SPARSE ASSOCIATIVE MEMORIES THROUGH THE LENS OF COMPACT KERNEL REGRESSION

Saul Santos [1,2], Nuno Gonçalves [1,2,5], Daniel C. McNamee [3], Marcos Treviso [1,2], André F. T. Martins [1,2,4]
[1]Instituto de Telecomunicações, [2]Instituto Superior Técnico, Universidade de Lisboa,
[3]Champalimaud Research, [4]TransPerfect, [5]Carnegie Mellon University
{saul.r.santos,andre.t.martins}@tecnico.ulisboa.pt

ABSTRACT

Recent work has revealed a link between self-attention 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. Motivated by neuroscience, where sparse hippocampal activity suggests memory retrieval is driven by only a small number of highly weighted similarities among well-separated memories, we develop a formal correspondence between sparse attention and compact kernels. We show that normalized ReLU and sparsemax attention arise from Epanechnikov kernel regression under fixed and adaptive normalizations, respectively. More generally, 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 as $n \to \infty$. This unified perspective explains how sparsity naturally arises from kernel design and provides principled alternatives to heuristic top-k attention and other associative memory mechanisms. Experiments with sparse kernel-regression-based variants of transformers—Memory Mosaics—show competitive performance on language modeling, in-context learning, and length generalization tasks.

1 INTRODUCTION

Self-attention is a central component of transformer architectures, enabling the modeling of long-range dependencies in sequences (Vaswani et al., 2017). A growing body of work has connected attention-based in-context learning to associative memory mechanisms, including test-time regression or memory-based computation (Wang et al., 2025; Behrouz et al., 2024; Sun et al., 2025, *inter alia*). A key insight from this literature is that standard softmax attention can be interpreted as *Gaussian kernel regression* over the key-value cache, a perspective that has motivated architectures such as *Memory Mosaics* (Zhang et al., 2025; Zhang & Bottou, 2025). However, this raises a natural question: *is the Gaussian kernel—and softmax attention—the best design choice?* In particular, the dense weighting induced by softmax has been shown to cause *attention dispersion* (Veličković et al., 2025).

In neurobiological systems such as humans, the problem of storing and retrieving sequences with long-range dependency structure arises in episodic memory (Tulving, 2002). 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 (Teyler & DiScenna, 1986; Goode et al., 2020). Notably, a core characteristic of hippocampal population activity is *sparsity*. This property of hippocampal activity is often attributed to its functional role of pattern separation Yassa & Stark (2011). From the perspective of the current work, hippocampal indices serve as keys, which can be queried, to neocortical values (Gershman et al., 2025). Thus, hippocampal sparsity can be understood as a mechanism by which attentional dispersion (Veličković et al., 2025) is avoided in retrieving neocortical memory traces. Specifically, 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 (Tulving, 2002).

Building on these insights, our work provides a unified kernel regression perspective on both dense and sparse attention mechanisms. We show that a range of attention transformations—including softmax,

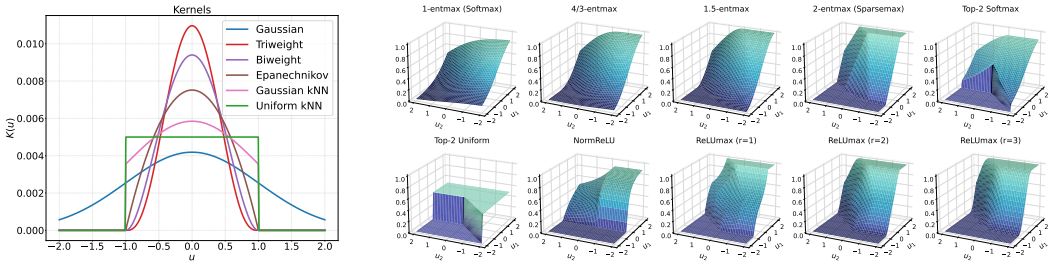

Figure 1: **Classical kernels and attention mechanisms.** Left: normalized one-dimensional kernels on $\boldsymbol{u} \in \mathbb{R}^n$. Right: attentions on $\boldsymbol{u} = [0, u_1, u_2]$. Softmax ($\alpha = 1$) corresponds to a Gaussian kernel. Normalized ReLU, sparsemax ($\alpha = 2$), and 1-ReLUmax correspond to the Epanechnikov kernel. $4/3$- and 1.5-entmax, as well as 2- and 3-ReLUmax, correspond to the biweight and triweight kernels. Uniform kNN yields a top-k uniform kernel, while top-k softmax yields a truncated Gaussian.

