# ON LEARNING EXPAND-AND-SPARSIFY REPRESENTATIONS

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

## ABSTRACT

Expand-and-sparsify (EaS) representations lift an input to a much higher dimension through a linear map and keep only the $k$ largest coordinates. Existing theory builds that map at random and needs the expanded dimension to grow exponentially in the input dimension, which puts the construction out of practical reach. We ask what is gained by learning the map by gradient descent instead, and how that gain depends on how wide the expansion is. We sweep 73 OpenML classification tasks and 48 regression tasks under a protocol that holds the datasets, expanded dimensions, sparsity levels and splits fixed across every method, and we summarise each comparison by a median over datasets and a Wilcoxon signed rank test. We report four findings. First, learning the expansion helps most when the expansion is narrowest, and its advantage over a random map decays to nothing by the widest expansions we run, so a learned EaS layer can be made small. Second, with the map learned, a top-$k$ representation that keeps at most a tenth of its coordinates is indistinguishable from a dense ReLU layer of the same expanded dimension at every expansion factor we test, while GELU and SiLU are not. Third, when the representation is forced to be binary the sparse code is the better of the two, and the reason is that binarization costs it nothing while it costs the dense layer accuracy. Fourth, how sparse the representation can be is governed by how wide it is: the sparsity level that is free rises as the expansion grows, so there is no fixed limit on how sparse it can be. The regression tasks reproduce the first finding and qualify the second and third.

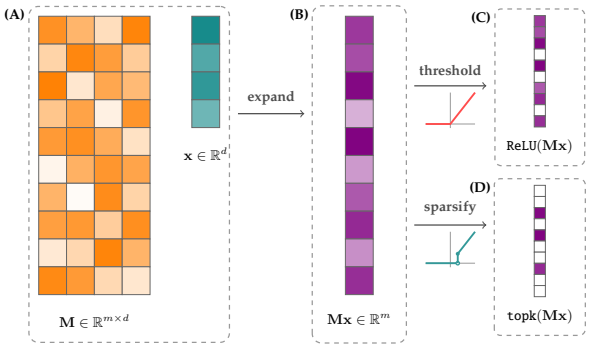
## 1 INTRODUCTION

*Expand-and-sparsify* or EaS representations are generated by projecting inputs in $\mathbb{R}^d$ to a much larger dimensionality $\mathbb{R}^m$ with $m \gg d$, followed by a sparsification step which zeros out all but the top $k$ values in the $m$-dimensional representation, with $k \ll m$. This scheme is neuro-biologically inspired, where the top-$k$ sparsification is achieved via inhibitory signals common in biological systems. For example, a version of this is seen in the olfactory circuit of fruit-flies (Turner et al., 2008; Caron et al., 2013), olfactory circuit of rodents, the visual cortex of cats, and the electrosensory system of electric fish (Stettler & Axel, 2009; Olshausen & Field, 2004; Chacron et al., 2011). This representation scheme has been used for locality sensitive hashing (Dasgupta et al., 2017) and outlier detection (Dasgupta et al., 2018). Subsequent theoretical studies have shown that a linear model on top of these representations can approximately express any sufficiently smooth function (Dasgupta & Tosh, 2020; Sinha, 2024).

While these representations come with interesting theoretical properties, their practical viability has not been established to the best of our knowledge. The theoretical studies rely on expansions via prescribed constructions, and require $m \sim O(\exp(d))$, thus limiting practical use. Learned expansions for supervised problems via stochastic gradient descent (SGD) and backpropagation with more practical values of $m \gg d$ have not been explored. Expanded representations (with more practical values of $m \geq d$) are often employed in the single-hidden-layer feedforward networks, serving as *channel mixers* in architectures such as the Transformer (Vaswani et al., 2017) or the MLP-mixer (Tolstikhin et al., 2021), and we think it is important to understand the capabilities of (learned) EaS representations relative to the commonly used learned unrestricted expanded representations.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.




(E)

| ReLU | Wins/Ties/Losses | Significant at |
|---|---|---|
| vs GELU | 302/56/186 | 5 of 8 |
| vs SiLU | 333/48/163 | 6 of 8 |
| vs topk (EaS) | 253/91/200 | 0 of 8 |
| vs LWTA@k (EaS) | 257/94/193 | 0 of 8 |

(F)

| Learned $M$ | Wins/Ties/Losses | Significant at |
|---|---|---|
| vs Random $M$ | 361/86/97 | 7 of 8 |
| vs Data-supp. $M$ | 367/62/112 | 8 of 8 |

Figure 1: **What we compare, and what we find.** **(A)** A linear operator $\mathbf{M} \in \mathbb{R}^{m \times d}$ is applied to the input $\mathbf{x} \in \mathbb{R}^d$, giving us **(B)** an expanded representation $\mathbf{Mx} \in \mathbb{R}^m$, which is then either **(C)** thresholded, giving a dense `ReLU` layer, or **(D)** sparsified, giving the `EaS` representation. **(E)** The table compares `ReLU` against alternate activations with $\mathbf{M}$ learned throughout — `GELU` and `SiLU` are other dense activations, and `topk` and `LWTA@k` serve as `EaS` versions with different sparsification schemes. **(F)** The table compares a learned $\mathbf{M}$ against theoretical constructions from Dasgupta & Tosh (2020) with `topk` activation throughout. We summarize the results in the form of wins/ties/losses over the (dataset, $m/d$) configurations, and "significant at" counts how many of the 8 expansion factors reach $p < 0.05$ under a two-sided Wilcoxon signed rank test (per-expansion factor results are in fig. 4 and table 1; more details are presented in section E and section F). The results show that the learned `EaS` representations not only *significantly improve upon the theoretical constructions*, but also provide *competitive performance to standard activations used in feedforward networks*.

