Sparse Attention as Compact Kernel Regression

Saul Santos[1,2,5], **Nuno Gonçalves**[1,2,3], **Daniel C. McNamee**[4],
Marcos Treviso[1,2,5], **André F. T. Martins**[1,2,5]

[1]Instituto Superior Técnico, Universidade de Lisboa [2]Instituto de Telecomunicações
[3]Carnegie Mellon University [4]Champalimaud Research [5]Gandara AI

Abstract

Recent work has revealed a link between self-attention mechanisms in transformers and test-time kernel regression via the Nadaraya-Watson estimator, with standard softmax attention corresponding to a Gaussian kernel. However, a kernel-theoretic understanding of *sparse* attention mechanisms is currently missing. In this paper, we establish a formal correspondence between sparse attention and *compact* (bounded support) kernels. We show that normalized ReLU and sparsemax attention arise from Epanechnikov kernel regression under fixed and adaptive normalizations, respectively. More generally, we demonstrate that widely used kernels in nonparametric density estimation—including Epanechnikov, biweight, and triweight—correspond to α-entmax attention with $\alpha = 1 + \frac{1}{n}$ for $n \in \mathbb{N}$, while the softmax/Gaussian relationship emerges in the limit $n \to \infty$. This unified perspective explains how sparsity naturally emerges from kernel design and provides principled alternatives to heuristic top-k attention and other associative memory mechanisms. Experiments with a kernel-regression-based variant of transformers—Memory Mosaics—show that kernel-based sparse attention achieves competitive performance on language modeling, in-context learning, and length generalization tasks, offering a principled framework for designing attention mechanisms.[1].

1 Introduction

Self-attention is a central component of transformer architectures, enabling models to capture long-range dependencies within sequences [38]. A growing body of work has established connections between attention-based in-context learning and variants of associative memories, including Hopfield networks [25], fast weight programmers [28], mesa-optimization [41], and various forms of test-time regression and memory-based computation [42, 4, 31, 3, *inter alia*]. A key observation in several of these works is that standard softmax-based attention can be interpreted as *Gaussian kernel regression* on the key-value cache—this has motivated new transformer-related architectures such as *Memory Mosaics* [49, 48]. However, *is the Gaussian kernel—and softmax attention—the best design choice*? It has been pointed out that the dense weights produced by softmax attention mechanisms may cause attention dispersion [40]—a phenomenon where many weak contributions from irrelevant positions dilute the influence of truly relevant ones. Evidence from brain research (see §5 for details) suggests that neocortical memory retrieval is mediated by a relatively small set of well-separated memories, rather than weakly integrating over an increasingly large number of memories [37, 9].

The observations above provide strong motivation for *sparse* attention alternatives, such as top-k softmax [14], sparsemax [19], and α-entmax [22]. The last two provide continuous, fully differentiable mappings from scores to attention weights while maintaining exact sparsity (Figure 1), enabling transformers to learn attention sparsity adaptively, rather than relying on a fixed truncation level.

[1]Correspondence to saul.r.santos@tecnico.ulisboa.pt

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

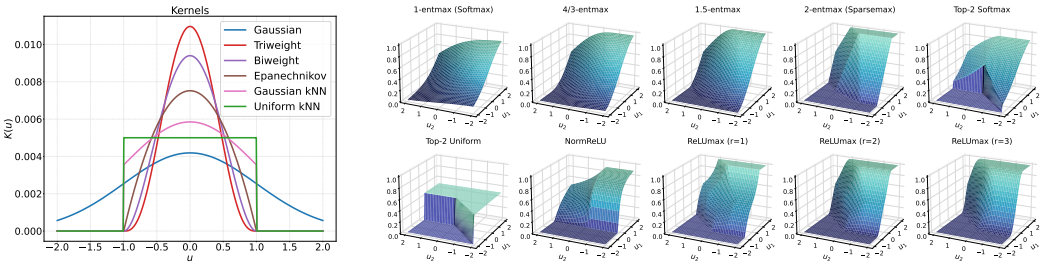


Figure 1: **Comparison between classical kernel functions and their corresponding attention activations.** The left plot shows normalized one-dimensional kernel functions commonly used in kernel regression evaluated on an input $\boldsymbol{u} \in \mathbb{R}^n$. The right panel shows different attention transformations applied to a 3D score vector $\boldsymbol{u} = [0, u_1, u_2]$. Each transformation allows an interpretation with corresponding kernels shown on the left: softmax ($\alpha = 1$) corresponds to the Gaussian kernel; normalized ReLU, ReLUmax, and sparsemax ($\alpha = 2$) are derived from Epanechnikov kernel variants ($r = 1$); The 4/3- and 1.5-entmax transformations correspond to the biweight ($r = 2$) and triweight ($r = 3$) kernels, respectively, while 1- and 2-ReLUmax follow analogous patterns. top-k uniform mirrors the top-k uniform kernel, while top-k softmax can be viewed as a truncated Gaussian kernel.

However, unlike the softmax case, a theoretical test-time regression understanding of sparse attention is currently missing.

Our work aims to fill that gap by providing a unified kernel regression perspective on both dense and sparse attention mechanisms. We show that a range of attention transformations—including softmax, sparsemax, α-entmax, top-k uniform, top-k softmax, normalized ReLU, and a new transformation, ReLUmax—can be interpreted as Nadaraya-Watson estimators with different kernel functions. This viewpoint not only explains how sparsity naturally emerges from kernel design but also offers a principled framework for developing attention mechanisms with controlled sparsity. Our main contributions are: [2]

1. **We establish that sparsemax attention corresponds to auto-normalized Epanechnikov kernel regression** through an adaptive bandwidth, providing a rigorous link between sparse attention and compact support kernels.
2. **More generally, we show that α-entmax attention with $\alpha > 1$ corresponds to kernel regression with compact support kernels** of the form $K(\boldsymbol{u}) \propto [1 - \|\boldsymbol{u}\|^2]_+^r$ where $r = (\alpha - 1)^{-1}$, encompassing widely used kernels such as Epanechnikov, biweight, and triweight.
3. **We unify top-k and fixed-normalization sparse attention within the same kernel regression framework**: top-k softmax corresponds to Gaussian kernel k-nearest-neighbor (kNN) regression, normalized ReLU corresponds to fixed-bandwidth Epanechnikov regression.
4. **We propose a new transformation, r-ReLUmax**, which corresponds to a max-anchored polynomial rectified variant.
5. **We conduct experiments demonstrating the benefits of compact kernels in Memory Mosaics**, a strong transformer-like architecture tied to kernel regression, achieving superior performance on simple language modeling, in-context learning, and length generalization tasks.

2 Background

2.1 Sparsemax and α-entmax

A generalization of softmax is given by the α-entmax transformation [22], which has the form

$$\alpha\text{-entmax}(\boldsymbol{z}) = [(\alpha - 1)\boldsymbol{z} - \tau \mathbf{1}]_+^{\frac{1}{\alpha - 1}}, \tag{1}$$

where $[x]_+ = \max(0, x)$ is the ReLU operator and τ is chosen such that $\sum_i p_i = 1$. This transformation arises as the solution of the optimization problem $\alpha\text{-entmax}(\boldsymbol{z}) := \arg\max_{\boldsymbol{p} \in \triangle_n} \boldsymbol{z}^\top \boldsymbol{p} +$

[2] Our code is available in `https://github.com/deep-spin/sparse_kernel_regression`.

$H_\alpha(\boldsymbol{p})$, where $H_\alpha(\boldsymbol{p}) = \frac{1}{\alpha(\alpha-1)}\left(1 - \sum_i p_i^\alpha\right)$ is the Tsallis α-entropy [35]. For $\alpha > 1$, α-entmax produces sparse probability distributions, with the degree of sparsity increasing as α grows. The limit $\alpha \to 1$ recovers softmax, while the special case $\alpha = 2$ yields *sparsemax* [19].

2.2 Attention and Kernel Regression

Attention mechanisms can be interpreted through the lens of nonparametric kernel regression [15, 36, §1.5]. In this view, attention computes a locally weighted average of values, with the weights determined by a similarity-dependent kernel centered at the query.

Formally, given data points $\{(\boldsymbol{k}_i, \boldsymbol{v}_i)\}_{i=1}^{n-1}$ and an input query $\boldsymbol{q} \equiv \boldsymbol{k}_n$, the *Nadaraya-Watson estimator* [20, 43] predicts the next target value as

$$\hat{\boldsymbol{v}}_n(\boldsymbol{q}) = \mathbb{E}[\boldsymbol{V}_n|\boldsymbol{q}] = \sum_{i=1}^{n-1} \frac{K_h(\boldsymbol{k}_i - \boldsymbol{q})}{\sum_{j=1}^{n-1} K_h(\boldsymbol{k}_j - \boldsymbol{q})} \boldsymbol{v}_i, \tag{2}$$

where $K_h(\boldsymbol{u}) = \frac{1}{h^d} K\left(\frac{\boldsymbol{u}}{h}\right)$ is a kernel on $\mathbb{R}^d$ with bandwidth h, and $\int_{\mathbb{R}^d} K(\boldsymbol{u}) d\boldsymbol{u} = 1$.[3] The kernel controls how strongly each data point $(\boldsymbol{k}_i, \boldsymbol{v}_i)$ contributes to the prediction as a function of the distance between $\boldsymbol{k}_i$ and the query $\boldsymbol{q}$.

Softmax attention as Gaussian kernel regression. Let $\boldsymbol{K} = [\boldsymbol{k}_1, ..., \boldsymbol{k}_{n-1}]^\top \in \mathbb{R}^{(n-1)\times d}$ denote the keys, $\boldsymbol{q} \equiv \boldsymbol{k}_n$ denote the query, and $\boldsymbol{V} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_n]^\top \in \mathbb{R}^{n\times d}$ the values. For clarity, we consider the setting in which the keys are ℓ_2-normalized, $\|\boldsymbol{q}\| = \|\boldsymbol{k}_i\| = 1$, which allows us to directly relate dot-product attention to Euclidean distances.