sparsemax, α-entmax, top-k uniform, top-k softmax, normalized ReLU, and a new transformation, r-ReLUmax—can be interpreted as Nadaraya-Watson estimators with different kernel functions.

2 Background

α-entmax transformation. A generalization of softmax is given by the α-entmax transformation (Peters et al., 2019), 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 $\sum_i p_i = 1$. 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* (Martins & Astudillo, 2016).

Attention and kernel regression. 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* (Nadaraya, 1964; Watson, 1964) 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 $\int_{\mathbb{R}^d} K(\boldsymbol{u}) d\boldsymbol{u} = 1$.

Let $\boldsymbol{K} \in \mathbb{R}^{(n-1)\times d}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times d}$ denote stacked keys and values. Furthermore, consider ℓ_2-normalized queries and keys, $\|\boldsymbol{q}\| = \|\boldsymbol{k}_i\| = 1$, which allows us to relate dot-product attention to Euclidean distances. Now, 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

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

This expression recovers softmax attention where the kernel bandwidth h corresponds to the square root of the temperature. The standard transformer scaling $\boldsymbol{k}_i^\top \boldsymbol{q}/\sqrt{d}$ therefore fixes an effective bandwidth $h = d^{1/4}$. This perspective raises the question: *what if we use other kernel functions?*

Beyond Gaussian kernels, attention can be derived from *compact-support* kernels, which assign zero weight beyond a fixed distance, promoting locality and sparsity. A convenient family is the r-order *rectified polynomial kernel*: $K(\boldsymbol{u}) \propto [1 - \|\boldsymbol{u}\|^2]_+^r$. Different choices of r yield familiar kernels: $r = 1$ gives the *Epanechnikov* kernel (Epanechnikov, 1969), $r = 2$ the *biweight* kernel, and $r = 3$ the *triweight* kernel (Scott, 2015). All assign zero weight outside the unit ball, enforcing sparsity, as illustrated in Fig. 1. Another way to enforce compact support is via truncated kernels. Both a

Gaussian restricted to the k nearest neighbors and a uniform kernel assign zero weight beyond a threshold $\delta_{\max}$, enforcing locality and sparsity:

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

We will show in §3, how these kernels induce different sparse attention mechanisms.

3 Sparse Attention via Compact Kernels

We formalize the connection between sparse attention and kernel regression with compact-support kernels. Starting from the kernel view in §2, we show how replacing the Gaussian kernel in softmax attention with compact-support kernels induces locality and sparsity.

Normalized ReLU and Epanechnikov kernel. Replacing the Gaussian kernel with the Epanechnikov kernel, and defining $\gamma = h^2/2$ and $b = 1 - 2/h^2$, yields the following query-key interaction and normalized attention rule:

$$K(\boldsymbol{k}_i - \boldsymbol{q}) = \big[\boldsymbol{k}_i^\top \boldsymbol{q}/\gamma + b\big]_+ \qquad \Longrightarrow \qquad \hat{\boldsymbol{v}}_n(\boldsymbol{q}) = \sum_{i=1}^{n-1} \frac{\big[\boldsymbol{k}_i^\top \boldsymbol{q}/\gamma + b\big]_+}{\sum_{j=1}^{n-1} \big[\boldsymbol{k}_j^\top \boldsymbol{q}/\gamma + b\big]_+} \boldsymbol{v}_i. \tag{5}$$

Thus, normalized ReLU attention is equivalent to Epanechnikov kernel regression. This was also noted by Hoover et al. (2025) in Hopfield-like energy functions. A limitation of (5) is that attention is undefined when all pre-activations are negative, yielding a $0/0$ indeterminacy. We next present two alternatives: (i) *auto-normalization* via adaptive bandwidth and (ii) *support anchoring* at nearest keys.

