the strict-Pareto-domination region. The RSAE curve enters the sweet zone for every architecture at every sparsity level.

Table 3. **Wall-clock runtime** of the RSAE pipeline per model, averaged across baseline SAEs, measured on a single NVIDIA RTX 5090 (32 GB). The Init Procedure depends only on the host language model and is therefore identical across baselines (std = 0); for the Finetune Procedure and Total we report mean ± std across baselines. The detailed per-(model, baseline) breakdown is given in Table 6.

Model	Init Procedure (500 steps)	Finetune Procedure (2K steps)	Total
GPT-2 small	25 s	202 ± 120 s	227 ± 120 s
Pythia-160m	20 s	86 ± 10 s	106 ± 10 s
Gemma-2-2B	60 s	259 ± 30 s	319 ± 30 s

(Table 4). On the algorithm-side axis, sweeping λ traces an RSAE Pareto curve that enters the strict-domination "sweet zone" against every teacher in Figure 2, indicating that a small per-(model, teacher) tuning of λ is sufficient to obtain better MSE and a higher alive-feature fraction at lower ℓ_0 than the teacher. On the teacher-side axis, we re-run the pipeline against *all six* SAEBench ReLU trainers on Pythia-160m, which span a wide teacher sparsity range $\ell_0 \in [51, 683]$; the RSAE wins on reconstruction, alive, and LR on every trainer (6/6) and on ℓ_0 on half of the trainers (3/6). The performance gain is therefore not specific to a particular teacher sparsity but is consistent across the full range we tested.

Table 4. **Consistency of RSAE Pareto-domination across teacher sparsity.** We run our pipeline against *all six* SAEBench ReLU trainers on Pythia-160m, which differ only in their training ℓ_1 penalty and span teacher $\ell_0 \in [51, 683]$. "T" denotes the teacher SAE and "R" the RSAE. recon, alive, and LR are won on *every* trainer ($6/6$); ℓ_0 on $3/6$.

trainer	teacher L_1	recon T	recon R	ℓ_0 T	ℓ_0 R	alive T	alive R	LR T	LR R
0	0.012	0.028	**0.024**	683	**619**	74.6	**75.0**	97.6	**98.5**
1	0.015	0.035	**0.022**	494	503	74.4	**75.9**	98.2	**98.6**
2	0.020	0.043	**0.034**	313	365	73.9	**75.4**	96.6	**97.5**
3	0.030	0.058	**0.052**	158	**149**	72.9	**74.5**	95.7	**95.9**
4	0.040	0.067	**0.060**	97	99	71.7	**74.0**	95.6	**96.3**
5	0.060	0.082	**0.075**	51	**50**	68.7	**71.8**	92.4	**92.6**

Table 5. **Sparse-probing interpretability** on Pythia-160m ReLU teacher (trainer 3) and our RSAE distilled from it, on the full SAEBench panel of 8 binary-classification tasks. We report full-dictionary probe accuracy per dataset (all SAE features, no k-sparsity constraint); higher is better. Bold marks the better SAE per row; green marks rows where the RSAE matches or exceeds the teacher (all differences within probe noise).

Dataset	Teacher full ↑	RSAE full ↑	Δ
bias_in_bios_set1	95.28	**95.30**	+0.02
bias_in_bios_set2	**93.12**	93.04	−0.08
bias_in_bios_set3	90.94	**91.22**	+0.28
amazon_reviews	87.98	**88.00**	+0.02
amazon_sentiment	91.40	91.40	0.00
github-code	**96.64**	96.50	−0.14
ag_news	94.18	**94.25**	+0.07
europarl	99.96	99.96	0.00
mean	93.69	**93.71**	+0.02

5.3. Interpretability via Sparse Probing

The sparse-probing panel is a *preservation* check, not an interpretability-improvement claim. Table 5 reports full-dictionary linear-probe accuracy on the eight-task SAEBench panel for the Pythia-160m ReLU teacher (trainer 3) and the RSAE initialized from it. Task-level differences show no systematic direction and lie within the probe's seed-to-seed variation: the RSAE matches or exceeds the teacher on $6/8$ tasks and trails on the other two by at most 0.14 percentage points, with every per-task delta inside a ± 0.3 pp band and a panel mean of 93.71 vs. 93.69 ($\Delta = +0.02$). RSAE thus **preserves** the teacher's probe-measured interpretability rather than improving it; its gains lie in reconstruction fidelity and in the downstream behavioural metrics (ΔCE, loss recovered) of Tables 1 and 2—quantities that affect the faithfulness of SAE-based circuit analysis and activation patching.

6. Conclusion

We introduced the *Rational Sparse Autoencoder* (RSAE), an SAE whose encoder activation is a learnable rational function supported by approximation theory tailored to the shallow SAE encoder: trainable rational activations compactly approximate the fixed gates used by current SAE families, while scalar-output single-layer piecewise-affine encoders can require many more activated coordinates for some rational targets. The corresponding deep-network statements provide a complementary extension beyond the SAE architecture. When implemented as an upgrading strategy from existing pretrained SAEs across three host language models and three baseline activation families (ReLU, JumpReLU, TopK), our RSAE achieves better fidelity at comparable sparsity and strictly improves the baseline across both reconstruction-side metrics (MSE, ℓ_0, alive-feature fraction) and downstream-behaviour metrics (cross-entropy degradation, loss recovered), and these gains hold uniformly across the full range of baseline sparsity we tested without sacrificing feature-level interpretability under sparse probing. Because the upgrade adds only a handful of scalar parameters per autoencoder and runs in minutes on a single consumer GPU, any released ReLU, JumpReLU, or TopK SAE can in principle be replaced by its RSAE counterpart at negligible cost.

Limitation. Our evaluation covers three open-weight LLMs and three activations, so the gains at other frontier models remain unexplored. For instance, combining the rational activation with orthogonal architectural variants such as Gated, BatchTopK, and Matryoshka SAEs is an interesting direction. Our interpretability evidence is also limited to sparse probing. Feature dashboards, automated-interpretability scoring, feature-splitting/merging, and causal ablation/steering comparisons between teacher and RSAE features are left to future work; we do not claim RSAE changes the qualitative feature inventory.

Broader Impact. As a drop-in upgrade to any released SAE, the RSAE can strengthen interpretability-based safety auditing or lower the cost of misuse-relevant feature steering, depending on how it is deployed.

Acknowledgements

N.Y. and Y.Y. acknowledge support by the Defense Advanced Research Projects Agency (DARPA) under award HR00112590032. This research was, in part, funded by the U.S. Government by an agreement with Cornell University.

References

Beckermann, B. and Townsend, A. On the singular values of matrices with displacement structure. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1227–1248, 2017.

Bloom, J. Open source sparse autoencoders for all residual stream layers of GPT2-small. https://www.lesswrong.com/posts/f9EgfLSurAiqRJySD/open-source-sparse-autoencoders-for-all-residual-stream, 2024. LessWrong / HuggingFace release: jbloom/GPT2-Small-SAEs-Reformatted.

Boullé, N., Nakatsukasa, Y., and Townsend, A. Rational neural networks. *Advances in neural information processing systems*, 33:14243–14253, 2020.

Braun, D., Taylor, J., Goldowsky-Dill, N., and Sharkey, L. Identifying functionally important features with end-to-end sparse dictionary learning. *Advances in Neural Information Processing Systems*, 37:107286–107325, 2024.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N. L., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features.