Consequently, we believe that `EaS` can enable unique advantages in practical architectures. Because the expanded representation is extremely ($k \ll m$) and predictably (always $k$ nonzeros) sparse, (i) we can use sparse linear algebra kernels when available, (ii) we need only load the rows of the next layer's weights that correspond to the $k$ nonzeros, (iii) we need only backpropagate through those same $k$ nodes (Gupta et al., 2021), and (iv) any cache of intermediate activations (Grave et al., 2017; Wimbauer et al., 2024) shrinks by the same factor. Section B works through what each of these is worth.

**Research question.** Given these motivations for `EaS` representations, here we focus on understanding whether learning the `EaS` representations via standard SGD (i) not only improves over the existing theoretical `EaS` constructions, but also (ii) produces representations that are competitive with existing unrestricted expanded representations in terms of expressivity and generalization.

**Findings.** We consider an evaluation across 73 OpenML classification tasks (binary and multi-class) and 48 OpenML regression tasks, controlling for the expanded dimension (with the expansion factor $m/d \in [1, 256]$) and the sparsity ($k \leq 0.1m$), training over roughly 50,000 configurations (more than 160,000 training runs across three train/validation/test splits for each configuration). [1] Every number below is a median over datasets of the per-dataset difference, with significance from a two-sided Wilcoxon signed rank test. We make the following observations (see summary in fig. 1).

- Learned `EaS` representations outperform the theoretical constructions, and by the largest margin where the expansion is narrowest. That learning helps is expected, since only it sees the labels; the useful part is the decay. Against a data-independent construction the median gain falls from 3.0 accuracy points at $m = d$ to 0.13 at $m = 256d$, significant up to $m = 128d$ ($p \leq 0.006$) but not beyond. Regression behaves the same way, with a median gain of 0.185 in $R^2$ at $m = d$. Against the data-supported construction the gain does *not* decay: it stays between 0.65 and 1.68 points across the whole range and is significant at every expansion factor ($p \leq 0.0011$). A learned `EaS` representation can therefore be made small, tolerating smaller values of $m \geq d$.

[1] Note that the existing theory of `EaS` representations, either in regression for arbitrarily accurate function approximation (Dasgupta & Tosh, 2020) or in classification for a minimax-optimal convergence rate (Sinha, 2024), requires $m \sim O(\exp(d))$, and in the classification setting the existing lower bound on $k$ is $\Omega(d \log m)$. In our setting $m$ is not scaled exponentially in $d$, so we instead work in an artificially restricted high-sparsity regime of $k \leq 0.1m$. See section H for the theoretical guarantees achievable in this regime of $(m, k)$.

- Learned EaS representations at $k \leq 0.1m$ are competitive with unrestricted ones. Against a ReLU layer of the same expanded dimension we see no significant difference at *any* expansion factor, with the median difference between $-0.22$ and $+0.05$ points and $p \geq 0.177$ throughout. This is not something every alternative achieves: GELU and SiLU both fall significantly short of ReLU over most of the range. On regression the same parity holds for $m \geq 4d$; below that the EaS representation trails ReLU significantly, by a median $0.04$ in $R^2$ at $m = d$ and $0.03$ at $m = 2d$.
- If the representation is restricted to be binary, the learned EaS representation is the better of the two from $m = 16d$ onward, by $0.50$ to $0.62$ accuracy points. Comparing each activation against its own value-passing form shows why: binarizing topk costs nothing at any expansion factor, while binarizing ReLU costs $0.4$ points at $m \geq 128d$. The sparse code is not gaining, it is simply not losing, since it already carries its information in which coordinates fire rather than how strongly. On regression binarizing costs every activation something, but it costs topk less than ReLU, so the binarized EaS representation is again ahead, at $m = 16d$ to $128d$.
- How sparse the representation can be is governed by how large the expanded dimension is. At $90\%$ sparsity it matches ReLU everywhere; at $99\%$ it falls behind, but the deficit shrinks monotonically from $2.4$ points at $m = 4d$ to $0.35$ at $m = 256d$. There is no single sparsity level at which EaS begins to lose; instead, the larger the expanded dimension, the sparser the representation can be before it costs accuracy.

Together these indicate that the practical benefits of EaS representations described above may be available without losing much in expressivity or generalization, which makes a case for their wider use in modern architectures.

**Related work.** EaS representations were introduced as a model of the fly olfactory circuit and used for similarity search (Dasgupta et al., 2017) and novelty detection (Dasgupta et al., 2018), and later applied to supervised classification (Sinha & Ram, 2021; Ram & Sinha, 2022). Their expressivity has been studied theoretically by Dasgupta & Tosh (2020), Sinha (2024) and Sinha & Tosh (2026), all of whom construct the expansion by sampling its rows from a fixed distribution rather than by learning them. Two lines of work do learn the projection: Ryali et al. (2020) train it with an unsupervised, biologically motivated rule for similarity search, and Liang et al. (2021) train the same circuit on text to produce word embeddings. Both learn the expansion, but neither measures the resulting representation against a dense layer of the same expanded dimension under a fixed sparsity budget, which is the comparison we make. Sparsification by competition among units, without the expansion, was studied by Srivastava et al. (2013), and top-$k$ activations have been used inside the feedforward block of a Transformer to reduce the memory cost of backpropagation (Gupta et al., 2021). We further elaborate on how our setting relates to each of these in section A, including the sense in which Dasgupta & Tosh (2020, §5) already consider a data-dependent expansion.