3.1 Auto-normalization (Adaptive Bandwidth)

An alternative to the normalization in (5) 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.

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

$$\sum_{i=1}^{n-1} \left[1 - \frac{\|\boldsymbol{k}_i - \boldsymbol{q}\|^2}{h^2}\right]_+^r = \left(\frac{2r\gamma}{h^2}\right)^r \sum_{i=1}^{n-1} \left[\frac{h^2}{2r\gamma} - \frac{1}{r\gamma} + \frac{\boldsymbol{k}_i^\top \boldsymbol{q}}{r\gamma}\right]_+^r. \tag{6}$$

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,

$$\sum_{i=1}^{n-1} \left[\frac{\boldsymbol{k}_i^\top \boldsymbol{q}}{r\gamma} - \underbrace{\left(\frac{1}{r\gamma} - \frac{h^2}{2r\gamma}\right)}_{:=\tau}\right]_+^r = 1 \quad \Longrightarrow \quad \hat{\boldsymbol{v}}_n(\boldsymbol{q}) = \sum_{i=1}^{n-1} \alpha\text{-entmax}_i\left(\frac{\boldsymbol{K}\boldsymbol{q}}{\gamma}\right) \boldsymbol{v}_i, \tag{7}$$

with an adaptive bandwidth $h = \sqrt{2 - 2r\gamma\tau}$. See §B for the derivation. The normalizer $\tau = \frac{1}{r\gamma} - \frac{h^2}{2r\gamma}$ coincides with the α-entmax threshold in (1), recovering the corresponding attention transformation (see Fig. 1). In particular, $r = 1$ recovers sparsemax with the Epanechnikov kernel, $r = 2$ gives 1.5-entmax with the biweight kernel, and $r = 3$ yields $\frac{4}{3}$-entmax with the triweight kernel, reproducing sparse associative memories studied in Hopfield networks (Wu et al., 2024; Santos et al., 2024; 2025).

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).

Table 1: **Performance across training set sizes** 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 ↓
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 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

$$\hat{\boldsymbol{v}}(\boldsymbol{q}) = \sum_{i\in\mathcal{N}_k(\boldsymbol{q})} \frac{K_{\text{top-}k}(\boldsymbol{k}_i - \boldsymbol{q})}{\sum_{j\in\mathcal{N}_k(\boldsymbol{q})} K_{\text{top-}k}(\boldsymbol{k}_j - \boldsymbol{q})}\, \boldsymbol{v}_i. \tag{8}$$

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 (Gupta et al., 2021).

r-ReLUmax. We introduce the *r-ReLUmax* family of *anchored polynomial kernels*, which enforce compact support while anchoring at the highest-scoring key. Let $m := \max_j \boldsymbol{k}_j^\top \boldsymbol{q}$ and $b > 0$ be a hyperparameter. The r-ReLUmax kernel and the corresponding Nadaraya-Watson estimate are

$$K_{r\text{-relumax}}(\boldsymbol{k}_i - \boldsymbol{q}) \propto \left[b + \frac{\boldsymbol{k}_i^\top \boldsymbol{q} - m}{h^2}\right]_+^r \quad\Longrightarrow\quad \hat{\boldsymbol{v}}(\boldsymbol{q}) = \sum_{i=1}^{n-1} \frac{\left[b + \frac{\boldsymbol{k}_i^\top \boldsymbol{q} - m}{h^2}\right]_+^r}{\sum_{k=1}^{n-1}\left[b + \frac{\boldsymbol{k}_k^\top \boldsymbol{q} - m}{h^2}\right]_+^r}\, \boldsymbol{v}_i. \tag{9}$$

The parameter r controls the sharpness of the kernel, with the familiar Epanechnikov ($r = 1$), biweight ($r = 2$), and triweight ($r = 3$) anchored polynomial kernels recovered as special cases. Because the kernel is anchored at the maximum score m, at least one weight is strictly positive, so the denominator cannot vanish. Hence, r-ReLUmax guarantees nondegenerate normalization while smoothly adapting the set of active keys to the score distribution, preserving compact support.