Bussmann, B., Leask, P., and Nanda, N. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.

Bussmann, B., Nabeshima, N., Karvonen, A., and Nanda, N. Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv:2503.17547*, 2025.

Chen, Z., Chen, F., Lai, R., Zhang, X., and Lu, C.-T. Rational neural networks for approximating jump discontinuities of graph convolution operator. *arXiv preprint arXiv:1808.10073*, 2018.

Delfosse, Q., Schramowski, P., Mundt, M., Molina, A., and Kersting, K. Adaptive rational activations to boost deep reinforcement learning. *arXiv preprint arXiv:2102.09407*, 2021. URL https://arxiv.org/abs/2102.09407. Introduces the safe-Padé parameterisation.

Dunefsky, J., Chlenski, P., and Nanda, N. Transcoders find interpretable llm feature circuits. *Advances in Neural Information Processing Systems*, 37:24375–24410, 2024.

Gao, L., Dupre la Tour, T., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations*, volume 2025, pp. 26721–26754, 2025.

Huben, R., Cunningham, H., Smith, L., Ewart, A., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations*, volume 2024, pp. 7827–7845, 2024.

Karvonen, A., Rager, C., Lin, J., Tigges, C., Bloom, J. I., Chanin, D., Lau, Y.-T., Farrell, E., Mcdougall, C. S., Ayonrinde, K., et al. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability. In *International Conference on Machine Learning*, pp. 29223–29264. PMLR, 2025.

Laptev, D., Balagansky, N., Aksenov, Y., and Gavrilov, D. Analyze feature flow to enhance interpretation and steering in language models. In *International Conference on Machine Learning*, pp. 32593–32616. PMLR, 2025.

Molina, A., Schramowski, P., and Kersting, K. Padé activation units: End-to-end learning of flexible activation functions in deep networks. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1907.06732.

Newman, D. J. *Approximation with rational functions*. Number 41. American Mathematical Soc., 1979.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramar, J., Shah, R., and Nanda, N. Improving sparse decomposition of language model activations with gated sparse autoencoders. *Advances in Neural Information Processing Systems*, 37:775–818, 2024a.

Rajamanoharan, S., Lieberum, T., Sonnerat, N., Conmy, A., Varma, V., Kramár, J., and Nanda, N. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024b.

Taggart, G. M. ProLU: A nonlinearity for sparse autoencoders. https://www.lesswrong.com/posts/HEpufTdakGTTKgoYF/prolu-a-pareto-improvement-for-all-sparse-autoencoders, 2024. LessWrong post.

Tang, M. and Townsend, A. Rational neural networks have expressivity advantages. *arXiv preprint arXiv:2602.12390*, 2026.

Telgarsky, M. Benefits of depth in neural networks. In *Conference on learning theory*, pp. 1517–1539. PMLR, 2016.

Telgarsky, M. Neural networks and rational functions. In *International Conference on Machine Learning*, pp. 3387–3393. PMLR, 2017.

Trefethen, L. N. *Approximation theory and approximation practice, extended edition*. SIAM, 2019.

Trimmel, M., Zanfir, M., Hartley, R., and Sminchisescu, C. Era: Enhanced rational activations. In *European Conference on Computer Vision*, pp. 722–738. Springer, 2022.

A. Detailed Proofs

Lemma 1 (Zolotarev; rational approximation of sign)**.** *For every $\delta \in (0,1)$ and $n \geq 1$ there is a type-$(2n+1, 2n)$ rational $s_{n,\delta}$ such that $\sup_{x\in E_\delta} \big|\mathrm{sign}(x) - s_{n,\delta}(x)\big| \leq 4\exp\big(-\pi^2 n/\log(4/\delta)\big)$.*

Consequently, for every $0 < \varepsilon < 1$, there is a rational function of size $\mathcal{O}\big(\log(1/\varepsilon)\log(1/\delta)\big)$ that approximates sign *on E_δ to uniform error ε. For deep-layer networks, there is a constant-width rational network of depth $\mathcal{O}\big(\log\log(1/\varepsilon) + \log\log(1/\delta)\big)$ that approximates* sign *on E_δ to uniform error ε.*

Proof. The first part is an immediate result from the classical Zolotarev bound in (Beckermann & Townsend, 2017). Namely, for any $n \geq 1$,

$$\max_{s_{n,\delta}} \sup_{x\in E_\delta} \big|\mathrm{sign}(x) - s_{n,\delta}(x)\big| \leq 4\exp\big(-\pi^2 n/\log(4/\delta)\big).$$

To guarantee uniform error at most $\varepsilon \in (0,1)$ on $E_\delta = [-1,-\delta] \cup [\delta, 1]$, it suffices to choose $n = n_{\varepsilon,\delta}$ so that

$$4\exp\big(-\pi^2 n_{\varepsilon,\delta}/\log(4/\delta)\big) \leq \varepsilon.$$

Taking logarithms gives

$$\frac{\pi^2 n_{\varepsilon,\delta}}{\log(4/\delta)} \geq \log(4/\varepsilon),$$

or equivalently

$$n_{\varepsilon,\delta} \geq \frac{\log(4/\delta)\,\log(4/\varepsilon)}{\pi^2}.$$

Thus one admissible choice is

$$n_{\varepsilon,\delta} := \left\lceil \frac{\log(4/\delta)\,\log(4/\varepsilon)}{\pi^2} \right\rceil.$$

The rational $s_{n_{\varepsilon,\delta},\delta}$ has numerator and denominator degree $\mathcal{O}(n_{\varepsilon,\delta})$, and hence size

$$\mathcal{O}(n_{\varepsilon,\delta}) = \mathcal{O}\big(\log(1/\varepsilon)\log(1/\delta)\big).$$

This gives the direct scalar rational approximation.

For the deep-layer implementation, standard realizability results for rational networks show that a scalar rational with numerator and denominator degree at most N can be implemented by a constant-width rational network with depth $\mathcal{O}(\log N)$; see, for instance, the construction used by Boullé et al. (2020). Applying this to $s_{n_{\varepsilon,\delta},\delta}$ gives depth

$$\mathcal{O}\big(\log n_{\varepsilon,\delta}\big) = \mathcal{O}\big(\log\log(1/\varepsilon) + \log\log(1/\delta)\big).$$

□

Theorem 2 (Rational approximation of ReLU)**.** *For every $0 < \varepsilon < 1$, there exists a scalar rational function $R_\varepsilon : [-1,1] \to [-1,1]$ of size*

$$\mathcal{O}\Big(\log^2(1/\varepsilon)\Big),$$

such that

$$\sup_{x\in[-1,1]} \big|R_\varepsilon(x) - \mathrm{ReLU}(x)\big| \leq \varepsilon.$$

Consequently, the ReLU activation block can be replaced in either of two implementations. First, R_ε can be applied coordinatewise as a trainable rational activation, with scalar size $\mathcal{O}\big(\log^2(1/\varepsilon)\big)$. Second, the same scalar map can be realized by a constant-width deep rational network of internal depth

$$M_R = \mathcal{O}\Big(\log\log(1/\varepsilon)\Big).$$

Under either implementation, the resulting activation block $\mathcal{R}_R : [-1,1]^{d_{\mathrm{sae}}} \to \mathbb{R}^{d_{\mathrm{sae}}}$ satisfies

$$\sup_{\boldsymbol{h}\in[-1,1]^{d_{\mathrm{sae}}}} \big\|\mathcal{R}_R(\boldsymbol{h}) - \boldsymbol{z}_{\mathrm{R}}(\boldsymbol{h})\big\|_\infty \leq \varepsilon.$$

Proof. Let $\rho(t) := \mathrm{ReLU}(t) = \max\{t, 0\}$. The scalar construction is the one used in Lemma 1 of Boullé et al. (2020); we recall the argument to make clear how it follows from the Zolotarev sign approximation in Lemma 1. For an integer $m \geq 1$, take the Zolotarev sign function s_m of type $(3^m, 3^m - 1)$. By the composition property of Zolotarev sign functions, s_m can be written as a composition of m rational maps of type $(3, 2)$, so it is represented by a constant-width rational network with internal depth m.