Consider the Gaussian kernel $K(\boldsymbol{u}) \propto \exp(-\frac{1}{2}\|\boldsymbol{u}\|^2)$. Using the identity $\frac{1}{2}\|\boldsymbol{k}_i - \boldsymbol{q}\|^2 = 1 - \boldsymbol{k}_i^\top \boldsymbol{q}$, the Nadaraya-Watson estimator (2) yields

$$\hat{\boldsymbol{v}}_n(\boldsymbol{q}) = \sum_{i=1}^{n-1} \mathrm{softmax}_i\left(\frac{\boldsymbol{K}\boldsymbol{q}}{h^2}\right) \boldsymbol{v}_i. \tag{3}$$

This expression recovers standard softmax attention [38], with the kernel bandwidth h corresponding to the *square root* of the softmax temperature. In transformers, the conventional scaling $\boldsymbol{k}_i^\top \boldsymbol{q}/\sqrt{d}$ therefore corresponds to choosing $h^2 = \sqrt{d}$, i.e., an effective bandwidth $h = d^{1/4}$. However, softmax produces dense attention weights, with all positions receiving nonzero probabilities. For long contexts, this can lead to attention dispersion [40] and makes token representations become less distinguishable [2], motivating sparse alternatives.

Choosing an appropriate bandwidth is central in kernel regression theory [36]. While vanilla transformers fix this choice via the $\sqrt{d}$ scaling, recent works adapt the effective bandwidth as a function of the sequence length n to control attention dispersion and long-context behavior [21, 39, 48]. Furthermore, this kernel regression view suggests a natural generalization, raising the following question: *"what if we use a different kernel than the Gaussian?"*.

Beyond Gaussian kernels. Classical kernel regression is not restricted to Gaussian kernels. A wide range of kernels has been studied, differing in smoothness and support properties. Of particular interest are kernels with *compact support*, which assign zero weight to points beyond a certain distance and therefore induce locality and sparsity.

Common examples include the *Epanechnikov kernel* $K(\boldsymbol{u}) \propto [1 - \|\boldsymbol{u}\|^2]_+$ [7], which has optimal properties in terms of minimizing mean integrated squared error (MISE), as well as its higher-order polynomial variants, including the *biweight* $K(\boldsymbol{u}) \propto [1 - \|\boldsymbol{u}\|^2]_+^2$ and *triweight* kernels $K(\boldsymbol{u}) \propto [1 - \|\boldsymbol{u}\|^2]_+^3$ [29]. These kernels have restricted support to a bounded neighborhood around the query, as illustrated in Figure 1.

Another way to enforce compact support is to truncate the Gaussian or uniform kernel to the k nearest neighbors:

$$K_{\text{top-}k}(\boldsymbol{u}) \propto \begin{cases} \exp\left(-\frac{1}{2}\|\boldsymbol{u}\|^2/h^2\right), & \|\boldsymbol{u}\| \le \delta_{\max}, \\ 0, & \text{otherwise}, \end{cases} \qquad K_{\text{uniform}}(\boldsymbol{u}) \propto \begin{cases} 1, & \|\boldsymbol{u}\| \le \delta_{\max}, \\ 0, & \text{otherwise}. \end{cases} \tag{4}$$

[3] Throughout, we use "kernel" in the classical sense of kernel density estimation and kernel regression. This should not be confused with positive-definite kernels in the RKHS sense.

where the threshold $\delta_{\max}$ is implicitly determined by the set $\mathcal{N}_k(\boldsymbol{q})$ of the k nearest neighbors of $\boldsymbol{q}$ in feature space, i.e., $\|\boldsymbol{k}_i - \boldsymbol{q}\| \leq \delta_{\max}$ if and only if $\boldsymbol{k}_i \in \mathcal{N}_k(\boldsymbol{q})$.

Epanechnikov, biweight, triweight, top-k Gaussian, and uniform kernels are all examples of kernels with compact support. As we show in §3, these classical kernel choices give rise to different sparse attention mechanisms, providing a unified kernel-based perspective on locality and sparsity in attention. We illustrate these kernels in Figure 1.

2.3 Memory Mosaics

Memory Mosaics [49] is an associative memory variant of the transformer architecture designed to perform prediction tasks using kernel regression; this architecture has recently shown to exhibit good performance at reasonable scales [48]. We base our work and experiments on this architecture as it **explicitly implements a Nadaraya-Watson estimator**, as described in (3), unlike vanilla transformers. Specifically, in Memory Mosaics, the query and key are *the same* by design, aligning exactly with kernel regression: $\boldsymbol{k}_n$ serves both as a query at the n^{th} time step to determine the weighting of past $(n-1)$ stored values, and as a key at subsequent time steps. Each memory unit contains a trainable feature extractor that computes keys and values from input vectors $\boldsymbol{x}_i \in \mathbb{R}^{d_z}$.

$$\begin{array}{rlll} \text{Keys:} & \boldsymbol{k}_n = \varphi(\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \ldots; \Theta) & \tilde{\boldsymbol{k}}_n = W_\varphi \boldsymbol{z}_n & \boldsymbol{k}_n = \mathrm{Norm}(\tilde{\boldsymbol{k}}_n + \lambda_\varphi \overline{\boldsymbol{k}}_{n-1}) \\ \text{Values:} & \boldsymbol{v}_n = \psi(\boldsymbol{x}_{n+1}, \boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \ldots; \Theta) & \tilde{\boldsymbol{v}}_n = W_\psi \boldsymbol{z}_n & \boldsymbol{v}_n = \mathrm{Norm}(\tilde{\boldsymbol{v}}_n + \lambda_\psi \tilde{\boldsymbol{v}}_{n+1}) \end{array} \tag{5}$$

Here, $\boldsymbol{k}_n, \boldsymbol{v}_n \in \mathbb{R}^d$ denote the key and value representations, and Θ represents learnable parameters. The key $\boldsymbol{k}_n$ depends only on current and past embeddings $(\boldsymbol{x}_i)_{i \leq n}$, whereas the value $\boldsymbol{v}_n$ incorporates a one-step lookahead to capture predictive structure. The projections use linear mappings $W_\varphi, W_\psi \in \mathbb{R}^{d \times d_z}$ followed by leaky averaging with coefficients $\lambda_\varphi, \lambda_\psi$, and normalization. The term $\overline{\boldsymbol{k}}_{n-1}$ denotes the running average of past keys.

In the Memory Mosaics architecture, each standard transformer block is replaced by two memories: a *contextual* memory unit, which applies attention via kernel regression using (3) and (5), and a *persistent* memory module. The persistent memory, inspired by earlier memory-augmented networks [30], consists of learned memory slots that store information across sequences, providing a form of long-term associative memory. Conceptually, it functions similarly to self-attention over fixed memory slots and can replace a standard feed-forward network by offering additional capacity and nonlinear processing. The attention mechanism excludes the main diagonal as opposed to vanilla transformers, making use of (3) where the sum runs only up to $n-1$ rather than n, which is why values can "peek" one step into the future in (5) to compensate. In this work, we focus on adapting the contextual memory unit to incorporate sparsity via different kernel choices while retaining the Nadaraya-Watson estimation principle.

3 Sparse Attention via Compact Kernels

We now formalize the connection between sparse attention mechanisms and kernel regression with compact support kernels. Starting from the kernel view of attention described in §2.2, we investigate how replacing the Gaussian kernel implicit in softmax attention with compact-support kernels naturally induces locality and sparsity.

Normalized ReLU and Epanechnikov kernel. Consider the Epanechnikov kernel mentioned earlier in §2.2, a compactly supported kernel widely used in kernel density estimation. Replacing the Gaussian kernel with the Epanechnikov kernel, yields the following query-key interaction and normalized attention rule:

$$K(\boldsymbol{k}_i - \boldsymbol{q}) = \left[\boldsymbol{k}_i^\top \boldsymbol{q}/\gamma + b\right]_+ \qquad \Longrightarrow \qquad \hat{\boldsymbol{v}}_n(\boldsymbol{q}) = \sum_{i=1}^{n-1} \frac{\left[\boldsymbol{k}_i^\top \boldsymbol{q}/\gamma + b\right]_+}{\sum_{j=1}^{n-1} \left[\boldsymbol{k}_j^\top \boldsymbol{q}/\gamma + b\right]_+} \boldsymbol{v}_i. \tag{6}$$

with $\gamma = h^2/2$ and $b = 1 - 2/h^2$. Thus, *normalized ReLU attention is equivalent to Epanechnikov kernel regression*, with sparsity arising from the ReLU truncation. This connection was also noted by [16], who used it to derive an Hopfield-like energy function. A limitation of the normalized ReLU (6) is that the attention weights are undefined when all pre-activations are negative, leading to $\frac{0}{0}$. Next, we show two principled ways to avoid this degeneracy: (i) *auto-normalization* (adaptive

bandwidth) leading to sparsemax and α-entmax, and (ii) *anchoring the support* at nearest keys, leading to ReLUmax.

3.1 Auto-normalization (Adaptive Bandwidth)

An alternative to the normalization (6) is to choose a *data-dependent bandwidth* h which ensures the denominator in the Nadaraya-Watson estimator is a constant—we call this *auto-normalization*. We show next that this procedure yields attention transformations such as sparsemax and α-entmax.

We begin with sparsemax and show how it arises from auto-normalized Epanechnikov regression.