## 2 Problem and Experimental Setup

**The model.** We consider supervised learning problems, given training data $S = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, n\}$ with input features $\mathbf{x}_i \in \mathbb{R}^d$ and the output $\mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^l$, and a model of the form

$$f(\mathbf{x}; \mathbf{W}, \mathbf{M}) = \mathbf{W}\mathbf{z}, \quad \mathbf{z} \triangleq \sigma(\mathbf{M}\mathbf{x}) \in \mathbb{R}^m, \quad \mathbf{M} \in \mathbb{R}^{m \times d}, \mathbf{W} \in \mathbb{R}^{l \times m}. \tag{1}$$

Here $\mathbf{M}$ is the expansion matrix, $\sigma$ is a nonlinear activation, $\mathbf{W}$ is the linear readout, and $(\mathbf{W}, \mathbf{M})$ are the learnable parameters. If $l$ is the number of classes, $f$ is a multi-class classifier; if $l = d$, this model is the feedforward channel mixer used in the Transformer and the MLP-Mixer. With $\sigma$ a standard activation such as ReLU, eq. (1) is a single-hidden-layer feedforward network; with $\sigma$ the topk function we get $\|\mathbf{z}\|_0 = k$, which is the EaS representation. We can further restrict the representation to be binary, $\mathbf{z} \in \{0, 1\}^m$, in which case a standard activation reduces to the Heaviside function while the EaS representation has exactly $k$ ones and $(m - k)$ zeros. Given a loss $\ell$, we learn $(\mathbf{W}, \mathbf{M})$ by minimizing the population loss $\sum_{i=1}^{n} \ell(\mathbf{y}_i, \mathbf{W} \cdot \sigma(\mathbf{M}\mathbf{x}_i))$ with SGD, for different activations and different values of $m$ and $k$. With theoretical constructions of $\mathbf{M}$, we learn only $\mathbf{W}$.

**Constructions of the expansion matrix.** We compare three ways of obtaining the expansion matrix $\mathbf{M}$ (fig. 2), holding everything else fixed. In the *learned* construction, $\mathbf{M}$ is trained by SGD jointly with the readout $\mathbf{W}$. In the *data-supported* construction, the $m$ rows of $\mathbf{M}$ are drawn from the training inputs with replacement, so that the expansion directions lie in the support of the data, which

is the setting Dasgupta & Tosh (2020, §5) analyze. In the *data-independent* construction, the rows are drawn i.i.d. from a fixed distribution (the uniform distribution on the unit sphere in $\mathbb{R}^d$, drawn by normalizing samples from $\mathcal{N}(0, \boldsymbol{I}_d)$). In all three cases, $\mathbf{W}$ is fitted by SGD on all $n$ training points.

Drawing with replacement matters for two reasons. It keeps the construction defined for every dataset at every expanded dimension, including $m > n$, so the three constructions are compared on the same datasets throughout. And it is i.i.d. sampling from the empirical distribution, whereas drawing $m$ distinct points is not i.i.d. at all. The cost is that $M$ can hold at most $n$ distinct rows, which section 3.1 shows is what eventually limits this construction.

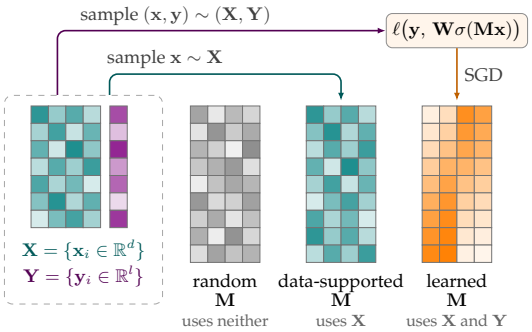



Figure 2: The three constructions of $\mathbf{M}$ and what each may use. The random (data-independent) $\mathbf{M}$ uses neither $\mathbf{X}$ nor $\mathbf{Y}$, the data-supported $\mathbf{M}$ samples its rows from $\mathbf{X}$, and the learned $\mathbf{M}$ is trained by SGD on a loss that sees both.

**Activations.** The sparse activations are `topk` and local winner-take-all (`LWTA@k`), which splits the representation into blocks of size $(m/k)$ and selects the top-1 in each block (Srivastava et al., 2013), resulting in one nonzero per block; with blocks of size $\mathrm{round}(m/k)$ that is exactly $k$ nonzeros when $k$ divides $m$ and can be slightly more otherwise (section F). We treat `LWTA@k` as an alternative sparsification scheme within `EaS` rather than as a separate method: both keep (about) $k$ of the $m$ coordinates and so carry the same sparsity budget, and they differ only in which $k$ survive. The `topk` activation lets all $m$ units compete, while `LWTA@k` lets them compete only inside blocks of size $(m/k)$. For any single input the two coincide under a permutation of the rows of $\mathbf{M}$ that puts the $k$ winners in distinct blocks, and when $\mathbf{M}$ is learned the assignment of directions to blocks is learned along with the directions, so the network can partly arrange that for itself. We therefore expect the two to behave similarly, and section 3.2 finds that they do. We run each of them in a value-passing form, which forwards the surviving pre-activations, and in a binary form, which forwards their indicator and is trained through a straight-through estimator. The dense baselines are `ReLU`, `GELU` and `SiLU`. Binarizing any of these three gives the same function, the indicator of a positive pre-activation, so there is one dense binary baseline rather than three and we run only that one. The binary forms are evaluated against each other in section 3.3 and are not mixed into the comparisons among value-passing activations.