As in the proof of Lemma 1 in Boullé et al. (2020), choose the gap parameter in the Zolotarev construction optimally. Then the product $t\,s_m(t)$ approximates $|t|$ on $[-1,1]$ with root-exponential accuracy: there is a universal constant $c > 0$ such that

$$\sup_{t\in[-1,1]} \big||t| - t\,s_m(t)\big| \leq 4\exp\big(-c\,3^{m/2}\big).$$

Using the identity

$$\rho(t) = \frac{|t| + t}{2},$$

define

$$r_m(t) := \frac{t\, s_m(t) + t}{2}.$$

It follows that

$$\sup_{t\in[-1,1]} |r_m(t) - \rho(t)| \le 2\exp\big(-c\,3^{m/2}\big).$$

Choose m so that this right-hand side is at most $\eta := \varepsilon/2$; equivalently,

$$m = \mathcal{O}\big(\log\log(1/\varepsilon)\big).$$

To keep the scalar approximant inside $[-1,1]$, set

$$R_\varepsilon(t) := \frac{r_m(t)}{1+\eta}.$$

Since $0 \le \rho(t) \le 1$ on $[-1,1]$ and $|r_m(t) - \rho(t)| \le \eta$, we have $-\eta \le r_m(t) \le 1+\eta$, hence $R_\varepsilon([-1,1]) \subset [-1,1]$. Moreover,

$$|R_\varepsilon(t) - \rho(t)| \le \frac{|r_m(t) - \rho(t)| + \eta|\rho(t)|}{1+\eta} \le 2\eta = \varepsilon.$$

This proves the scalar approximation guarantee. For the direct trainable rational-activation implementation, note that s_m has degree $\mathcal{O}(3^m)$, and therefore R_ε has numerator and denominator degree $\mathcal{O}(3^m)$. With the above choice of m,

$$3^m = \mathcal{O}\big(\log^2(1/\varepsilon)\big).$$

Thus R_ε is representable by $\mathcal{O}(\log^2(1/\varepsilon))$ coefficients, which is the stated size bound for the scalar rational function.

The same scalar map also admits a constant-width deep rational-network implementation of internal depth $m = \mathcal{O}(\log\log(1/\varepsilon))$, since s_m is a composition of m type-$(3,2)$ rational maps and the final affine rescaling does not change the asymptotic depth.

Now define the vector-valued activation block coordinate-wise by

$$\mathcal{R}_R(\boldsymbol{h})_i := R_\varepsilon(h_i), \qquad i = 1,\dots,d_{\text{sae}}.$$

Since $\boldsymbol{z}_{\text{R}}(\boldsymbol{h})_i = \rho(h_i)$, the scalar uniform bound gives

$$\big\|\mathcal{R}_R(\boldsymbol{h}) - \boldsymbol{z}_{\text{R}}(\boldsymbol{h})\big\|_\infty = \max_{1\le i\le d_{\text{sae}}} |R_\varepsilon(h_i) - \rho(h_i)| \le \varepsilon$$

for every $\boldsymbol{h} \in [-1,1]^{d_{\text{sae}}}$. □

Theorem 3 (Rational approximation of JumpReLU)**.** *Fix positive thresholds $\boldsymbol{\theta} \in (\mathbb{R}^+)^{d_{\text{sae}}}$ and a margin $\delta \in (0,1)$, and define*

$$\Omega_\delta := \{\boldsymbol{h} \in [-1,1]^{d_{\text{sae}}} : |h_i - \theta_i| \ge \delta \textit{ for every } i\}.$$

For every $0 < \varepsilon < 1$, there is a rational activation block $\mathcal{R}_J : [-1,1]^{d_{\text{sae}}} \to \mathbb{R}^{d_{\text{sae}}}$ whose scalar coordinate maps admit either of two implementations. First, each coordinate map can be implemented directly as a trainable scalar rational activation of size

$$\mathcal{O}\Big(\log(1/\varepsilon)\log(1/\delta)\Big),$$

with constants depending only on the fixed threshold scale. Second, each coordinate map can be realized by a constant-width deep rational network of internal depth

$$M_J \;=\; \mathcal{O}\Big(\log\log(1/\varepsilon) + \log\log(1/\delta)\Big)$$

and per-coordinate size $\mathcal{O}(M_J)$. Under either implementation, $\mathcal{R}_J$ satisfies

$$\sup_{\boldsymbol{h}\in\Omega_\delta} \big\|\mathcal{R}_J(\boldsymbol{h}) - \boldsymbol{z}_{\text{J}}(\boldsymbol{h})\big\|_\infty \;\le\; \varepsilon.$$

Proof. For each coordinate,

$$z_{\text{J},i}(\boldsymbol{h}) = h_i\, H(h_i - \theta_i) = h_i \cdot \frac{1 + \operatorname{sign}(h_i - \theta_i)}{2}.$$

Thus the problem reduces to approximating the scalar gate $H(h_i - \theta_i)$ on the margin-separated set $|h_i - \theta_i| \ge \delta$ and then multiplying by h_i.

Let

$$C_\theta := 1 + \|\boldsymbol{\theta}\|_\infty.$$

Since $|h_i| \le 1$, we have $|h_i - \theta_i| \le C_\theta$ for every coordinate. Apply Lemma 1 to the rescaled variable $u = (h_i - \theta_i)/C_\theta$, which ranges over $[-1, -\delta/C_\theta] \cup [\delta/C_\theta, 1]$ on Ω_δ. This gives a scalar rational function s such that

$$\sup_{|t|\in[\delta, C_\theta]} \Big|\operatorname{sign}(t) - s\Big(\frac{t}{C_\theta}\Big)\Big| \le \varepsilon.$$

Define

$$\widetilde{H}(t) := \frac{1 + s(t/C_\theta)}{2},$$

then for all $|t| \ge \delta$,

$$|\widetilde{H}(t) - H(t)| = \frac{1}{2}\Big|s\Big(\frac{t}{C_\theta}\Big) - \operatorname{sign}(t)\Big| \le \frac{\varepsilon}{2}.$$

By Lemma 1, this scalar gate is implemented by a constant-width rational network of depth

$$\mathcal{O}\big(\log\log(1/\varepsilon) + \log\log(C_\theta/\delta)\big) = \mathcal{O}\big(\log\log(1/\varepsilon) + \log\log(1/\delta)\big), \quad (8)$$

where the second equality uses that C_θ is an architectural constant.