4 Experiments

4.1 In-Context Learning

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 (Akyürek et al., 2024)Our models match the sequence length and vocabulary size of Akyürek et al. (2024) and follow the same hyperparameter tuning strategy. Full experimental details are provided in §C.1. 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.

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) consistently match or slightly outperform the Gaussian kernel in the low-data regime (1k–5k),

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

achieving both higher accuracy and lower TVD. Within the Epanechnikov family, sparsemax performs best overall, followed closely by 1-ReLUmax in terms of accuracy and TVD. 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 top-k uniform attention underperforms across all data scales. 2- and 3-ReLUmax show competitive behavior when compared with adaptive bandwidth methods.

4.2 Length Generalization

We evaluate the length generalization capabilities of our Memory Mosaics using synthetic tasks, following the setup of Vasylenko et al. (2025), 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 §C.2.

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 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 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, benefitting from its max-anchoring operation, while 2- and 3-ReLUmax show competitive performance with the corresponding α-entmax models.

5 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; 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.

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), by the Portuguese Recovery and Resilience Plan through project C645008882-00000055 (Center for Responsible AI), and by FCT/MECI through national funds and when applicable co-funded EU funds under UID/50008: Instituto de Telecomunicações.

REFERENCES

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms, 2024. URL https://arxiv.org/abs/2401.12973.

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

V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969. doi: 10.1137/1114019. URL https://doi.org/10.1137/1114019.

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

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. doi: 10.1016/j.neuron.2020.07.011. Epub 2020 Aug 6.

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 (eds.), *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pp. 39–52, Virtual, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.sustainlp-1.5. URL https://aclanthology.org/2021.sustainlp-1.5/.

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

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 (eds.), *International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1614–1623, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/martins16.html.

E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964. doi: 10.1137/1109020. URL https://doi.org/10.1137/1109020.

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*, pp. 1504–1519, 2019.

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. URL http://jmlr.org/papers/v26/24-1961.html.

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.

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

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.

T. J. Teyler and P. DiScenna. The hippocampal memory indexing theory. *Behavioral Neuroscience*, 100(2):147–154, 1986. doi: 10.1037/0735-7044.100.2.147.

Endel Tulving. Episodic memory: From mind to brain. *Annual Review of Psychology*, 53:1–25, 2002. doi: 10.1146/annurev.psych.53.100901.135114.

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. URL https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Pavlo Vasylenko, Hugo Pitorro, André F. T. Martins, and Marcos Treviso. Long-context generalization with sparse attention, 2025. URL https://arxiv.org/abs/2506.16640.

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. URL https://openreview.net/forum?id=S4JmmpnSPy.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory, 2025. URL https://arxiv.org/abs/2501.12352.

Geoffrey S. Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 26(4):359–372, 1964. ISSN 0581572X. URL http://www.jstor.org/stable/25049340.

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.

Michael A. Yassa and Craig E. L. Stark. Pattern separation in the hippocampus. *Trends in Neurosciences*, 34(10):515–525, 2011. doi: 10.1016/j.tins.2011.06.006.

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

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 (eds.), *International Conference on Representation Learning*, volume 2025, pp. 36412–36433, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/59c3ac496e6fe7be0c2c2b95014e2210-Paper-Conference.pdf.

A Appendix

B Main Results

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:

$$K_h(\boldsymbol{k}_i - \boldsymbol{q}) = \left[1 - \frac{\|\boldsymbol{k}_i - \boldsymbol{q}\|^2}{h^2}\right]_+^r \tag{10}$$

$$= \left[1 - \frac{2(1 - \boldsymbol{k}_i^\top \boldsymbol{q})}{h^2}\right]_+^r \tag{11}$$

$$= \left[\frac{h^2 - 2 + 2\boldsymbol{k}_i^\top \boldsymbol{q}}{h^2}\right]_+^r. \tag{12}$$