**Parameterization.** We report results as functions of the expansion factor $m/d$ and the sparsity level $k/m$ rather than of $m$ and $k$. These are equivalent, but the ratios are comparable across datasets of differing dimension, and $k$ on its own cannot be separated from $m$ in our grid: since $k = (k/m) \cdot (m/d) \cdot d$, a large absolute $k$ arises only at a large expanded dimension. We sweep $m/d \in \{1, 2, 4, 8, 16, 64, 128, 256\}$ and $k/m \in \{0.001, 0.01, 0.03, 0.05, 0.10\}$, restricting throughout to $k \leq 0.1m$.

**Data.** We use 73 classification tasks from OpenML, admitted by input dimension $d \in [8, 120]$ and sample size $n \in [300, 20000]$, with missing values imputed and categorical features one-hot encoded. Of these, 37 are binary and 36 are multi-class. We use a further 48 regression tasks admitted by the same criteria. We deliberately chose a large and varied set of tasks so that we can see how generally the behaviour of `EaS` holds.

**Protocol.** Every construction sees identical datasets, expanded dimensions, sparsity levels and train/test splits, so all comparisons are paired. We repeat every configuration on 3 random train/validation/test splits (70/10/20, stratified for classification), with a different seed for each, and report the mean test score. We train with Adam at a learning rate of $10^{-3}$, no weight decay and a batch size of 256, for up to 300 epochs with early stopping at a patience of 30. For each (dataset, $m/d$, activation, construction) configuration we select $k$ on validation accuracy and report the corresponding test accuracy; the dense activations have no $k$ and are unaffected by this selection. We cap $m \leq 30{,}000$ due to computational limitations.

Two things follow from that cap and from $k$ being an integer, and both change which datasets are available at a given expansion factor. A dataset drops out at the wide end when $m = (m/d) \cdot d$ exceeds the cap, which happens first for the highest-dimensional datasets. And it drops out at the narrow end when $m$ is too small for any integer $k$ to satisfy $k \leq 0.1m$, which happens first for the lowest-dimensional ones. The pool is therefore not the same at every $m/d$, so we compare methods only within an expansion factor and never read a trend along a row of counts. Section F gives the pool sizes at each expansion factor and which of the two causes is responsible.

**Statistics.** For each expanded dimension, we compare two methods only on the datasets where both are available, so every comparison is paired. Within a given $m/d$ we use the same set of datasets for every method being compared, so that a method is never scored on an easier subset than the one it is being compared against. We report raw metrics (accuracy $\in [0, 1]$ for classification, and $R^2$ for regression), except in fig. 3 where we use the scaled form of the metrics to aggregate performance across datasets (Ram & Sinha, 2022). Because $R^2$ is unbounded below, we verify every regression conclusion twice, once with $R^2$ floored at zero and once left raw, and we say so wherever the two disagree. In both cases $k$ is chosen on the raw validation score, so flooring changes what is reported but not which model is selected. When comparing a pair of methods we report the median difference, the inter-quartile range, the number of datasets on which the first method wins, ties or loses, and the $p$-value of a two-sided Wilcoxon signed rank test. We count a tie when the difference is at most $0.001$ (that is $0.1$ percentage points of accuracy or $0.001$ in $R^2$).

## 3 Main Results

We organise the results around the four findings. Section 3.1 compares the three ways of building $\mathbf{M}$. Section 3.2 compares the sparse representation against unrestricted ones at the same expanded dimension. Section 3.3 restricts both to binary codes. Section 3.4 fixes the sparsity level instead of tuning it. Classification and regression are reported together in each except section 3.4, which is classification only, and every comparison is paired within an expansion factor as described in section 2.

### 3.1 What is gained by learning the expansion?

Here we compare the performance of the learned expansion matrix to existing theoretical constructions, and the results are presented in fig. 3. We first compare the learned expansion to the data-independent one. The learned expansion gives higher accuracy at every expansion factor we consider, and the improvement is largest when the expansion is narrowest. The median gain over datasets is $3.0$ accuracy points at $m = d$, and it drops to $2.55$ points at $m = 4d$, $1.0$ point at $m = 16d$ and $0.13$ points at $m = 256d$. We see a significant difference at every expansion factor up to $m = 128d$ ($p \leq 0.006$, and $p < 10^{-4}$ for $m \leq 16d$), and no significant difference at $m = 256d$ ($p = 0.127$). The regression tasks behave the same way: learning the projection is worth a median $0.185$ in $R^2$ at $m = d$, that gain falls to $0.002$ at $m = 256d$, and it is significant at every expansion factor up to and including $m = 64d$ but at neither of the two widest ($p \geq 0.14$), with $R^2$ raw or floored at zero. In other words, learning the expansion gives much of what a larger expanded dimension would have given anyway, and once it is large enough there is little left for learning to add. In practice this says a learned `EaS` layer can be made small.

We next compare the learned expansion to the data-supported one. Here the gain does not decay with the expanded dimension. It is $0.90$ points at $m = d$, $1.18$ at $m = 8d$, $0.65$ at $m = 64d$ and $1.12$ at $m = 256d$, and we see a significant difference at every expansion factor ($p \leq 0.0011$), on a pool that stays between $57$ and $71$ datasets throughout. On regression the gain shrinks, from $0.13$ in $R^2$ at $m = d$ to $0.03$ at $m = 256d$, but it too is significant at every expansion factor ($p < 10^{-4}$, $R^2$ raw or floored). So against a projection drawn without reference to the data learning stops paying once the expansion is wide, and against one whose rows come from the data it keeps paying. Section G traces this difference to the data-supported $\mathbf{M}$ running out of distinct rows as $m$ grows, on regression as on classification.