For the direct trainable rational-activation implementation, keep the same scalar rational gate instead of factorizing it

into a deep constant-width composition. If $n_{\varepsilon,\delta}$ denotes the degree selected in Lemma 1 with margin δ/C_θ, then

$$n_{\varepsilon,\delta} = \mathcal{O}\Big(\log(C_\theta/\delta)\log(1/\varepsilon)\Big) = \mathcal{O}\Big(\log(1/\delta)\log(1/\varepsilon)\Big),$$

again treating C_θ as fixed. The rational gate $t \mapsto \widetilde{H}(t)$ therefore has numerator and denominator degree $\mathcal{O}(n_{\varepsilon,\delta})$.

We now further define the rational approximation

$$\widetilde{z}_i(\boldsymbol{h}) := h_i\,\widetilde{H}(h_i - \theta_i).$$

The product is realised exactly by the standard multiplication identity

$$xy = \frac{1}{4}\big[(x+y)^2 - (x-y)^2\big],$$

which is polynomial of degree 2 and therefore belongs to the rational class with constant additional depth. On Ω_δ we have

$$|\widetilde{z}_i(\boldsymbol{h}) - z_{\mathrm{J},i}(\boldsymbol{h})| = |h_i|\,|\widetilde{H}(h_i-\theta_i) - H(h_i-\theta_i)| \le \frac{\varepsilon}{2} \le \varepsilon.$$

So each coordinate is approximated uniformly to error at most ε.

Moreover, the scalar map $h_i \mapsto \widetilde{z}_i(\boldsymbol{h})$ is itself a univariate rational function of degree $\mathcal{O}(n_{\varepsilon,\delta})$: multiplying by h_i only increases the numerator degree by one.

For the direct scalar rational implementation, the numerator and denominator degrees are therefore $\mathcal{O}(n_{\varepsilon,\delta})$, so each coordinate map has size $\mathcal{O}\big(\log(1/\varepsilon)\log(1/\delta)\big)$. Define $\mathcal{R}_J(\boldsymbol{h})_i := \widetilde{z}_i(\boldsymbol{h})$ and apply the maps coordinatewise. The rational coefficients in the gate $\widetilde{H}$ are shared across coordinates; the coordinate dependence enters only through the affine shift $h_i \mapsto h_i - \theta_i$ and the final multiplication by h_i. Equivalently, since $\theta_i > 0$, one may use a shared prototype gate $r_{1/2}$ at threshold $1/2$ and apply it as $r_{1/2}(h_i/(2\theta_i))$, because

$$\frac{h_i}{2\theta_i} - \frac{1}{2} = \frac{h_i - \theta_i}{2\theta_i}.$$

The fixed threshold scale only changes constants in the margin, so the same asymptotic size bound applies. The scalar error bound gives

$$\sup_{\boldsymbol{h}\in\Omega_\delta} \|\mathcal{R}_J(\boldsymbol{h}) - \boldsymbol{z}_{\mathrm{J}}(\boldsymbol{h})\|_\infty \le \varepsilon,$$

which is the direct scalar-rational realization claimed in the theorem.

For the deep implementation, realize the same shared gate, either $\widetilde{H}$ or the equivalent prototype $r_{1/2}$, by the constant-width rational network above and use d_{sae} copies in parallel. The coordinate-dependent affine shifts or scalings and the final multiplication by h_i add only constant depth, so the vector-valued realization has width proportional to the number of coordinates and depth $\mathcal{O}(\log\log(1/\varepsilon) + \log\log(1/\delta))$. □

Theorem 4 (Rational approximation of supplied-threshold TopK gate)**.** *Fix $1 \le k < d_{\mathrm{sae}}$ and a margin $\delta \in (0,1)$ and define*

$$\Omega_\delta^{\mathrm{T}} := \{(\boldsymbol{h},\tau_k) \in [-1,1]^{d_{\mathrm{sae}}} \times [-1,1] : \\ h_{(k)} - \tau_k \ge \delta,\ \tau_k - h_{(k+1)} \ge \delta\},$$

where $h_{(1)} \ge \cdots \ge h_{(d_{\mathrm{sae}})}$ are the sorted pre-activations and τ_k is a supplied separator between the k-th and $(k+1)$-st order statistics, not the k-th activation itself. Suppose the scalar threshold $\tau_k \in [-1,1]$ is supplied together with each pre-activation vector. For every $0 < \varepsilon < 1$, there is a rational network $\mathcal{R}_T : [-1,1]^{d_{\mathrm{sae}}+1} \to \mathbb{R}^{d_{\mathrm{sae}}}$ whose scalar coordinate maps admit either of two implementations. First, each coordinate map can be implemented directly as a trainable scalar rational activation of size

$$\mathcal{O}\Big(\log(1/\varepsilon)\log(1/\delta)\Big).$$

Second, each coordinate map can be realized by a constant-width deep rational network of internal depth

$$M_T \;=\; \mathcal{O}\Big(\log\log(1/\varepsilon) + \log\log(1/\delta)\Big).$$

Under either implementation, $\mathcal{R}_T$ satisfies

$$\sup_{(\boldsymbol{h},\tau_k)\in\Omega_\delta^{\mathrm{T}}} \big\|\mathcal{R}_T(\boldsymbol{h},\tau_k) - \boldsymbol{z}_{\mathrm{T}}(\boldsymbol{h};\tau_k)\big\|_\infty \;\le\; \varepsilon.$$

Proof. Set $t_i := h_i - \tau_k$. For each coordinate,

$$z_{\mathrm{T},i}(\boldsymbol{h};\tau_k) = h_i\,H(h_i - \tau_k) = h_i \cdot \frac{1 + \mathrm{sign}(h_i - \tau_k)}{2}.$$

On $\Omega_\delta^{\mathrm{T}}$, the top-$k$ coordinates satisfy $h_i - \tau_k \ge \delta$ and the remaining coordinates satisfy $h_i - \tau_k \le -\delta$. Therefore $|t_i| \in [\delta, 2]$ for every coordinate, because $h_i, \tau_k \in [-1,1]$. Hence the JumpReLU proof applies with the learned threshold θ_i replaced by the supplied threshold τ_k and with the fixed scale 2. Applying Lemma 1 to $u = t_i/2$ gives a shared scalar rational function s such that

$$\sup_{|t|\in[\delta,2]} \Big|\mathrm{sign}(t) - s\Big(\frac{t}{2}\Big)\Big| \le \varepsilon.$$

Define

$$G(u) := \frac{1 + s(u)}{2},$$
$$\widetilde{H}(t) := \frac{1 + s(t/2)}{2},$$
$$\widetilde{z}_i(\boldsymbol{h};\tau_k) := h_i\,\widetilde{H}(h_i - \tau_k).$$

Then, for every $(\boldsymbol{h}, \tau_k) \in \Omega_\delta^{\mathrm{T}}$,

$$\begin{aligned}|\widetilde{z}_i(\boldsymbol{h};\tau_k) - z_{\mathrm{T},i}(\boldsymbol{h};\tau_k)| \leq& |h_i|\,|\widetilde{H}(h_i-\tau_k) - H(h_i-\tau_k)| \\ \leq& \frac{\varepsilon}{2} \\ \leq& \varepsilon.\end{aligned}$$

For the direct trainable rational-activation implementation, keep the shared gate G unfactored and apply it to the affine scalar $(h_i - \tau_k)/2$. By Lemma 1, the numerator and denominator degrees are $\mathcal{O}(\log(1/\varepsilon)\log(1/\delta))$; multiplying by h_i increases only the numerator degree by one. Thus each coordinate map has size $\mathcal{O}(\log(1/\varepsilon)\log(1/\delta))$, with the rational coefficients shared across coordinates.