Since we have normalized queries and keys, we can rewrite the squared distance as $\frac{1}{2}\|\boldsymbol{k}_i - \boldsymbol{q}\|^2 = 1 - \boldsymbol{k}_i^\top \boldsymbol{q}$. Then, introducing a temperature γ, the denominator in (2) can be rewritten as

$$\sum_{i=1}^{n-1}\left[1 - \frac{\|\boldsymbol{k}_i - \boldsymbol{q}\|^2}{h^2}\right]_+ = \left(\frac{2\gamma}{h^2}\right)\sum_{i=1}^{n-1}\left[\frac{h^2}{2\gamma} - \frac{1}{\gamma} + \frac{\boldsymbol{k}_i^\top \boldsymbol{q}}{\gamma}\right]_+ \tag{7}$$

In Nadaraya-Watson estimation, the denominator enforces that the weights sum to one. An alternative is to absorb this normalization into the kernel scale such that, rather than fixing h, we choose it so that the rectified responses sum to one. Concretely,

$$\sum_{i=1}^{n-1}\left[\frac{\boldsymbol{k}_i^\top \boldsymbol{q}}{\gamma} - \underbrace{\left(\frac{1}{\gamma} - \frac{h^2}{2\gamma}\right)}_{:=\tau}\right]_+ = 1 \quad \Longrightarrow \quad \hat{\boldsymbol{v}}_n(\boldsymbol{q}) = \sum_{i=1}^{n-1}\text{sparsemax}_i\left(\frac{\boldsymbol{K}\boldsymbol{q}}{\gamma}\right)\boldsymbol{v}_i, \tag{8}$$

showing that Epanechnikov kernel regression with auto-normalization is equivalent to sparsemax attention. In this construction the effective bandwidth is therefore *adaptive* $h = \sqrt{2 - 2\gamma\tau}$, where τ (hence h) is determined by the attention scores via (16), which depends only on the keys (but not the values). This contrasts with common bandwidth schedules that depend only on sequence length [48]. It is also interesting to compare (8) with (3): in the softmax attention induced by the Gaussian kernel, we simply have $h = \sqrt{\gamma}$.

From sparsemax to α-entmax via compact kernels. Instead of the linear truncation used by sparsemax, α-entmax introduces a higher-order rectification controlled by a parameter $\alpha > 1$, producing attention weights with an implicitly defined threshold τ. On the kernel side, this corresponds to rectified polynomial kernels of the form $K_h(\boldsymbol{u}) \propto [1 - \|\boldsymbol{u}\|^2/h^2]_+^r$, where $r = 1/(\alpha - 1)$. This family recovers Epanechnikov ($r = 1$), biweight ($r = 2$), and triweight ($r = 3$) as special cases. Proposition 1 formalizes this correspondence, and the proof is given in Appendix A.

Proposition 1. *Let $r \geq 1$ and $\alpha = 1 + \frac{1}{r}$. Let $\boldsymbol{K} = [\boldsymbol{k}_1, ..., \boldsymbol{k}_{n-1}]^\top \in \mathbb{R}^{(n-1)\times d}$ be normalized keys, $\boldsymbol{q} \equiv \boldsymbol{k}_n \in \mathbb{R}^d$ the query, and $\boldsymbol{V} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_n]^\top$ the values. The α-entmax attention with temperature γ satisfies*

$$\sum_{i=1}^{n-1}\alpha\text{-entmax}_i\left(\frac{\boldsymbol{K}\boldsymbol{q}}{\gamma}\right)\boldsymbol{v}_i = \frac{\sum_{i=1}^{n-1} K_h(\boldsymbol{k}_i - \boldsymbol{q})\,\boldsymbol{v}_i}{\sum_{i=1}^{n-1} K_h(\boldsymbol{k}_i - \boldsymbol{q})}, \tag{9}$$

where $K_h(\boldsymbol{u}) \propto \left[1 - \frac{\|\boldsymbol{u}\|^2}{h^2}\right]_+^r$ and h is implicitly defined by $h = \sqrt{2 - 2r\gamma\tau}$.

Proposition 1 yields a continuous family of compact-support kernel attention mechanisms indexed by α. In particular, $r = 2$ corresponds to 1.5-entmax and the biweight kernel, while $r = 3$ corresponds to $\frac{4}{3}$-entmax and the triweight kernel, recovering compact-kernel associative memories studied in Hopfield networks and related energy-based models [44, 27, 26]. Figure 1 summarizes these correspondences between classical compact kernels and sparse attention transformations.

The rectified polynomial family is only one way to obtain compact support. A second, complementary route is to *anchor* sparsity directly on the nearest keys by restricting attention to a neighborhood defined by kNN selection, i.e., by truncating an underlying kernel to a data-dependent support set. We turn to these anchored compact kernels next.

3.2 Anchored Compact Kernels

In this second route, instead of changing the kernel shape, we keep a base kernel and make its *support set* data-adaptive, either by truncation (top-k) or by max-anchoring (ReLUmax).

Top-k average pooling (uniform-kNN). Consider the uniform kernel in (4). If we choose $\delta_{\max}$ so that exactly k keys satisfy $\|\boldsymbol{k}_i - \boldsymbol{q}\| \le \delta_{\max}$, then the support set is precisely the k-nearest-neighbor set $\mathcal{N}_k(\boldsymbol{q})$. Because all selected keys receive identical weight under the uniform kernel, the Nadaraya-Watson estimate reduces to $\hat{\boldsymbol{v}}(\boldsymbol{q}) = \frac{1}{k}\sum_{i\in\mathcal{N}_k(\boldsymbol{q})} \boldsymbol{v}_i$. Thus, top-$k$ average pooling is equivalent to uniform-kNN kernel regression with an adaptively chosen radius that admits exactly k neighbors.

Top-k softmax (truncated Gaussian). A smoother, anchored alternative is to use a Gaussian kernel within the k-nearest-neighbor set and set all weights to zero outside it. This corresponds to the truncated kernel in (4). Normalizing these weights yields

$$\hat{\boldsymbol{v}}(\boldsymbol{q}) = \sum_{i\in\mathcal{N}_k(\boldsymbol{q})} \frac{K_{\text{top-}k}(\boldsymbol{k}_i - \boldsymbol{q})}{\sum_{j\in\mathcal{N}_k(\boldsymbol{q})} K_{\text{top-}k}(\boldsymbol{k}_j - \boldsymbol{q})}\, \boldsymbol{v}_i. \tag{10}$$

This mechanism can be interpreted as Gaussian kernel regression with a data-adaptive compact support set, given by $\mathcal{N}_k(\boldsymbol{q})$, yielding a sparse kNN-style attention mechanism [14].

r-ReLUmax. The top-k constructions above enforce compact support by *hard* truncation via kNN selection. We consider a new transformation, r-ReLUmax, which anchors support *softly* at the best-matching key while retaining a compact kernel and avoiding the empty-support degeneracy of normalized ReLU. Let $m := \max_j \boldsymbol{k}_j^\top \boldsymbol{q}$ and $b > 0 \in \mathbb{R}$ be a hyperparameter. The r-ReLUmax kernel and the corresponding Nadaraya-Watson estimate are

$$K_{r\text{-relumax}}(\boldsymbol{k}_i - \boldsymbol{q}) \propto \left[b + \frac{\boldsymbol{k}_i^\top \boldsymbol{q} - m}{h^2}\right]_+^r \quad\Longrightarrow\quad \hat{\boldsymbol{v}}(\boldsymbol{q}) = \sum_{i=1}^{n-1} \frac{\left[b + \frac{\boldsymbol{k}_i^\top \boldsymbol{q} - m}{h^2}\right]_+^r}{\sum_{k=1}^{n-1}\left[b + \frac{\boldsymbol{k}_k^\top \boldsymbol{q} - m}{h^2}\right]_+^r}\, \boldsymbol{v}_i. \tag{11}$$

Importantly, since $b > 0$, in the worst case where all attention scores equal m, at least one rectified score will be strictly positive, and so the denominator in (11) cannot be zero. Hence, ReLUmax avoids the $0/0$ degeneracy of normalized ReLU (6) while preserving compact support.Furthermore, we note that (11) yields an explicit characterization of the active set:

$$K_{r\text{-relumax}}(\boldsymbol{k}_i - \boldsymbol{q}) > 0 \quad\Longleftrightarrow\quad b + \frac{\boldsymbol{k}_i^\top \boldsymbol{q} - m}{h^2} > 0 \quad\Longleftrightarrow\quad \boldsymbol{k}_i^\top \boldsymbol{q} > m - bh^2. \tag{12}$$

Thus, ReLUmax selects *all keys within a fixed similarity margin of the maximum similarity*. This is fundamentally different from top-k since the number of active keys adapts to the distribution of score gaps rather than being fixed. In fact, this points to a *top-p* viewpoint. Top-k fixes the *cardinality* of the support, whereas ReLUmax fixes a *margin-to-maximum* threshold, with keys being kept whenever their score exceeds $m - bh^2$. Consequently, the retained set expands when many keys have scores close to the maximum (small gaps) and shrinks when the maximum is well separated (large gaps). In this sense, ReLUmax behaves like a continuous analogue of nucleus selection as it adaptively includes "as many keys as needed" near the top of the score distribution, but does so via a smooth, differentiable margin.

4 Experiments

We conduct three sets of experiments to validate our theoretical findings and demonstrate the practical benefits of our kernel-based attention mechanisms in Memory Mosaics.