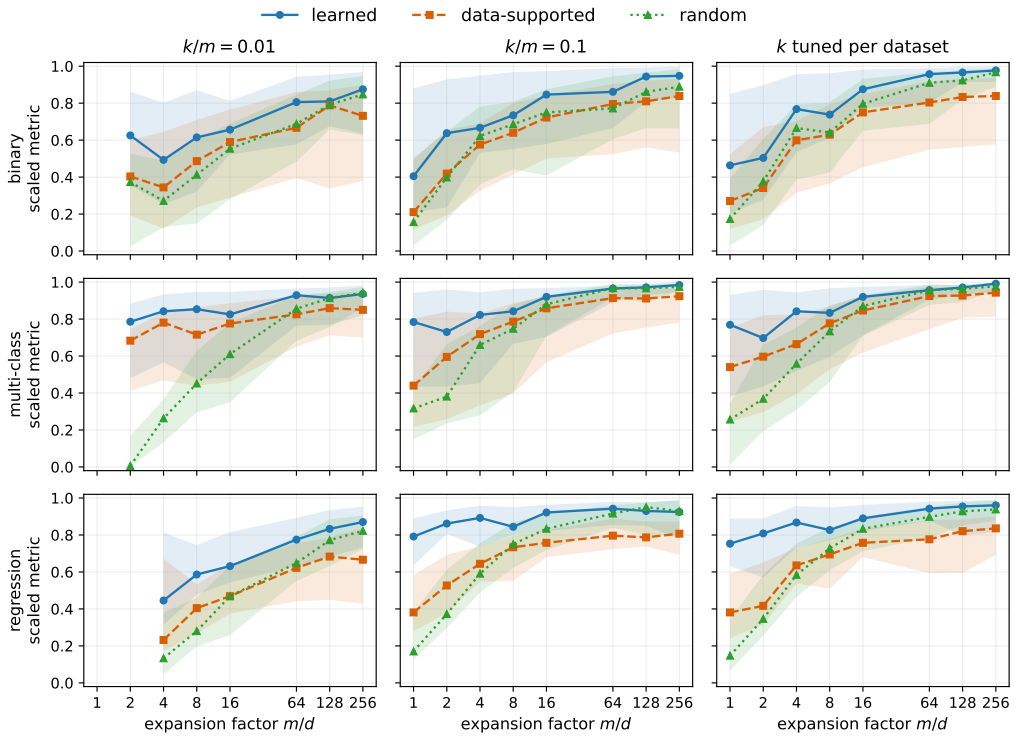



Figure 3: Scaled metrics (accuracy for classification, $R^2$ for regression; *higher is better*) of the three constructions of $\mathbf{M}$ against the expansion factor $(m/d)$, with `topk` sparsification throughout. Rows are task types (binary classification, multi-class classification and regression) and columns are sparsity levels $(k/m)$, with the rightmost column selecting $k$ on validation. Each line is the median over datasets and the ribbon is the inter-quartile range. As discussed in section 2, evaluations for $k < 1$ and $m > 30000$ are skipped so results for small $m/d$ and small $k/m$ are absent when $d$ is small. Learned `EaS` improves upon both theoretical constructions at small expansion factors in every panel. As the expansion grows the random construction catches up, while the data-supported one mostly stays below (section 3.1).

Table 1: Each activation against `ReLU` at equal expanded dimension on the 73 classification tasks, with $\mathbf{M}$ learned for both methods. Each cell gives wins/ties/losses over datasets and, in parentheses, the $p$-value of a two-sided Wilcoxon signed rank test on the per-dataset differences; a star marks $p < 0.05$. Within a column every activation is scored on the same datasets. Five of the eight expansion factors are shown here; table 2 in the section E gives all eight.

| activation | $m/d = 1$ | $m/d = 4$ | $m/d = 16$ | $m/d = 64$ | $m/d = 256$ |
|---|---|---|---|---|---|
| top-$k$ | 17/15/25 (.198) | 27/9/35 (.585) | 24/12/35 (.727) | 27/9/35 (.800) | 31/9/27 (.321) |
| LWTA | 19/14/24 (.517) | 25/9/37 (.425) | 25/11/35 (.373) | 21/16/34 (.725) | 28/13/26 (.616) |
| GELU | 25/3/29 (.899) | 23/4/44 (.023)$^*$ | 19/10/42 (.002)$^*$ | 21/8/42 (.005)$^*$ | 22/8/37 (.101) |
| SiLU | 21/6/30 (.255) | 21/4/46 (.001)$^*$ | 15/6/50 (<.001)$^*$ | 21/6/44 (.002)$^*$ | 20/11/36 (.007)$^*$ |

### 3.2 How does EaS fare against unrestricted expanded representations?

Here we compare the learned `EaS` representations to standard (dense) expanded representations, and the results are presented in fig. 4 and table 1. In fig. 4 we can see that the boxes for `topk` and `LWTA@k` sit on zero at every expansion factor while those for `GELU` and `SiLU` sit below it once $m \geq 4d$, and the table makes this precise. The `EaS` representation at $k \leq 0.1m$ is statistically indistinguishable from the unrestricted one at every expansion factor we study regardless of whether we use `topk` or `LWTA@k` for the sparsification (as we had expected): For `topk`, the median difference stays between $-0.22$ and $+0.05$ accuracy points and no Wilcoxon test reaches significance