For the deep implementation, realize the same shared gate G by a constant-width rational network of internal depth $\mathcal{O}(\log\log(1/\varepsilon) + \log\log(1/\delta))$. The affine difference and fixed scaling $(h_i - \tau_k)/2$, together with the final multiplication by h_i, add only constant depth. Applying this construction coordinatewise gives the stated block $\mathcal{R}_T$ and the uniform ℓ_∞ error bound. □

Theorem 5 (Lower bound for ReLU/JumpReLU/TopK networks)**.** *There exists a scalar rational target map $\mathcal{R}_\eta^\star : [-1,1] \to [0,1]$, realizable with $\mathcal{O}(1)$ rational parameters, such that the following hold.*

First, any scalar map $\mathcal{S} : [-1,1] \to [0,1]$ realized by a scalar-output ReLU/JumpReLU/supplied-threshold TopK network and satisfying

$$\|\mathcal{S} - \mathcal{R}_\eta^\star\|_{L^\infty([-1,1])} \leq \varepsilon$$

must satisfy $P = \Omega(\log(1/\varepsilon))$, where P is the number of trainable parameters.

Second, any scalar map $\mathcal{S} : [-1,1] \to [0,1]$ realized by the scalar-output version of the single-layer encoder architecture in (1), using ReLU, JumpReLU, or a supplied-threshold TopK gate with N activated coordinates, and satisfying

$$\|\mathcal{S} - \mathcal{R}_\eta^\star\|_{L^\infty([-1,1])} \leq \varepsilon$$

must satisfy $N = \Omega(\varepsilon^{-1/2})$.

Proof. Consider a rational function:

$$\mathcal{R}_\eta^\star(x) := \frac{\eta^2}{x^2+\eta^2}, \qquad x \in [-1,1],$$

with $\eta \in (0, 1/2)$ fixed and the interval $I_\eta = [-\eta/2, \eta/2]$. A direct calculation gives

$$(\mathcal{R}_\eta^\star)''(x) = \frac{2\eta^2\big(3x^2-\eta^2\big)}{(x^2+\eta^2)^3}.$$

For $|x| \leq \eta/2$, we have $3x^2 - \eta^2 \leq -\eta^2/4$ and $x^2 + \eta^2 \leq 5\eta^2/4$, hence

$$(\mathcal{R}_\eta^\star)''(x) \leq -\frac{\eta^4/2}{(5\eta^2/4)^3} = -\frac{32}{125\eta^2} =: -c_\eta.$$

Thus $\mathcal{R}_\eta^\star$ is uniformly concave on I_η.

Let $g : [-1,1] \to [0,1]$ be piecewise affine with R affine pieces and assume $\|g - \mathcal{R}_\eta^\star\|_{L^\infty([-1,1])} \leq \varepsilon$. Since $|I_\eta| = \eta$, one affine piece of g contains a subinterval $J = [a,b] \subset I_\eta$ of length

$$\ell := |J| \geq \frac{\eta}{R}.$$

Let s_J be the secant line of $\mathcal{R}_\eta^\star$ on J, and let $m = (a+b)/2$ be the midpoint of J. Because $(\mathcal{R}_\eta^\star)'' \leq -c_\eta$ on J, Taylor's theorem at the midpoint gives

$$\mathcal{R}_\eta^\star(m) - s_J(m) \geq \frac{c_\eta \ell^2}{8}.$$

Now write the affine restriction of g on J as

$$g|_J = s_J + q,$$

where q is affine. Since q is affine,

$$q(m) = \frac{q(a)+q(b)}{2}.$$

Also, because $s_J(a) = \mathcal{R}_\eta^\star(a)$ and $s_J(b) = \mathcal{R}_\eta^\star(b)$, the uniform error bound at the endpoints implies

$$|q(a)| \leq \varepsilon, \qquad |q(b)| \leq \varepsilon,$$

and therefore $|q(m)| \leq \varepsilon$. Evaluating the error at the midpoint, we obtain

$$\varepsilon \geq \big|\mathcal{R}_\eta^\star(m) - g(m)\big| = \big|\mathcal{R}_\eta^\star(m) - s_J(m) - q(m)\big| \geq \frac{c_\eta \ell^2}{8} - \varepsilon.$$

Hence

$$\varepsilon \geq \frac{c_\eta \ell^2}{16} \geq \frac{c_\eta \eta^2}{16R^2},$$

which yields

$$R \geq c'_\eta\, \varepsilon^{-1/2}$$

for a constant $c'_\eta > 0$ depending only on η.

This already gives the single-layer encoder lower bound. Indeed, a scalar-output version of the SAE encoder in (1) has the form

$$g(x) = a_0 + \sum_{j=1}^{N} a_j\, \psi_j(w_j x + b_j),$$

where each ψ_j is a ReLU, JumpReLU, or supplied-threshold TopK gate. Each supplied-threshold TopK summand is interpreted with its supplied threshold fixed along this scalar

restriction, so every summand is affine except at at most one breakpoint. Hence the union of all breakpoints has size at most N and the scalar map has at most $N+1$ affine pieces. Combining this with the piece-count lower bound above gives

$$c'_\eta \, \varepsilon^{-1/2} \le R \le N+1,$$

and hence $N = \Omega(\varepsilon^{-1/2})$.

For the arbitrary-depth statement, if $\mathcal{S}$ is a scalar-valued ReLU network with depth L and M hidden units, then the network has at most $(M/L)^L$ breakpoints, and the same conclusion holds for JumpReLU and supplied-threshold TopK networks (see, e.g., Telgarsky (2016)). Therefore, in either case,

$$c'_\eta \, \varepsilon^{-1/2} \le R \le (M/L)^L.$$

Taking logarithms of both sides of $c'_\eta \, \varepsilon^{-1/2} \le (M/L)^L$ gives

$$\log(c'_\eta) + \tfrac{1}{2}\log(1/\varepsilon) \;\le\; L\log(M/L).$$

Since each layer has at least 2 neurons we have $M \ge 2L$, so $M/L \ge 2 > 1$. Using $\log x < x$ for all $x > 1$ yields $\log(M/L) < M/L$, and therefore

$$L\log(M/L) \;<\; L \cdot \frac{M}{L} \;=\; M.$$

Combining the two inequalities,

$$M \;>\; \log(c'_\eta) + \tfrac{1}{2}\log(1/\varepsilon),$$

so $M = \Omega(\log(1/\varepsilon))$. Since each hidden unit contributes at least one bias parameter, the total trainable parameter count satisfies $P \ge M$, and therefore $P = \Omega(\log(1/\varepsilon))$. □

B. Detailed Empirical Results

B.1. Implementation Details and Experimental Setup

Models and hook points. We evaluate on three open-weight language models. For **GPT-2 small** we hook the residual stream at layer 6 (`blocks.6.hook_resid_post`, $d_{\text{in}} = 768$); for **Pythia-160m-deduped** at layer 8 (`blocks.8.hook_resid_post`, $d_{\text{in}} = 768$); and for **Gemma-2-2B** at layer 12 (`blocks.12.hook_resid_post`, $d_{\text{in}} = 2304$). All language-model forwards are run in `bf16`.