4.1 Language Modeling

Setup. We replicate Memory Mosaics [49] with our proposed kernel-based sparse attention mechanisms and conduct language modeling experiments on small, self-contained narratives written in simple English, designed to be comprehensible to a young child. Specifically, we follow the methodology of [18] and [49], and evaluate on BabiStories. To study the effect of different attention

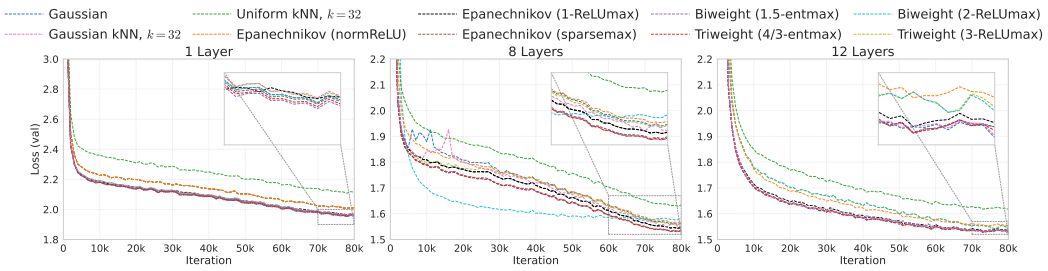


Figure 2: **Validation loss of Memory Mosaic models with different kernels.** Comparison on the BabiStories dataset for varying model depths. The horizontal axis shows the number of training iterations.

mechanisms, we replace the standard softmax attention used in Memory Mosaics [49] with the family of kernel-based methods described in §3. Namely, we consider Gaussian (softmax) attention and compact rectified polynomial kernels: Epanechnikov (r=1) via normalized ReLU, sparsemax, and 1-, 2-, and 3-ReLUmax (b=1), and higher-order biweight (r=2) and triweight (r=3), which correspond to α-entmax with α=1.5 and $\alpha=\frac{4}{3}$, respectively. In addition to r-order rectified polynomial kernels, we evaluate discrete uniform and Gaussian kNN attention. Experimental details can be found in §B.1. For 2-ReLUmax and 8 layers, a slightly lower learning rate was required due to training instabilities.

Discussion. Figure 2 shows a comparison of models equipped with different kernel-based attention mechanisms across model depths in terms of validation loss. For shallow models (one layer), compactly supported kernels, corresponding to adaptive bandwidth, achieve lower validation loss than dense Gaussian attention and top-k baselines. As depth increases to 8 layers, this advantage remains, and rectified polynomial kernels with adaptive bandwidth outperform dense softmax attention and anchored top-k variants across iterations. With 12 layers, the gap is narrower and, except normReLU and uniform kNN, all approaches achieve roughly the same validation loss. Interestingly, while top-k Gaussian attention can reach competitive validation loss at intermediate depths, it exhibits mild instability, likely due to the non-smoothness introduced by discrete neighbor selection. Among sparse mechanisms, auto-normalized attention mappings—corresponding to sparsemax and α-entmax—consistently perform better than normReLU. While r-ReLUmax outperforms normReLU with 8 and 12 layers, it slightly underperforms adaptive normalization methods. We use AdaSplash [10] for efficiency, with compact kernels, and discuss the efficiency of all methods in §B.4.

4.2 In-Context Learning

Setup. We also evaluate our models on an in-context learning task. To rigorously compare the in-context learning capabilities of different architectures, we adopt the RegBench benchmark [1], a diagnostic suite designed to probe sequence models' ability to infer the generative processes of regular languages, which constructs random artificial languages defined by a PFA with sequences consisting of 10 to 20 strings sampled from the same PFA. We compare baseline transformers and kernelized variants of our sparse Memory Mosaic architectures, as well as several established baselines including S4 [13], Gla [45], RetNet [32], H3 [8], Mamba [12], and standard transformers [38]. Our models match the sequence length and vocabulary size of [1] and follow the same hyperparameter tuning strategy. Full experimental details are provided in §B.2. Each model is trained on datasets ranging from 1k to 40k examples and evaluated using per-token loss as a function of the generated token position for a test set of 1k examples.

Discussion. Table 1 compares models on RegBench using last-token accuracy and total variation distance (TVD). Among the Memory Mosaic variants, rectified polynomial kernels (Triweight, Biweight, and Epanechnikov) with α-entmax consistently match or slightly outperform the Gaussian kernel in the low-data regime (1k–5k), achieving both higher accuracy and lower TVD. The adaptive bandwidth family (α-entmax) performs best overall as compared to max-anchoring (r-ReLUmax). In contrast, normalized ReLU shows a clear degradation in both metrics, suggesting that simple thresholded normalization without adaptive support selection is insufficient in this setting. Fixed

Table 1: **Performance comparison across varying training set sizes in the Regbench in-context learning benchmark.** Accuracy (↑) measures the correct prediction rate, and TVD (↓) measures divergence from the target distribution. We **bold** the best performing models.

Method	1k		2.5k		5k		10k		20k		40k	
	Acc ↑	TVD ↓	Acc ↑	TVD ↓	Acc ↑	TVD ↓	Acc ↑	TVD ↓	Acc ↑	TVD ↓	Acc ↑	TVD ↓
RetNet	51.3	67.4	50.5	63.9	81.5	33.3	88.4	25.7	90.8	22.5	92.6	18.9
H3	51.5	65.1	58.0	57.0	70.9	47.4	76.9	41.4	80.7	36.3	87.7	25.4
Gla	51.8	65.3	51.1	62.2	82.1	31.0	90.2	24.5	91.2	21.2	92.9	19.0
S4	52.1	62.8	51.7	62.2	54.5	59.0	63.3	52.9	72.1	45.8	82.5	33.8
Mamba	69.4	51.0	79.4	41.0	88.7	30.6	90.5	28.4	91.3	27.0	91.5	27.0
Transformer	54.5	62.6	92.4	20.5	92.7	18.9	94.8	15.7	95.1	15.0	95.5	13.9
Memory Mosaics:												
Gaussian	93.9	34.1	93.7	20.8	94.6	17.7	95.0	15.9	**95.4**	15.3	95.4	14.2
Gaussian kNN, $k = 16$	93.4	33.3	93.7	19.8	94.4	17.2	94.8	16.0	94.8	16.0	95.0	14.9
Uniform kNN, $k = 16$	84.6	41.1	85.6	28.3	86.1	27.2	86.2	26.4	86.5	26.4	86.4	26.4
Epanechnikov (normReLU)	84.3	43.6	87.0	29.6	90.1	24.6	91.6	21.7	93.1	19.4	93.3	18.6
Epanechnikov (1-ReLUmax)	93.8	36.8	94.2	20.4	94.3	17.8	94.9	**15.7**	**95.4**	**15.0**	95.4	14.1
Epanechnikov (sparsemax)	**94.2**	**32.5**	94.0	19.3	94.4	17.1	94.9	15.9	95.0	**15.0**	95.4	14.0
Biweight (2-ReLUmax)	93.8	36.4	94.4	20.1	94.6	17.2	95.1	16.1	95.3	15.1	95.1	14.8
Biweight (1.5-entmax)	93.8	33.0	**94.3**	19.2	94.7	16.6	**95.2**	**15.7**	95.0	15.8	**95.5**	**13.8**
Triweight (3-ReLUmax)	93.9	36.6	94.2	20.5	94.7	16.9	94.8	16.4	95.1	15.4	95.3	14.3
Triweight (4/3-entmax)	93.9	32.9	**94.3**	**19.0**	**94.8**	**16.2**	94.9	16.1	95.1	15.3	95.3	14.4

top-k uniform attention underperforms across all data scales, highlighting the limitations of hard, data-independent sparsity compared to kernel-based adaptive weighting. Overall, these results indicate that adaptive sparsity induced by compact kernels is most beneficial in data-scarce regimes.

4.3 Length Generalization

Setup. We evaluate the length generalization capabilities of our Memory Mosaics using synthetic tasks, following the setup of [39], such as MQMTAR (multi-query multi-token associative recall), sequence sorting, and reversing, which provide precise control over training and test sequence lengths. These tasks test the model's ability to maintain attention on relevant, sparse subsets of tokens, and probe the model's ability to learn an underlying length-agnostic algorithm, rather than memorizing patterns. Full experimental details are provided in §B.3. We note that Memory Mosaics does not rely on positional embeddings. For completeness, we also evaluate a variant combining the Epanechnikov kernel with ALiBi positional bias [23]. As shown in §C.1, this modification leads to a consistent degradation in performance, motivating our decision to omit positional embeddings in the final model.

Discussion. The results in Table 2 show that Gaussian attention performs well in-distribution but quickly deteriorates when extrapolating to long sequences. Top-k uniform attention is too crude, failing to extrapolate for any of the tasks. Gaussian kNN mitigates some degradation by limiting attention to nearest neighbors, yet it still achieves poor length extrapolation, especially on very long sequences. In contrast, our rectified polynomial kernels (α-entmax) exhibit strong and stable generalization, particularly on the MQMTAR task. Triweight and biweight kernels outperform Gaussian and top-k mechanisms on MQMTAR, with triweight also leading on the reverse task, while preserving high accuracy over longer sequences. Epanechnikov-based attention mechanisms, specifically 1-ReLUmax and sparsemax, consistently achieve perfect in-distribution accuracy while maintaining superior length extrapolation performance, with sparsemax showing better generalization performance. Finally, 1-ReLUmax outperforms normReLU (for $r = 1$), benefiting from its max-anchoring operation, with improved stability at longer sequence lengths.

5 Discussion and Related Work

Sparse attention mechanisms. Sparse attention offers an alternative to dense softmax, enabling models to focus on a subset of relevant tokens. Early approaches such as top-k softmax [14] enforced sparsity by keeping only the largest attention scores, but suffered from non-smooth gradients and a fixed attention budget. One line of work employs predefined sparse patterns, such as local sliding windows or fixed strides, such as LongFormer [5] and BigBird [47]. Other methods introduce end-to-

Table 2: Exact match accuracy on representative synthetic tasks. In-distribution results ($n = 64$) and out-of-distribution performance at increasing sequence lengths are reported. Values show the mean across three seeds, with the maximum across seeds indicated in superscript. Best average performance is in **bold** and overall maximum performance is underlined. L is the number of layers.