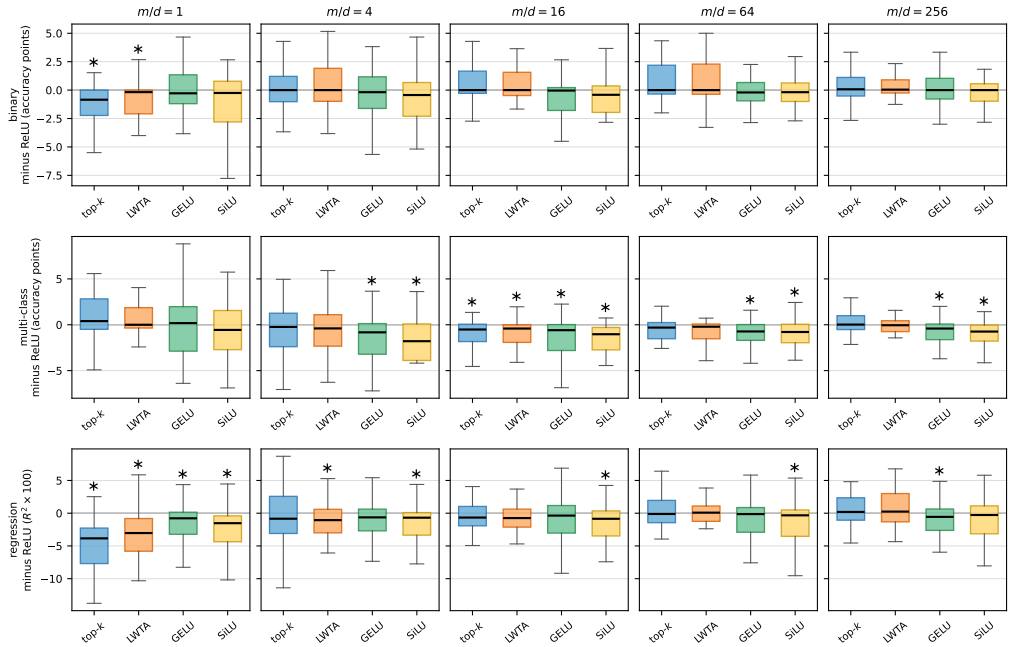



Figure 4: Per-dataset differences against `ReLU` at equal expanded dimension, with $\mathbf{M}$ learned for both methods and $k$ selected on validation for the sparse activations. Rows are task types and columns are expansion factors. Each box is the distribution over datasets of the difference for one activation, so values above zero mean the activation beats `ReLU` on that dataset. Within a panel every activation is scored on the same datasets, and a star marks a significant difference from `ReLU` in that panel (two-sided Wilcoxon, $p < 0.05$). Whiskers extend to 1.5 times the interquartile range and outliers are not drawn. All differences are multiplied by 100: accuracy differences in percentage points for classification and differences in $R^2$ for regression.

($p \geq 0.177$); for `LWTA@k`, $p \geq 0.105$. The smooth activations instead do not necessarily match `ReLU`. `GELU` falls short of `ReLU` by a median 0.16 to 0.44 points and is significantly behind at $m/d \in \{4, 8, 16, 64, 128\}$, and `SiLU` falls short by 0.25 to 0.92 points and is significantly behind at every expansion factor from $m = 4d$ upwards. Parity with `ReLU` is therefore not something every alternative activation achieves.

On the regression tasks, the `EaS` representation matches `ReLU` once the expansion is at least $4d$. With `topk` it is significantly worse only at $m = d$ and $m = 2d$. From $m = 4d$ upwards no test is significant ($p \geq 0.070$) and the median difference stays within 0.014 of zero, whether $R^2$ is left raw or floored at zero. `LWTA@k` behaves the same way from $m = 8d$ ($p \geq 0.082$). It is also significantly behind at $m = 4d$, but that result disappears when we remove the few configurations where `LWTA@k` keeps more than $k$ units (section F). As on classification, the smooth activations fall short of `ReLU`: `GELU` is significantly behind at $m/d \in \{1, 2, 8, 256\}$, and `SiLU` at every expansion factor up to $m = 64d$ (and at $m = 256d$ when $R^2$ is floored), by medians of 0.003 to 0.010 and 0.003 to 0.018 in $R^2$ respectively. Table 4 gives every value. This range overlaps the one where learning $\mathbf{M}$ still pays: at $m = 4d$ the learned $\mathbf{M}$ is ahead of a random one by a median 0.09 in $R^2$, and it stays significantly ahead through $m = 64d$. So between $m = 4d$ and $m = 64d$ the representation can be both learned and sparse at no measurable cost.

We state two important qualifications: (i) The sparsity level $k$ is selected on validation for the `EaS` representations, and the dense representations do not have any such equivalent knob. (ii) The aggregated classification results in table 1 hide a difference between the task types: `topk` is behind `ReLU` on the binary tasks at $m = d$ (0.85 points, $p = 0.0005$) and on the multi-class tasks at $m = 8d$ and $m = 16d$ (0.49 and 0.52 points, $p \leq 0.041$). These are three small significant results out of the sixteen comparisons obtained by splitting each of the eight expansion factors by task type, and fig. 4 marks those at the expansion factors it shows.