Teacher SAE checkpoints. Teacher SAEs are pre-trained, publicly released checkpoints. For GPT-2 small the ReLU teacher is `gpt2-small-res-jb` (Bloom, 2024) ($d_{\text{sae}} = 24{,}576$) and the TopK teacher is the OpenAI-v5 `gpt2-small-resid-post-v5-32k` (Gao et al., 2025) ($d_{\text{sae}} = 32{,}768$, with layer-norm pre-processing of the residual stream). For Pythia-160m the ReLU teacher is `adamkarvonen/saebench_pythia-160m-deduped_width-2pow14_date-0108` ($d_{\text{sae}} = 16{,}384$); JumpReLU and TopK teachers are taken from the matching SAEBench (Karvonen et al., 2025) releases at the same width. For Gemma-2-2B all three teacher families are taken from the corresponding SAEBench releases at $d_{\text{sae}} = 4{,}096$.

RSAE activation and parameter init. Every RSAE uses the standard-Padé rational activation (3) with degree $(p, q) = (3, 2)$ for ReLU teachers and $(9, 8)$ for JumpReLU and TopK teachers (chosen from the synthetic ablation of §5.1; see Figure 3). Coefficients $(\boldsymbol{a}, \boldsymbol{b})$ are initialised from the converged relaxed Remez fit of the teacher activation (§4); per-feature scales $\log C_{\text{in}}$ and $\log C_{\text{out}}$ are initialised to zero and learned. Encoder/decoder weights and biases are inherited verbatim from the teacher checkpoint at $t = 0$ so that the RSAE approximately matches the teacher before fine-tuning. To preserve the standard SAE non-negativity convention $\boldsymbol{z} \ge \mathbf{0}$, we apply a $\max(0, \phi(\boldsymbol{h}))$ clamp on the rational output element-wise.

Activation buffer and calibration. Activations are streamed from a Pile / OpenWebText mixture using TransformerLens hooks; we collect a calibration buffer of ∼2M tokens that is used both to fit the activation distillation step (6) and to provide statistics for the per-feature pre-activation scale used in (3). A separate held-out evaluation buffer of ∼10K tokens (disjoint from the calibration buffer) is used for all reported metrics.

Init procedure (500 steps). The empirical activation distillation step fits $(\boldsymbol{a}, \boldsymbol{b}, C_{\text{in}}, C_{\text{out}})$ to the teacher activation ϕ^{teacher} on calibration-buffer pre-activations, with all other RSAE parameters frozen. We use Adam with learning rate 10^{-3}, batch size $1{,}024$ tokens, and run for 500 steps.

Fine-tune procedure (2,000 steps). The full RSAE is then fine-tuned jointly on the ℓ_1-regularised reconstruction objective (7) for $2{,}000$ Adam steps with learning rate 5×10^{-4} and a cosine decay to 0, batch size $4{,}096$ tokens, and gradient clipping at norm 1.0. SAE training itself runs in `fp32`. The sparsity coefficient λ is swept on a small grid per (model, teacher) pair to trace the Pareto front of Figure 2; the values reported in Tables 1–2 use the λ that maximises the number of strict per-axis wins over the teacher.

Hardware. All RSAE training reported in Table 3 is run on a single NVIDIA RTX 5090 (32 GB). End-to-end wall-clock per (model, baseline) is dominated by the fine-tuning procedure.

Algorithm 1 Rational Sparse Autoencoder (RSAE) construction.

Require: Pre-trained baseline SAE $\{\widetilde{\boldsymbol{W}_{\text{enc}}}, \widetilde{\boldsymbol{b}_{\text{enc}}}, \widetilde{\boldsymbol{W}_{\text{dec}}}, \widetilde{\boldsymbol{b}_{\text{dec}}}\}$ with activation $\phi^{\text{teacher}} \in \{\text{ReLU}, \text{JumpReLU}_\theta, \text{TopK}_k\}$; rational type (p, q); calibration buffer $\{\boldsymbol{x}_n\}_{n=1}^{N_{\text{cal}}}$; dense grid size N; sparsity coefficient λ.

Step 1 — RSAE Initialization Procedure.
Build dense grid $\{(t_\ell, y_\ell)\}_{\ell=1}^{N} \subset [-1, 1]$ with $y_\ell = \phi^{\text{teacher}}(t_\ell)$.
Fit rational coefficients $(\boldsymbol{a}^*, \boldsymbol{b}^*)$ to the teacher activation by the relaxed Remez exchange targeting (5) (Appendix B.2); fall back to the L^2 or smoothed-L^∞ surrogate if Remez is numerically unstable. (Optionally distill onto the safe-Padé family by least squares on $\{t_\ell\}_{\ell=1}^{N}$.)
Compute teacher pre-activations $\boldsymbol{h}_n = \widetilde{\boldsymbol{W}_{\text{enc}}}\,(\boldsymbol{x}_n - \widetilde{\boldsymbol{b}_{\text{dec}}}) + \widetilde{\boldsymbol{b}_{\text{enc}}}$ for $n = 1, \ldots, N_{\text{cal}}$.
Adapt the rational activation to the teacher by minimising (6) over $(\boldsymbol{a}, \boldsymbol{b}, C_{\text{in}}, C_{\text{out}})$ with $(\boldsymbol{a}, \boldsymbol{b})$ initialised from $(\boldsymbol{a}^*, \boldsymbol{b}^*)$, yielding $(\widetilde{\boldsymbol{a}}, \widetilde{\boldsymbol{b}}, \widetilde{C}_{\text{in}}, \widetilde{C}_{\text{out}})$.
Step 2 — RSAE Fine-Tuning Procedure.
Inherit teacher weights $\{\boldsymbol{W}_{\text{enc}}, \boldsymbol{b}_{\text{enc}}, \boldsymbol{W}_{\text{dec}}, \boldsymbol{b}_{\text{dec}}\} \leftarrow \{\widetilde{\boldsymbol{W}_{\text{enc}}}, \widetilde{\boldsymbol{b}_{\text{enc}}}, \widetilde{\boldsymbol{W}_{\text{dec}}}, \widetilde{\boldsymbol{b}_{\text{dec}}}\}$ and set $(\boldsymbol{a}, \boldsymbol{b}, C_{\text{in}}, C_{\text{out}}) \leftarrow (\widetilde{\boldsymbol{a}}, \widetilde{\boldsymbol{b}}, \widetilde{C}_{\text{in}}, \widetilde{C}_{\text{out}})$.
Collect $\Theta \leftarrow \{\boldsymbol{W}_{\text{enc}}, \boldsymbol{b}_{\text{enc}}, \boldsymbol{W}_{\text{dec}}, \boldsymbol{b}_{\text{dec}}, \boldsymbol{a}, \boldsymbol{b}, C_{\text{in}}, C_{\text{out}}\}$.
repeat
  Sample mini-batch $\boldsymbol{x} \sim \mathcal{D}$; compute $\boldsymbol{h} = \boldsymbol{W}_{\text{enc}}\,(\boldsymbol{x} - \boldsymbol{b}_{\text{dec}}) + \boldsymbol{b}_{\text{enc}}$, $\boldsymbol{z} = \phi(\boldsymbol{h})$ via (3), and $\hat{\boldsymbol{x}} = \boldsymbol{W}_{\text{dec}}\,\boldsymbol{z} + \boldsymbol{b}_{\text{dec}}$.
  Take an Adam step on Θ minimising the integrand of (7), with the unit-norm-row constraint on $\boldsymbol{W}_{\text{dec}}$ enforced by gradient projection and renormalisation.
until convergence (joint Adam fine-tuning of (7)).