	MQMTAR ($L = 4$)							Reverse ($L = 4$)		Sort ($L = 2$)		
Kernel	$1\times$	$2\times$	$4\times$	$8\times$	$16\times$	$32\times$	$64\times$	$1\times$	$1.5\times$	$1\times$	$2\times$	$4\times$
Transformer	**1.00**$^{(1.00)}$	**1.00**$^{(1.00)}$	0.99$^{(0.99)}$	0.09$^{(0.15)}$	0.01$^{(0.01)}$	0.00$^{(0.00)}$	0.00$^{(0.00)}$	**1.00**$^{(1.00)}$	**0.52**$^{(0.62)}$	**1.00**$^{(1.00)}$	0.00$^{(0.00)}$	0.00$^{(0.00)}$
Gaussian	0.99$^{(1.00)}$	0.96$^{(1.00)}$	0.93$^{(0.99)}$	0.85$^{(0.97)}$	0.64$^{(0.81)}$	0.14$^{(0.20)}$	0.00$^{(0.00)}$	**1.00**$^{(1.00)}$	0.14$^{(0.38)}$	**1.00**$^{(1.00)}$	0.06$^{(0.17)}$	0.00$^{(0.00)}$
Gaussian kNN, $k = 32$	0.99$^{(1.00)}$	0.97$^{(1.00)}$	0.93$^{(0.99)}$	0.84$^{(0.96)}$	0.68$^{(0.85)}$	0.27$^{(0.36)}$	0.05$^{(0.05)}$	0.99$^{(1.00)}$	0.00$^{(0.00)}$	**1.00**$^{(1.00)}$	0.80$^{(0.91)}$	0.01$^{(0.02)}$
Uniform kNN, $k = 32$	0.04$^{(0.08)}$	0.00$^{(0.00)}$	0.00$^{(0.00)}$	0.00$^{(0.00)}$	0.00$^{(0.00)}$	0.00$^{(0.00)}$	0.00$^{(0.00)}$	0.42$^{(0.66)}$	0.00$^{(0.00)}$	**1.00**$^{(1.00)}$	0.00$^{(0.00)}$	0.00$^{(0.00)}$
Epanechnikov (normReLU)	**1.00**$^{(1.00)}$	**1.00**$^{(1.00)}$	0.89$^{(1.00)}$	0.66$^{(1.00)}$	0.44$^{(1.00)}$	0.33$^{(0.98)}$	**0.31**$^{(0.94)}$	**1.00**$^{(1.00)}$	0.19$^{(0.56)}$	**1.00**$^{(1.00)}$	0.05$^{(0.14)}$	0.00$^{(0.00)}$
Epanechnikov (1-ReLUmax)	**1.00**$^{(1.00)}$	**1.00**$^{(1.00)}$	0.99$^{(1.00)}$	0.92$^{(0.96)}$	0.74$^{(0.87)}$	0.42$^{(0.59)}$	0.12$^{(0.18)}$	**1.00**$^{(1.00)}$	0.23$^{(0.68)}$	**1.00**$^{(1.00)}$	0.80$^{(1.00)}$	0.30$^{(0.91)}$
Epanechnikov (sparsemax)	**1.00**$^{(1.00)}$	**1.00**$^{(1.00)}$	**1.00**$^{(1.00)}$	**0.98**$^{(1.00)}$	0.86$^{(0.98)}$	**0.63**$^{(0.84)}$	**0.31**$^{(0.51)}$	**1.00**$^{(1.00)}$	0.03$^{(0.06)}$	**1.00**$^{(1.00)}$	0.92$^{(1.00)}$	0.29$^{(0.79)}$
Biweight (2-ReLUmax)	0.99$^{(1.00)}$	0.97$^{(1.00)}$	0.93$^{(1.00)}$	0.85$^{(0.96)}$	0.69$^{(0.87)}$	0.27$^{(0.43)}$	0.04$^{(0.06)}$	**1.00**$^{(1.00)}$	0.26$^{(0.74)}$	**1.00**$^{(1.00)}$	0.93$^{(1.00)}$	**0.34**$^{(0.74)}$
Biweight (1.5-entmax)	**1.00**$^{(1.00)}$	**1.00**$^{(1.00)}$	0.99$^{(1.00)}$	0.95$^{(0.99)}$	0.75$^{(0.93)}$	0.51$^{(0.74)}$	0.23$^{(0.42)}$	**1.00**$^{(1.00)}$	0.05$^{(0.06)}$	**1.00**$^{(1.00)}$	0.39$^{(1.00)}$	0.26$^{(0.77)}$
Triweight (3-ReLUmax)	0.99$^{(1.00)}$	0.97$^{(1.00)}$	0.93$^{(0.99)}$	0.85$^{(0.96)}$	0.71$^{(0.87)}$	0.37$^{(0.58)}$	0.08$^{(0.18)}$	**1.00**$^{(1.00)}$	0.29$^{(0.88)}$	**1.00**$^{(1.00)}$	0.95$^{(0.97)}$	0.30$^{(0.62)}$
Triweight (4/3-entmax)	**1.00**$^{(1.00)}$	**1.00**$^{(1.00)}$	0.99$^{(1.00)}$	0.97$^{(0.98)}$	**0.89**$^{(0.95)}$	0.61$^{(0.76)}$	0.25$^{(0.45)}$	**1.00**$^{(1.00)}$	0.26$^{(0.70)}$	**1.00**$^{(1.00)}$	**0.98**$^{(0.99)}$	0.07$^{(0.11)}$

end learnable sparsity by replacing softmax with differentiable alternatives such as sparsemax [19] and its generalization α-entmax [22]. More recently, [17], [44], [27, 26], and [16] develop sparse Hopfield networks—mathematically equivalent to sparse attention mechanisms—, while [39] show that α-entmax attention can mitigate attention dispersion and representational collapse for long sequences. Our paper provides a test-time regression interpretation of these frameworks.

Kernel methods in transformers. Early interpretations view softmax attention as a kind of similarity kernel, where attention weights arise from pairwise comparisons between queries and keys [34]. Building on this statistical view, recent work has shown that at test time transformers can behave like implicit Nadaraya-Watson estimators, with attention patterns converging to regression-like weights that resemble classical kernel regression outputs [42]. Related research explores test-time regression and meta-learning behaviours in transformers and fast weight systems where models implicitly solve regression or optimization tasks in context [41, 28, 4, 31]. While prior work focuses on dense kernels and efficient approximations, we extend the kernel regression view to sparse, compactly supported kernels, unifying dense and sparse attention within a single framework.

Neurobiological context. In neurobiological systems, the problem of understanding the long-range dependency structure in sequences arises in episodic memory [37]. The hippocampal indexing theory suggests that the brain solves this problem with hippocampal engrams serving as indices with which the brain retrieves neocortical traces of previous experiences [33, 11]. Notably, a core characteristic of hippocampal population activity is sparseness. This property of hippocampal activity is often attributed to its functional role of pattern separation [46]. From the perspective of the current work, hippocampal indices serve as keys, which can be queried, to neocortical values [9]. Thus, sparsity can be understood as an implicit use of compact-support kernels whereby neocortical memory retrieval is mediated by a relatively small set of well-separated memories, rather than weakly integrating over an increasingly large number of memories accumulated over a lifetime [37].

6 Conclusions

In this work, we show that sparse attention mechanisms can be interpreted as kernel regression with compactly supported kernels, providing a principled framework that unifies dense and sparse attention. We demonstrate that r-order rectified polynomial kernels allow controlled sparsity, from which several attention mechanisms, including α-entmax, can be derived. For $r = 1$, the Epanechnikov kernel underlies several attention mechanisms: normalized ReLU, which can produce empty-support indeterminacies when all pre-activations are negative; 1-ReLUmax, which anchors the support at the maximum key to avoid this issue; and sparsemax, which further introduces adaptive normalization by adjusting the kernel bandwidth based on the attention scores also ensuring non-degenerate attention. Our empirical results confirm that attention mechanisms with adaptive normalization outperform dense Gaussian and heuristic top-k methods, generally delivering superior performance. While we experiment with models of various sizes, our experiments are all with relatively small models ($<$1B parameters). Further scaling experiments, beyond our compute budget, are needed to assess the benefits of compact kernels in larger models. We highlight this as a direction for future work.

Acknowledgments

We would like to thank Hugo Pitorro, Lili Mou, Pavlo Vasylenko, Mathias Lindemann, and the SARDINE lab team for the helpful discussions. We thank the Champalimaud Foundation for supporting this research. This work was supported by the project DECOLLAGE (ERC-2022-CoG 101088763), and by FCT/MECI through national funds and when applicable co-funded EU funds under UID/50008: Instituto de Telecomunicações.

References

[1] Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms, 2024.

[2] Federico Barbero, Andrea Banino, Steven Kapturowski, Dharshan Kumaran, João Madeira Araújo, Oleksandr Vitvitskyi, Razvan Pascanu, and Petar Veličković. Transformers need glasses! information over-squashing in language tasks. *Advances in Neural Information Processing Systems*, 37:98111–98142, 2024.

[3] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. *arXiv preprint arXiv:2504.13173*, 2025.

[4] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

[5] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv:2004.05150 [cs]*, April 2020. arXiv: 2004.05150.

[6] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[7] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969.

[8] Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry Hungry Hippos: Towards language modeling with state space models. In *International Conference on Learning Representations*, 2023.

[9] Samuel J. Gershman, Ila Fiete, and Kazuki Irie. Key-value memory in the brain. *arXiv:2501.02950 [q-bio.NC]*, 2025. Preprint, Jan 2025.

[10] Nuno Gonçalves, Marcos V Treviso, and Andre Martins. Adasplash: Adaptive sparse flash attention. In *Forty-second International Conference on Machine Learning*, 2025.

[11] Travis D. Goode, Kazumasa Z. Tanaka, Amar Sahay, and Thomas J. McHugh. An integrated index: Engrams, place cells, and hippocampal memory. *Neuron*, 107(5):805–820, 2020. Epub 2020 Aug 6.

[12] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[13] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations (ICLR)*, 2022.

[14] Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. Memory-efficient transformers via top-k attention. In Nafise Sadat Moosavi, Iryna Gurevych, Angela Fan, Thomas Wolf, Yufang Hou, Ana Marasović, and Sujith Ravi, editors, *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pages 39–52, Virtual, November 2021. Association for Computational Linguistics.