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 (2) can be expressed as

$$\sum_{i=1}^{n-1} \left[1 - \frac{\|\boldsymbol{k}_i - \boldsymbol{q}\|^2}{h^2}\right]_+^r = \left(\frac{2r\gamma}{h^2}\right)^r \sum_{i=1}^{n-1} \left[\frac{h^2}{2r\gamma} - \frac{1}{r\gamma} + \frac{\boldsymbol{k}_i^\top \boldsymbol{q}}{r\gamma}\right]_+^r. \tag{13}$$

Choosing h such that

$$\sum_{i=1}^{n-1} \left[\frac{\boldsymbol{k}_i^\top \boldsymbol{q}}{r\gamma} - \underbrace{\left(\frac{1}{r\gamma} - \frac{h^2}{2r\gamma}\right)}_{:=\tau}\right]_+^r = 1, \tag{14}$$

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

$$\hat{\boldsymbol{v}}_n(\boldsymbol{q}) = \sum_{i=1}^{n-1} \alpha\text{-entmax}_i\left(\frac{\boldsymbol{K}\boldsymbol{q}}{\gamma}\right) \boldsymbol{v}_i, \tag{15}$$

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

C Experimental Details

C.1 In-Context Learning

For evaluation, we use RegBench (Akyürek et al., 2024). 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 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 3.

Table 3: 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}

C.2 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 (Teyler & DiScenna, 1986; Goode et al., 2020). 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 4. We show in Table 5 additional results for different k for the kNN variants.

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.

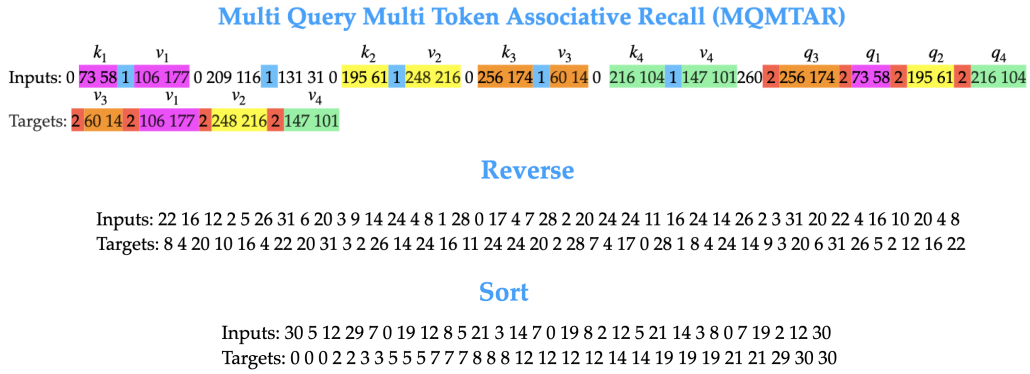

Figure 2: Illustration of the MQMTAR, reverse, and sort tasks. In the MQMTAR task, Soft colors indicate different keys, values, and queries for visual clarity. In the reverse task, the model receives a sequence of integers and must output them in reverse order. In the sort task, the model receives a sequence of integers and must output the sequence in ascending order.

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.

Table 4: 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

Table 5: 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, $k = 16$	**0.99**$^{(1.00)}$	**0.97**$^{(1.00)}$	**0.93**$^{(0.99)}$	**0.85**$^{(0.96)}$	$0.65^{(0.78)}$	$0.25^{(0.36)}$	**0.05**$^{(0.09)}$	$0.86^{(1.00)}$	**0.00**$^{(0.00)}$	**1.00**$^{(1.00)}$	$0.74^{(0.89)}$	$0.00^{(0.00)}$
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, $k = 16$	**0.58**$^{(0.71)}$	**0.01**$^{(0.01)}$	**0.00**$^{(0.00)}$	**0.00**$^{(0.00)}$	**0.00**$^{(0.00)}$	**0.00**$^{(0.00)}$	**0.00**$^{(0.00)}$	$0.00^{(0.00)}$	**0.00**$^{(0.00)}$	**1.00**$^{(1.00)}$	**0.41**$^{(0.63)}$	**0.00**$^{(0.00)}$
Uniform kNN, $k = 32$	$0.04^{(0.08)}$	**0.00**$^{(0.00)}$	$0.00^{(.00)}$	**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**$^{(0.00)}$

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.