Ensure: RSAE with parameters Θ.

Detailed wall-clock runtime (Table 6). Table 6 reports the per-(model, baseline) wall-clock breakdown that underlies the mean ± std summary of Table 3 in the main text. Init- and Finetune-procedure timings are dominated by the language-model forward pass through the host network at each training step.

Table 6. **Detailed wall-clock runtime** of the RSAE pipeline per (model, baseline SAE), measured on a single NVIDIA RTX 5090 (32 GB).

Model	Baseline SAEs	Init Procedure (500 steps)	Finetune Procedure (2K steps)	Total
GPT-2 small	ReLU	25 s	117 s	142 s
	TopK	25 s	287 s	312 s
Pythia-160m	ReLU	20 s	80 s	100 s
	JumpReLU	20 s	98 s	118 s
	TopK	20 s	81 s	101 s
Gemma-2-2B	ReLU	60 s	240 s	300 s
	JumpReLU	60 s	294 s	354 s
	TopK	60 s	243 s	303 s

Evaluation protocol. Reconstruction MSE, ℓ_0 (at the $|z| > 10^{-6}$ threshold), and the alive-feature fraction (a latent is "alive" if it fires on at least one held-out token) are evaluated on the ∼10K-token held-out buffer. Cross-entropy degradation ΔCE and the loss-recovered fraction LR are computed over 128 Pile / OpenWebText sequences of length 128 tokens, by routing the residual stream through $\hat{\boldsymbol{x}}$ at the SAE's host layer and reading the language model's next-token cross-entropy at every position; CE_{zero} is the cross-entropy obtained when the residual stream at the host layer is replaced by the zero vector.

Sparse-probing setup (Table 5). We use the SAEBench probing suite of 8 binary-classification tasks. Per dataset we train an ℓ_2-regularised logistic regression on the full SAE feature vector (no k-sparsity constraint) using SAEBench's default split, and report mean test accuracy across the dataset's evaluation seeds.

B.2. Relaxed Remez Exchange Details

This appendix expands on the relaxed Remez exchange used in Step 1 of §4 to solve the min–max objective (5). Writing $r_{(\boldsymbol{a},\boldsymbol{b})}(t) = P(t)/Q(t)$ in the standard-Padé form $P(t) = \sum_{i=0}^{p} a_i\, t^i$, $Q(t) = 1 + \sum_{j=1}^{q} b_j\, t^j$, and setting $K = p+q+1$, the r-th outer iteration carries an amplitude estimate E_r together with $K+2$ alternation nodes $\{t_d^{(r)}\}_{d=0}^{K+1} \subset [-1, 1]$. Replacing the unknown amplitude on the right-hand side of the Chebyshev equioscillation condition with E_r yields the $K+2$ *linear* equations

$$P\big(t_d^{(r)}\big) \;-\; \big[\, y_d^{(r)} - (-1)^d\, E_r \,\big]\,\big(\, Q\big(t_d^{(r)}\big) - 1 \,\big) \;-\; (-1)^d\, E_{r+1} \;=\; y_d^{(r)}, \quad d = 0, \ldots, K+1, \tag{9}$$

with $y_d^{(r)} = \phi^{\text{teacher}}\big(t_d^{(r)}\big)$, which we solve in the least-squares sense for $(\boldsymbol{a}, \boldsymbol{b}, E_{r+1})$; the nodes are then relocated to the $K+2$ largest local extrema of $|r_{(\boldsymbol{a},\boldsymbol{b})}(t_\ell) - y_\ell|$ on the dense grid $\{t_\ell\}_{\ell=1}^{N}$. The relaxation, using E_r in place of the unknown E_{r+1} on the right-hand side, reduces each inner step to a single linear solve and makes the algorithm robust on discontinuous targets such as JumpReLU_θ. At convergence ($|E_{r+1} - E_r| < \varepsilon$) the residual equioscillates and

$r_{(\boldsymbol{a},\boldsymbol{b})}$ attains the best L^∞ rational approximation of type (p, q) on $[-1, 1]$ by Chebyshev's characterisation (Trefethen, 2019). Because Remez fits the standard-Padé denominator Q rather than the safe-Padé form of (3), we then distill the converged Remez rational onto the safe-Padé family by least squares on $\{t_\ell\}_{\ell=1}^N$.

B.3. Rational Fitting on Synthetic Data

This appendix records the full numerical comparison of the four rational-fitting procedures of §4 across the activation primitives used by current SAE families. We sweep superdiagonal types $(p, q) \in \{(3, 2), (5, 4), \ldots, (19, 18)\}$ (extended to $(29, 28)$ for the safe-Padé baselines), evaluate every fit on a uniform dense grid of $N = 4001$ points over $[-1, 1]$, and report the best L^2 mean-squared error attained by each fitter. The procedures compared are: (i) standard-Padé Remez, the relaxed exchange of Chen et al. (2018) that targets the L^∞ minimax objective (5) in the family $Q(t) = 1 + \sum_j \phi_j t^j$ with signed ϕ_j; (ii) safe-Padé Remez via warm-start (Route A), in which the converged standard-Padé Remez coefficients are distilled onto the pole-free family $Q(t) = 1 + \sum_j |b_j||t|^j$ of (3) by least-squares fitting on $\{t_\ell\}_{\ell=1}^N$; (iii) L^2 safe-Padé fit, minimising $\frac{1}{N}\sum_\ell (r_{(\boldsymbol{a},\boldsymbol{b})}(t_\ell) - y_\ell)^2$ over the safe-Padé parameters via Adam with cosine learning-rate decay; and (iv) L^∞ safe-Padé fit, minimising the smoothed-supremum log-sum-exp surrogate of $\max_\ell |r_{(\boldsymbol{a},\boldsymbol{b})}(t_\ell) - y_\ell|$ via Adam. We refer to the safe-Padé form distilled from the standard-Padé Remez fit as Route A.