[15] Wolfgang Härdle. *Applied nonparametric regression*. Number 19. Cambridge university press, 1990.

[16] Benjamin Hoover, Krishna Balasubramanian, Dmitry Krotov, and Parikshit Ram. Dense associative memory with epanechnikov energy. In *New Frontiers in Associative Memories*, 2025.

[17] Jerry Yao-Chieh Hu, Donglin Yang, Dennis Wu, Chenwei Xu, Bo-Yu Chen, and Han Liu. On sparse modern hopfield model. *Advances in Neural Information Processing Systems*, 36:27594–27608, 2023.

[18] Yuanzhi Li and Ronen Eldan. Tinystories: How small can language models be and still speak coherent english, 2024.

[19] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pages 1614–1623, New York, New York, USA, 20–22 Jun 2016. PMLR.

[20] E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[21] Ken M Nakanishi. Scalable-softmax is superior for attention. *arXiv preprint arXiv:2501.19399*, 2025.

[22] Ben Peters, Vlad Niculae, and André FT Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1504–1519, 2019.

[23] Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022.

[24] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[25] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, et al. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2020.

[26] Saul Santos, Vlad Niculae, Daniel McNamee, and Andre F.T. Martins. Hopfield-fenchel-young networks: A unified framework for associative memory retrieval. *Journal of Machine Learning Research*, 26(265):1–51, 2025.

[27] Saul José Rodrigues Santos, Vlad Niculae, Daniel C McNamee, and Andre Martins. Sparse and structured hopfield networks. In *Forty-first International Conference on Machine Learning*, 2024.

[28] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021.

[29] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[30] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[31] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. In *Forty-second International Conference on Machine Learning*, 2025.

[32] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, 2023.

[33] T. J. Teyler and P. DiScenna. The hippocampal memory indexing theory. *Behavioral Neuroscience*, 100(2):147–154, 1986.

[34] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *EMNLP*, 2019.

[35] Constantino Tsallis. Possible generalization of boltzmann-gibbs statistics. *Journal of statistical physics*, 52:479–487, 1988.

[36] Alexandre B Tsybakov. Nonparametric estimators. In *Introduction to Nonparametric Estimation*, pages 1–76. Springer, 2008.

[37] Endel Tulving. Episodic memory: From mind to brain. *Annual Review of Psychology*, 53:1–25, 2002.

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

[39] Pavlo Vasylenko, Hugo Pitorro, André F. T. Martins, and Marcos Treviso. Long-context generalization with sparse attention, 2025.

[40] Petar Veličković, Christos Perivolaropoulos, Federico Barbero, and Razvan Pascanu. Softmax is not enough (for sharp size generalisation). In *Forty-second International Conference on Machine Learning*, 2025.

[41] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

[42] Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory, 2025.

[43] Geoffrey S. Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 26(4):359–372, 1964.

[44] Dennis Wu, Jerry Yao-Chieh Hu, Weijian Li, Bo-Yu Chen, and Han Liu. Stanhop: Sparse tandem hopfield model for memory-enhanced time series prediction. In *The Twelfth International Conference on Learning Representations*, 2024.

[45] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

[46] Michael A. Yassa and Craig E. L. Stark. Pattern separation in the hippocampus. *Trends in Neurosciences*, 34(10):515–525, 2011.

[47] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 2020.

[48] Jianyu Zhang and Léon Bottou. Memory mosaics at scale. In *Advances in Neural Information Processing Systems*, 2025.

[49] Jianyu Zhang, Niklas Nolte, Ranajoy Sadhukhan, Beidi Chen, and Leon Bottou. Memory mosaics. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Representation Learning*, volume 2025, pages 36412–36433, 2025.

A Proof of Proposition 1

Proof. Let $\boldsymbol{q} \equiv \boldsymbol{k}_n$ and $r = \frac{1}{\alpha-1}$. Since $\|\boldsymbol{k}_i\| = \|\boldsymbol{q}\| = 1$, we have $\frac{1}{2}\|\boldsymbol{k}_i - \boldsymbol{q}\|^2 = 1 - \boldsymbol{k}_i^\top \boldsymbol{q}$. The r-weight polynomial kernel is defined as:

$$K_h(\boldsymbol{k}_i - \boldsymbol{q}) = \left[1 - \frac{\|\boldsymbol{k}_i - \boldsymbol{q}\|^2}{h^2}\right]_+^r \tag{13}$$

$$= \left[1 - \frac{2(1 - \boldsymbol{k}_i^\top \boldsymbol{q})}{h^2}\right]_+^r \tag{14}$$

$$= \left[\frac{h^2 - 2 + 2\boldsymbol{k}_i^\top \boldsymbol{q}}{h^2}\right]_+^r. \tag{15}$$

That is, the induced kernel takes the form $K_h(\boldsymbol{u}) \propto \left[1 - \|\boldsymbol{u}\|^2\right]_+^r$, a rectified polynomial kernel of order r, encompassing classical kernels such as the biweight ($r = 2$) and triweight ($r = 3$). Now, introducing a temperature γ, the denominator in (9) can be expressed as

$$\sum_{i=1}^{n-1} \left[1 - \frac{\|\boldsymbol{k}_i - \boldsymbol{q}\|^2}{h^2}\right]_+^r = \left(\frac{2r\gamma}{h^2}\right)^r \sum_{i=1}^{n-1} \left[\frac{h^2}{2r\gamma} - \frac{1}{r\gamma} + \frac{\boldsymbol{k}_i^\top \boldsymbol{q}}{r\gamma}\right]_+^r. \tag{16}$$

Choosing h such that

$$\sum_{i=1}^{n-1} \left[\frac{\boldsymbol{k}_i^\top \boldsymbol{q}}{r\gamma} - \underbrace{\left(\frac{1}{r\gamma} - \frac{h^2}{2r\gamma}\right)}_{:=\tau}\right]_+^r = 1, \tag{17}$$

the normalizer $\tau = \frac{1}{r\gamma} - \frac{h^2}{2r\gamma}$ coincides with the α-entmax threshold (1), yielding

$$\hat{\boldsymbol{v}}_n(\boldsymbol{q}) = \sum_{i=1}^{n-1} \alpha\text{-entmax}_i\left(\frac{\boldsymbol{K}\boldsymbol{q}}{\gamma}\right) \boldsymbol{v}_i, \tag{18}$$

with an adaptive bandwidth $h = \sqrt{2 - 2r\gamma\tau}$. □

B Experimental Details

B.1 Language Modeling

In this section, we provide additional details of our experimental setup, including model configurations and training hyperparameters for the language modeling task (Table 3). We also show the full training and validation curves for the BabiStories dataset (see statistics in Table 4) [49], complementing the results in the main text (Figure 3). All models use the GPT-2 tokenizer [24] and share a sequence length of 512 tokens. Experiments cover different numbers of parallel layers of persistent and contextual memories. For single-layer models, we vary $k \in 16, 32, 64$ and report results for the best-performing value ($k = 32$), which is then used for all multi-layer configurations. Each contextual memory unit computes leaky-average keys and one-step-ahead values, with retrieval performed via kernel-weighted interpolation over past embeddings. All models are trained for 80k iterations with a global batch size of 512, optimized with Adam, and using learning rate schedules consistent with GPT-2 Small [24]. For 12-layer models, the learning rate is reduced to 10^{-3}. Sparse attention patterns for the rectified polynomial kernels are learned efficiently using AdaSplash [10]. While in practice the normalized ReLU is not often met, we use uniform weighting for those cases. These experiments were run on h100 and h200 GPUs.

B.2 In-Context Learning

For evaluation, we use RegBench [1]. RegBench constructs random artificial languages defined by a PFA. Each input sequence consists of 10–20 strings sampled from the same PFA, separated by special

Table 3: Main training and model hyperparameters across model depths. Unless otherwise specified, hyperparameters are shared across all experiments.* means that for 2-ReLUmax we used a learning rate of 1×10^{-3} and minimum learning rate of 5×10^{-5} due to training instability.

Hyperparameter	**1 layer**	**8 layers**	**12 layers**
Optimization			
Batch size (total)	512	512	512
Learning rate	5×10^{-3}	5×10^{-3} *	1×10^{-3}
Min learning rate	1×10^{-4}	1×10^{-4} *	5×10^{-5}
Max iterations	80k	80k	80k
Weight decay	0.1	0.1	0.1
Adam β_1 / β_2	0.9 / 0.95	0.9 / 0.95	0.9 / 0.95
Gradient clipping	1.0	1.0	1.0
Learning rate schedule			
Scheduler		Cosine decay	
Warmup iterations		2k	
LR decay iterations		80k	
Model architecture			
Context length (n)		512	
Embedding dimension (d)		768	
Attention heads		12	
Persistent memory			
Memory size		2688	

Table 4: BabiStories dataset statistics [49].