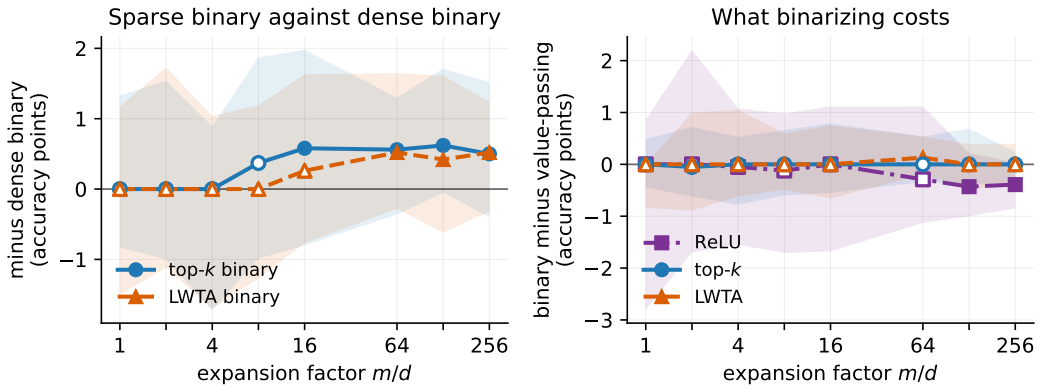



Figure 5: The effect of binarizing expanded representations on classification tasks. **[Left]** The signed difference between the accuracy of the binarized `EaS` and binarized `ReLU` representations (*higher values indicate better `EaS` performance relative to the dense versions*). **[Right]** Per-method performance degradation due to binarization (*lower values imply larger degradations*). Lines are medians over datasets and ribbons the interquartile range; filled markers are significant under a two-sided Wilcoxon signed rank test ($p < 0.05$) and hollow ones are not. Table 3 gives every value.

### 3.3 How robust is EaS to binarization of expanded representations?

We now restrict the expanded representation to be binary, so that the expanded representations post-activation $\mathbf{z} = \sigma(\mathbf{Mx}) \in \{0, 1\}^m$; we learn the model parameters using the straight-through estimator. Figure 5 shows the classification comparisons and table 3 gives every value; the regression results close this section. Binarizing `ReLU`, `GELU` or `SiLU` gives the same function, so there is a single dense binary baseline.

On classification, the binarized `EaS` representation is the better of the two once the expanded dimension is large. With `topk` sparsification, we see no significant difference for $m \leq 8d$, and then a median advantage of $0.58$ points at $m = 16d$ ($p = 0.043$), $0.56$ at $m = 64d$ ($p = 0.023$), $0.62$ at $m = 128d$ ($p = 0.0003$) and $0.50$ at $m = 256d$ ($p = 0.002$). The binarized `LWTA@k` sparsification behaves the same way, reaching a median advantage of $0.52$ points at $m = 256d$ ($p = 0.005$). So the advantage appears once $m$ is at least $16d$.

That advantage could stem from the `EaS` representations improving or the dense one degrading. We separate the two by comparing each activation against its own value-passing form, which tells us what binarization costs it. Binarizing `ReLU` costs a median $0.43$ points at $m = 128d$ ($p = 0.027$) and $0.39$ points at $m = 256d$ ($p = 0.004$), and nothing detectable at smaller expanded dimensions. Binarizing `topk` costs nothing at any expansion factor we study: the median difference is zero throughout and no test comes close to significance ($p \geq 0.229$). Binarizing `LWTA@k` is the same, with one exception at $m = 64d$ where it is a small *gain* of $0.13$ points ($p = 0.025$).

These results indicate that the advantage of `EaS` under binarization is not that the sparse representation improves, but that it loses nothing while the dense one loses about $0.4$ accuracy points. The `EaS` representation can carry information in its nonzero coordinates and thus appears to handle the binarization without any degradation. The memory and compute savings of a binary representation (section B) are therefore available from `EaS` at no measurable cost in accuracy.

In regression (with 48 datasets) binarized `topk` representation is the better than binarized `ReLU` representations once the expansion is large — by a median $0.014$, $0.028$ and $0.017$ in $R^2$ at $m = 16d$, $64d$ and $128d$ ($p \leq 0.031$, $R^2$ raw or floored), with no significant difference below $m = 16d$ or at $m = 256d$. Binarized `LWTA@k` is ahead of `ReLU` at $m = 64d$ and $128d$. The reason differs from classification. On regression, binarizing costs every activation something: `topk` loses a median $0.010$ to $0.037$ in $R^2$, significantly at every expansion factor, and `ReLU` loses $0.015$ to $0.068$, significantly everywhere except at $m = 256d$ with $R^2$ floored. The `EaS` representation keeps its advantage because it loses less, not because it loses nothing. See detailed results in table 4 in section E.

### 3.4 How sparse can the representation be?

In section 3.2 we selected $k$ on validation, so the sparsity level that ended up being used is tuned. Here we fix it instead (fig. 6, table 5), do no selection over $k$, and ask at which expanded dimensions each level is affordable. The point is not that more sparsity costs more accuracy, which is expected, but the rate at which a larger expanded dimension buys that cost back.

At 90% sparsity ($k/m = 0.10$) the `EaS` representation matches `ReLU` at every expansion factor, with a median difference of zero and no significant difference anywhere ($p \geq 0.21$). At 99% sparsity ($k/m = 0.01$) it is behind, and significantly so from $m = 4d$ upwards, but the deficit shrinks monotonically as the expanded dimension grows: 2.42 points at $m = 4d$, 2.06 at $m = 8d$, 2.00 at $m = 16d$, 0.99 at $m = 64d$, 0.61 at $m = 128d$ and 0.35 at $m = 256d$. The 97% level sits between the two. It is significantly behind `ReLU` at $m/d \in \{4, 8, 16\}$, by a median 0.59, 0.53 and 0.33 points, and we see no significant difference from $m = 64d$ onwards.