Table 7. **Best-MSE rational fits across procedures.** Each cell reports the (best degree, L^2 MSE on a uniform $N = 4001$ grid over $[-1, 1]$). Standard-Padé Remez minimises L^∞ but is reported here under the same L^2 MSE for fair comparison; safe-Padé Remez (Route A) is the standard-Padé Remez fit distilled onto the safe-Padé family of (3). "–" marks targets we did not run with the Route-A pipeline. **Bold** marks the best fitter per target.

target	Std-Padé Remez	Safe-Padé Remez	L^2 safe-Padé	L^∞ safe-Padé
ReLU	**(15,14)** 3.8×10^{-7}	(17,16) 9.6×10^{-7}	(15,14) 3.8×10^{-6}	(11,10) 4.0×10^{-6}
$\mathrm{JumpReLU}_{\theta=0.1}$	**(19,18)** 2.4×10^{-6}	(19,18) 1.4×10^{-5}	(29,28) 7.5×10^{-5}	(19,18) 7.6×10^{-4}
$\mathrm{JumpReLU}_{\theta=0.2}$	**(11,10)** 7.1×10^{-5}	–	(23,22) 3.8×10^{-4}	(17,16) 2.7×10^{-3}
$\mathrm{JumpReLU}_{\theta=0.3}$	(5,4) 9.8×10^{-4}	–	**(29,28)** 8.9×10^{-4}	(13,12) 7.5×10^{-3}
$\mathrm{JumpReLU}_{\theta=0.4}$	**(9,8)** 9.7×10^{-4}	–	(29,28) 1.6×10^{-3}	(9,8) 1.6×10^{-2}
$\mathrm{JumpReLU}_{\theta=0.5}$	(5,4) 4.3×10^{-3}	–	**(29,28)** 2.2×10^{-3}	(3,2) 1.9×10^{-2}

Standard-Padé Remez dominates on smooth and small-jump targets. On four of the six rows of Table 7 (ReLU and $\mathrm{JumpReLU}_\theta$ for $\theta \in \{0.1, 0.2, 0.4\}$), the relaxed Remez exchange in the standard-Padé family attains the best MSE by a margin of roughly 2–30× over the next-best independent fitter (and 1.6–5.8× over Route-A safe-Padé Remez, which is itself distilled from this Remez solution). This is the expected behaviour of a Chebyshev best-rational solution: by Trefethen (2019)'s characterisation, the equioscillation property of Remez's converged residual is necessary and sufficient for L^∞ optimality, and on smooth or small-jump targets the resulting L^∞ error tracks closely the L^2 MSE because the residual oscillation amplitude itself controls both norms. In particular, on ReLU at type $(15, 14)$ Remez reaches MSE 3.8×10^{-7}, well below the noise floor of any downstream SAE reconstruction objective, and on $\mathrm{JumpReLU}_{0.1}$ at type $(19, 18)$ MSE 2.4×10^{-6}, with sup-error tracking the information-theoretic half-jump floor $\sup|r - f| \geq \theta/2$.

Safe-Padé Remez (Route A) recovers most of the standard-Padé quality while staying pole-free. The safe-Padé family $Q(t) = 1 + \sum_j |b_j||t|^j$ is a strict subset of the standard-Padé family, so it cannot exceed standard-Padé Remez in approximation power; the relevant question is how much accuracy the pole-safety constraint costs. Route A warm-starts a safe-Padé fit from the converged standard-Padé Remez coefficients via $a_i \leftarrow \psi_i$, $b_j \leftarrow |\phi_j|$ and refines under L^2 on the same dense grid. The result on the two targets we ran end-to-end (Table 7) is a 2.5× MSE gap to standard-Padé Remez on ReLU (9.6×10^{-7} vs. 3.8×10^{-7}) and a ∼6× gap on $\mathrm{JumpReLU}_{0.1}$ (1.4×10^{-5} vs. 2.4×10^{-6}). Both are still 4–50× tighter than the direct L^2 and L^∞ safe-Padé fits at the same family, indicating that the warm-start from the Remez minimax solution materially helps the optimisation.

L^2 safe-Padé wins at large jump sizes due to Remez's numerical instability. At $\theta = 0.3$ and $\theta = 0.5$, standard-Padé Remez underperforms the L^2 safe-Padé fit even at lower degree (Remez's best is $(5, 4)$ vs. L^2's best at $(29, 28)$). This is not a deficiency of the Remez objective itself but of the linearised exchange: when the target's discontinuity is large compared to its derivative scale, the linear system (9) becomes severely ill-conditioned at high (p, q), causing the exchange to oscillate or diverge. The L^2 safe-Padé fit, by contrast, is solved by Adam on a smooth non-convex landscape and remains numerically stable across all degrees we tested. The MSE gap is small in absolute terms (compare 9.8×10^{-4} vs. 8.9×10^{-4} at $\theta = 0.3$) and reflects the fact that at large θ both methods are dominated by the half-jump floor anyway.

L^∞ safe-Padé is consistently weakest in L^2 MSE, by design. Across every row, the smoothed-supremum fit attains the worst L^2 MSE among the four procedures. This is expected: the log-sum-exp surrogate concentrates training pressure on the largest-error points, sacrificing L^2 MSE for tighter L^∞ control. We include L^∞ safe-Padé in the table for completeness, since it remains the right choice in any application that cares about worst-case error rather than average error.

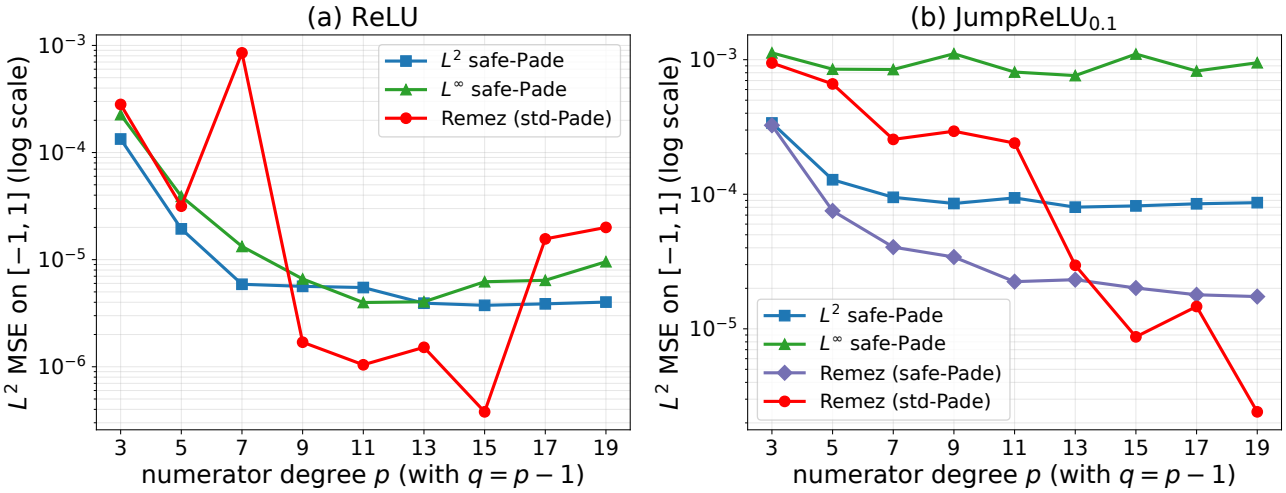

Figure 3. L^2 **MSE versus rational degree on** $[-1, 1]$**.** Each curve traces the best L^2 MSE attained by one fitter as the type (p, q) is swept across $\{(3, 2), (5, 4), \ldots, (19, 18)\}$. **(a)** On ReLU, standard-Padé Remez (red) decays near-exponentially with degree until type $(15, 14)$, beyond which numerical conditioning of the linearised exchange dominates and the curve flattens. The pole-free safe-Padé L^2 (blue) and L^∞ (green) fits saturate earlier, near $\mathrm{MSE} \approx 4 \times 10^{-6}$, due to the strictly smaller capacity of the safe family. **(b)** On $\mathrm{JumpReLU}_{0.1}$, the same pattern holds and the Route-A safe-Padé Remez fit (purple) tracks the standard-Padé Remez curve at a fixed $\sim 7\times$ offset, the safe-Padé family ceiling.