Dataset partition	# Stories	# Tokens (GPT-2 tokenizer)	Avg. # Characters per story
Train	2.2M	474,704,907	888
Valid	2.2k	4,749,107	889

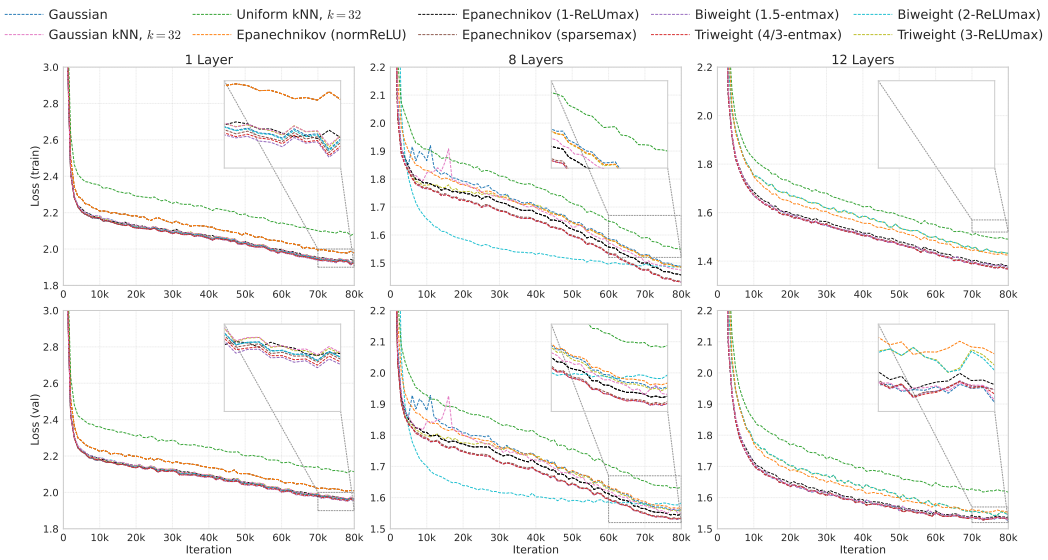


Figure 3: **Training and validation loss of Memory Mosaic models with different kernels.** Comparison on the BabiStories dataset for varying model depths. The horizontal axis shows the number of training iterations.

delimiter tokens. Models are evaluated on their ability to predict the last token of testing sequences generated from held-out PFAs, effectively measuring their capacity to generalize the underlying generative rules.

For Memory Mosaics, we perform a grid search over the Gaussian kernel hyperparameters, select the best configuration based on validation loss, and apply those hyperparameters to the remaining kernels. During the hyperparameter search for the Gaussian kernel variant, various configurations were tested, including different number of blocks and weight decay. Particular attention was given to the hyperparameters of the Memory Mosaics, such as the number of heads and embedding size. For kNN-based approaches, we tune $k \in \{16, 32, 64\}$ on 1k training examples and use that value ($k = 16$) for the remaining training datasets and for uniform kNN. All hyperparameters tested are shown in Table 5. These experiments were run on one single a6000 GPU.

Table 5: Hyperparameter space for the in-context learning experiment. Depth denotes the number of stacked blocks comprising persistent and contextual memories. Embedding size corresponds to the head dimension in transformers. We report results from the epoch achieving the lowest validation loss.

Parameter	**Range**
weight decay	$\{10^{-1}, 10^{-2}\}$
depth	{2, 4, 8}
embedding size	{32, 128, 256}
number of heads	{1, 2, 4, 8}
learning rate	$\{5 \times 10^{-4}\}$
maximum number of epochs	{200}
batch size	{32}

B.3 Length Generalization

For biological agents, episodic memory scales with the length of life rather than a fixed context window, placing memory retrieval inherently in the long-context regime and requiring continual length generalization. In such settings, dense aggregation over all stored memories would lead to severe interference, motivating the use of sparse retrieval mechanisms that selectively engage only a small subset of relevant memories. This perspective aligns with hippocampal indexing theory, where retrieval operates over sparse indices rather than dense memory traces [33, 11]. Inspired by this neurobiological constraint, we evaluate sparse attention mechanisms in a set of controlled synthetic tasks designed to isolate core sequence-modeling abilities. Specifically, we consider multi-query multi-token associative recall (MQMTAR), sequence reversal, and sequence sorting, which respectively probe long-range associative retrieval, sequential manipulation, and global reordering. To ensure comparability across tasks and focus on the effects of the attention mechanism itself, we fix all model hyperparameters to modest values rather than performing extensive task-specific tuning; the shared hyperparameters are summarized in Table 6. We show in Table 7 additional results for different k for the kNN variants. For the Transformer model, we use the GPT2 [24] architecture with the ALiBi [23] positional embedding. These experiments were run on A6000 GPUs.

Multi-query Multi-token Associative Recall (MQMTAR). The Multi-query Multi-token Associative Recall task evaluates a model's ability to store and retrieve multiple key–value associations within a single sequence and to answer several queries in parallel. The vocabulary consists of 256 integers, augmented with utility tokens {0, 1, 2} and additional special tokens used for structural delimitation. Each key k_i, value v_i, and query q_j is represented by a fixed-length sequence of two discrete tokens, corresponding to two numbers drawn from the base vocabulary. During input construction, multiple key–value pairs (k_i, v_i) are presented sequentially, followed by a set of four query keys $\{q_1, q_2, q_3, q_4\}$, all sharing the same two-token structure as the keys seen during the context. The model's objective is to retrieve, for each query q_j, the value v_i associated with the matching key k_i encountered earlier in the sequence. The target output is formed by concatenating the two-token representations of the four retrieved values in query order. This task jointly probes multi-token associative binding, robustness to interference across multiple stored associations, and the ability to perform accurate multi-query recall within a single forward pass.

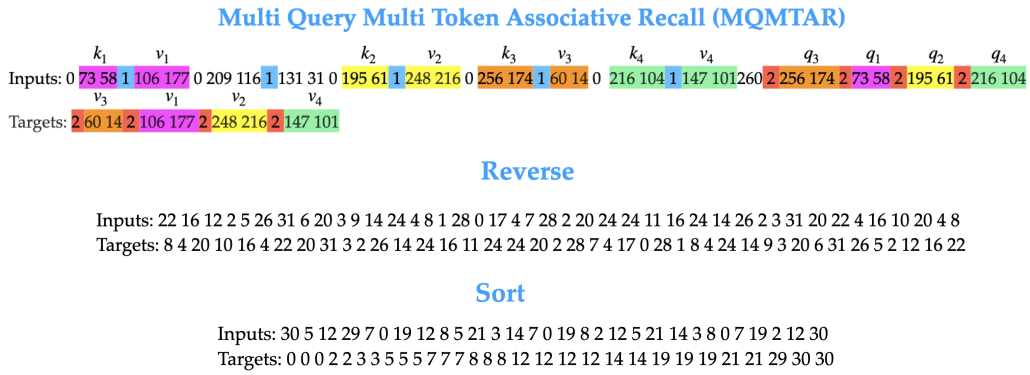


Figure 4: Illustration of the MQMTAR, reverse, and sort tasks. In the MQMTAR task, Soft colors indicate different keys, values, and queries for visual clarity. In the reverse task, the model receives a sequence of integers and must output them in reverse order. In the sort task, the model receives a sequence of integers and must output the sequence in ascending order.

Table 6: Hyperparameters used for MQMTAR recall, reverse, and sorting tasks. Hyperparameters were fixed to keep models simple and comparable.

Parameter	**MQMTAR**	**Reverse**	**Sorting**
Training examples	50×10^6	30×10^6	40×10^6
# epochs	1	1	1
Vocab size	263	36	36
Weight decay	10^{-1}	10^{-1}	10^{-1}
Depth	4	4	2
Embedding size	512	512	512
Number of heads	8	8	8
Learning rate	10^{-3}	10^{-3}	10^{-3}
Batch size	256	256	256
Training sequence length	up to 64	64	64

Reverse. The reverse task evaluates a model's ability to store and reproduce the elements of a discrete sequence in reverse order. The vocabulary consists of integers from 0 to 31, and additional special tokens used for structural delimitation.

During input construction, a sequence of integers $(x_1, x_2, \ldots, x_n)$ is presented to the model. The model's objective is to output the sequence in reverse order $(x_n, x_{n-1}, \ldots, x_1)$. This task probes the model's ability to memorize and manipulate sequential information over short-to-moderate context lengths.

Sort. The sort task evaluates a model's ability to correctly reorder a sequence of discrete tokens in ascending order. The vocabulary consists of integers from 0 to 31, including repetitions.

During input construction, a sequence of integers $(x_1, x_2, \ldots, x_n)$ is presented to the model. The model's objective is to output the sequence sorted in non-decreasing order $(y_1 \leq y_2 \leq \cdots \leq y_n)$. This task probes the model's ability to maintain and manipulate sequential information over short-to-moderate context lengths.

B.4 Efficiency Considerations

In standard dense softmax attention, each query distributes probability mass over all keys, resulting in computation and memory costs that scale linearly with the sequence length. In contrast, compact-support kernels and sparse attention transformations can assign exactly zero weight to many positions, meaning that only a subset of tokens participates in the weighted aggregation. This introduces the potential for reduced computational and memory overhead, particularly in long-context settings.

Table 7: Exact match accuracy on representative synthetic tasks. In-distribution results ($n = 64$) and out-of-distribution performance at increasing sequence lengths are reported. Values show the mean across three seeds, with the maximum across seeds indicated in superscript. Best average performance is in **bold** and overall maximum performance is underlined. L represents the number of layers.

	MQMTAR ($L = 4$)							Reverse ($L = 4$)		Sort ($L = 2$)		
Kernel	$1\times$	$2\times$	$4\times$	$8\times$	$16\times$	$32\times$	$64\times$	$1\times$	$1.5\times$	$1\times$	$2\times$	$4\times$
Gaussian kNN, $k = 16$	$\mathbf{0.99}^{(\underline{1.00})}$	$\mathbf{0.97}^{(\underline{1.00})}$	$\mathbf{0.93}^{(\underline{0.99})}$	$\mathbf{0.85}^{(\underline{0.96})}$	$0.65^{(0.78)}$	$0.25^{(\underline{0.36})}$	$\mathbf{0.05}^{(\underline{0.09})}$	$0.86^{(\underline{1.00})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{1.00}^{(\underline{1.00})}$	$0.74^{(0.89)}$	$0.00^{(0.00)}$
Gaussian kNN, $k = 32$	$\mathbf{0.99}^{(\underline{1.00})}$	$\mathbf{0.97}^{(\underline{1.00})}$	$\mathbf{0.93}^{(\underline{0.99})}$	$0.84^{(\underline{0.96})}$	$\mathbf{0.68}^{(\underline{0.85})}$	$\mathbf{0.27}^{(\underline{0.36})}$	$\mathbf{0.05}^{(0.05)}$	$\mathbf{0.99}^{(\underline{1.00})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{1.00}^{(\underline{1.00})}$	$\mathbf{0.80}^{(\underline{0.91})}$	$\mathbf{0.01}^{(\underline{0.02})}$
Uniform kNN, $k = 16$	$\mathbf{0.58}^{(\underline{0.71})}$	$\mathbf{0.01}^{(\underline{0.01})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$0.00^{(0.00)}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{1.00}^{(\underline{1.00})}$	$\mathbf{0.41}^{(\underline{0.63})}$	$\mathbf{0.00}^{(\underline{0.00})}$
Uniform kNN, $k = 32$	$0.04^{(0.08)}$	$\mathbf{0.00}^{(\underline{0.00})}$	$0.00^{(.00)}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{0.42}^{(\underline{0.66})}$	$\mathbf{0.00}^{(\underline{0.00})}$	$\mathbf{1.00}^{(\underline{1.00})}$	$0.00^{(0.00)}$	$\mathbf{0.00}^{(\underline{0.00})}$

Table 8: Exact match accuracy on the MQMTAR task. In-distribution results ($1\times$) and out-of-distribution performance at increasing sequence lengths are reported. Values show the mean across seeds, with the maximum indicated in superscript. Best average performance is in **bold** and overall maximum performance is underlined.

	MQMTAR						
Model	$1\times$	$2\times$	$4\times$	$8\times$	$16\times$	$32\times$	$64\times$
Transformer (ALiBi)	$\mathbf{1.00}^{(\underline{1.00})}$	$\mathbf{1.00}^{(\underline{1.00})}$	$0.99^{(0.99)}$	$0.09^{(0.15)}$	$0.01^{(0.01)}$	$0.00^{(0.00)}$	$0.00^{(0.00)}$
MM, Epanechnikov (ALiBi)	$0.99^{(\underline{1.00})}$	$0.96^{(\underline{1.00})}$	$0.92^{(\underline{1.00})}$	$0.85^{(\underline{1.00})}$	$0.72^{(0.94)}$	$0.43^{(0.71)}$	$0.14^{(0.29)}$
MM, Epanechnikov (no ALiBi)	$\mathbf{1.00}^{(\underline{1.00})}$	$\mathbf{1.00}^{(\underline{1.00})}$	$\mathbf{1.00}^{(\underline{1.00})}$	$\mathbf{0.98}^{(\underline{1.00})}$	$\mathbf{0.86}^{(\underline{0.98})}$	$\mathbf{0.63}^{(\underline{0.84})}$	$\mathbf{0.31}^{(\underline{0.51})}$

We note that the experiments in this paper operate in a short-context regime (512 tokens for the language modeling and up to a few thousands for the synthetic tasks) in which attention is not the dominant cost and the wall-clock gains of any fused attention kernel are modest [6]. Our focus is therefore on the modeling implications of compact-support kernels rather than on raw throughput.

At longer context lengths, however, the extent to which these theoretical benefits translate into practical speedups depends strongly on implementation. By the kernel regression correspondence established in this paper, the entire α-entmax family (e.g. Epanechnikov, biweight, triweight, ...) are fully compatible with Adasplash [10], which can match and even outperform FlashAttention [6] in high-sparsity regimes which naturally emerge at longer context lengths.

By contrast, combinatorial sparsification mechanisms such as top-k attention often require explicit selection or sorting operations, which are difficult to fuse into optimized GPU kernels and may limit practical efficiency despite inducing sparsity in the attention pattern.

C Additional Experiments

C.1 Sparse Memory Mosaics with Positional Embedding

To assess whether our gains stem from the proposed method rather than existing sparsity-inducing mechanisms, we compare against a standard Transformer with ALiBi [23], which introduces a positional bias favoring local attention.

The results shown in Table 8 show that, while ALiBi introduces a locality bias that encourages sparsity by prioritizing nearby tokens, it does not yield comparable performance. In contrast, our approach employs adaptive compact kernels to enable data-dependent sparsity, allowing the model to attend to relevant tokens irrespective of distance. As a result, Memory Mosaics maintains strong performance as context length increases, even without positional embedding, whereas the Transformer with ALiBi degrades rapidly.

C.2 Out-Of-Distribution

Out-of-distribution generalization on Simple English Wikipedia. Figure 5 compares the prediction performance of Memory Mosaics models trained on the BabiStories corpus and evaluated

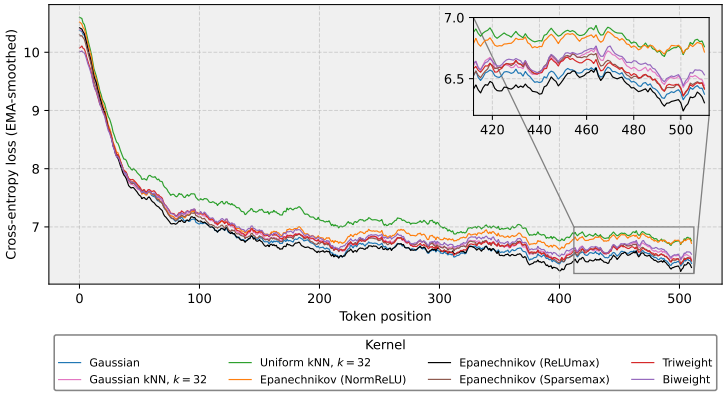


Figure 5: Prediction performance on the Simple English Wikipedia dataset using models trained on the BABISTORIES corpus. The plot reports the per-token average loss as a function of the token position within a 512-token input sequence.

out-of-distribution on Simple English Wikipedia[4]. The reported curves correspond to an exponential moving average (EMA) of the token-wise cross-entropy loss, computed over previously observed tokens, which provides a smoothed estimate of predictive confidence as the sequence unfolds.

At the beginning of each sequence, the model has limited contextual information and operates in a high-uncertainty regime. In this early phase, all attention variants exhibit similar behavior, with rapidly decreasing loss as additional context is incorporated, although higher-order compact kernels such as biweight and triweight typically start from slightly lower loss values. This reflects that, under uncertainty, strongly localized kernels may initially outperform dense Gaussian attention, which aggregates information more broadly.

As the sequence progresses and the model becomes more confident, clear differences emerge. Attention mechanisms based on compactly supported kernels—Epanechnikov, biweight, and triweight—achieve loss comparable to Gaussian attention, indicating that localized, sparse aggregation promotes selective, context-relevant memory retrieval in out-of-distribution settings. Among these, ReLUmax performs best overall: by combining sparse support with max anchoring, it avoids empty-support degeneracies while retaining locality, yielding stable and robust generalization as contextual evidence accumulates.

Impact Statement

This work advances the understanding of sparse attention mechanisms by framing them as kernel regression with compactly supported kernels. While we do not foresee ethical complications as a direct result of our method, we note that our contributions can be applied for training language models, and thus they inherit the broader ethical considerations associated with transformer-based language models. We emphasize responsible deployment, careful dataset curation, and energy-efficient training practices in such cases.

[4] `https://huggingface.co/datasets/rahular/simple-wikipedia`

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly and accurately state the paper's main contributions. The core claim, that sparse attention mechanisms can be understood as kernel regression with compact-support kernels, is explicitly presented in the abstract and then elaborated in the introduction. The introduction further lists concrete contributions (e.g., connections between sparsemax/α-entmax and Epanechnikov-family kernels, the unification of top-k methods, and the proposal of ReLUmax), which align with the theoretical developments in §3 and the proofs provided later (App. A).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: While the paper does not include a dedicated limitations section, we note drawbacks and limitations of specific methods, such as the instability of top-k attention in § 4.1 and the degeneracy issues in normalized ReLU in §3. We also discuss the limitations in the conclusion (§6), where we point out that while we experiment with models of various sizes, our experiments are all with relatively small models ($<$1B parameters). Further scaling experiments, beyond our compute budget, are needed to assess the benefits of compact kernels in larger models.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper presents its main theoretical results (e.g., Proposition 9 linking α-entmax to compact-support kernels) with clearly stated assumptions such as normalized keys/queries and kernel forms. The formal proof is provided in App. A, with the derivation steps shown in detail and consistent with the stated result.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides substantial details required for reproducibility. It specifies model architectures (Memory Mosaics), training procedures, hyperparameters (e.g., batch size, learning rates, optimizer settings), and datasets (e.g., BabiStories, RegBench, synthetic tasks), along with references. Additional implementation details are included in the appendix (§B) (e.g., training schedules, hyperparameter grids, and evaluation protocols), enabling a practitioner to replicate the experiments. While the code has not yet been released (promised upon acceptance), the methodological and experimental descriptions are sufficiently detailed to reproduce the main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code will be made available upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides all the necessary details for reproducibility. It specifies model architectures (Memory Mosaics), training procedures, hyperparameters (e.g., batch size, learning rates, optimizer settings), and datasets (e.g., BabiStories, RegBench, synthetic tasks), along with references and descriptions of how these datasets are constructed. Additional implementation details are provided in the appendix (§B) (e.g., training schedules, hyperparameter grids, and evaluation protocols), enabling practitioners to replicate the experiments. While the code has not yet been released (it will upon acceptance), the methodological and experimental descriptions are sufficiently detailed to reproduce the main results.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: While the paper does not include error bars, confidence intervals, or statistical significance tests, we ran experiments with multiple seeds. Table 2 reports averages over three seeds for all methods and reports the best results of each method across these seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We detail in the Appendix the type of compute workers used for each experiment. We further specify the global batch size for each experiment and discuss efficiency in §B.4 of all methods, as evaluating training times depends on several external factors.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification:

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We present a broader impact section in §C.2.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite all the papers on which we based our code, and in the future release of the code (upon acceptance), we cite the GitHub repositories on which we based our code.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [No]

Justification: LLMs are employed solely for tasks such as writing, editing, or formatting and does *not* affect the fundamental methodology, scientific rigor, or originality of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.