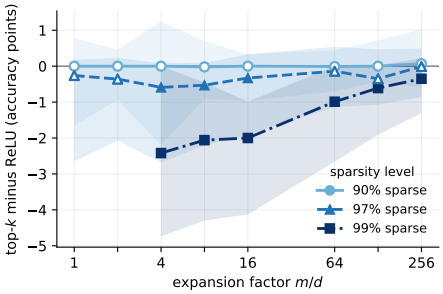



Figure 6: The signed difference between `topk` based `EaS` at a fixed sparsity $(k/m)$ and dense `ReLU` representations with varying expansion factors $(m/d)$. Lines are medians over datasets, ribbons the interquartile range; filled markers are significant ($p < 0.05$). A point is drawn only where at least 30 datasets are present: 99% sparsity leaves 8 and 23 datasets at $m = d$ and $m = 2d$. Table 5 gives every value.

We see the expected behaviour, with the cost of a sparsity level falling as the expanded dimension grows. What the numbers add is the exchange rate: the sparsity level that is free moves from 90% at $m = 4d$ to 97% at $m = 64d$. There is therefore no single sparsity level at which `EaS` begins to lose; instead, the larger the expanded dimension, the sparser the representation can be before it costs accuracy.

## 4 Concluding Remarks

In this work, we considered an elaborate evaluation with 121 datasets spanning binary/multiclass classification and regression and roughly 50,000 configurations to study the empirical behaviour of the neurobiologically inspired and theoretically motivated `EaS` representations. Our results support the practical viability of these representations when learned via standard SGD: First, we see that the learned `EaS` representations perform significantly better than the existing theoretical constructions, especially when $m \in [d, 256d]$ are of practical sizes (as opposed to the required $m \sim O(\exp(d))$ for the theoretical constructions). Then, we see that these learned `EaS` representations are quite competitive to the standard expanded representation obtained with `ReLU` networks (and similar activations), and when restricted to binary representations, `EaS` even outperforms standard representations once the expansion is large enough.

**Limitations.** The main limitation of our work is that it is an empirical study without theoretical guarantees for the learned representations, limited to single-hidden-layer networks applied to tabular datasets (albeit over 120 of them). Our choice of this network is due to its prevalence in architectures such as Transformers and MLP-Mixers. However, we do not directly consider such architectures in our evaluations as they would introduce confounders that would make it hard to isolate the effect of the `EaS` representations. One thing of note is that the explicit $k$-sparsification in `EaS` representations allows us to tune the sparsity level $(k/m)$ in a problem specific manner, whereas the existing unrestricted expanded representations do not have any such knob (see section C). Furthermore, within our elaborate evaluation, we end up with some inconsistencies. First, the minimal sparsity requirement of $k \geq 1$ and the computational limit of $m \leq 30000$ leads to situations where some configurations are not valid for datasets whose $d$ is either too large or too small. While we do not have any solution for this, we report the size of the dataset pool for each expansion in section F. Second, with the data-supported construction, when $m$ is large relative to $n$, the expansion matrix $\mathbf{M}$ ends up with repeated rows, thereby introducing redundancies and hampering the benefits of the expansion. Again, we do not have a solution for this but we report it in detail in section G.

## REPRODUCIBILITY STATEMENT

The experimental protocol is given in full in §2: the dataset admission criteria, the three constructions of $M$, the activation set, the expansion and sparsity grids, the train/validation/test splits, the selection of $k$ on validation, and the pairing and significance testing used for every comparison. §F states every case in which a method is absent from a comparison and why. All comparisons are paired on identical datasets, expanded dimensions, sparsity levels and train/test splits. The sweep and analysis code, and the raw result files from which every figure and number in this paper is computed, will be released with the paper.

## ETHICS STATEMENT

This work uses only publicly available benchmark datasets from OpenML. We collect no new data and involve no human subjects, and we do not foresee ethical concerns specific to this study.

## AI USE STATEMENT

In this work, we used generative AI tools for writing and editing portions of the text, for writing the experiment, analysis and plotting code (including the dataset preprocessing) from which every figure and table is generated, for proposing experimental designs and diagnostics, for helping interpret results, and for retrieving and summarizing prior work. We have not used generative AI tools to generate the research question or the hypotheses, which are the authors' own, and the remaining categories requiring disclosure are not applicable to this work.

We have reviewed all AI-assisted work. All code was run by the authors and its outputs verified; the analysis was cross-checked against a second, independently written script. Every citation and every attribution of a result to prior work was checked by the authors against the original paper; two claims of novelty produced this way were incorrect and were withdrawn following that check. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES


S. Boucheron, O. Bousquot, and G. Lugosi. Theory of classification : A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

Sophie J. C Caron, Vanessa Ruta, L. F Abbott, and Richard Axel. Random convergence of olfactory inputs in the drosophila mushroom body. *Nature*, 497(7447):113–117, 2013. URL https://doi.org/10.1038/nature12063.

M. Chacron, A. Longtin, and L. Maler. Efficient computation via sparse coding in electrosensory neural networks. *Current Opinion in Neurobiology*, 21(5):752–760, 2011. URL https://doi.org/10.1016/j.conb.2011.05.016.

Sanjoy Dasgupta and Christopher Tosh. Expressivity of expand-and-sparsify representations, 2020. URL https://arxiv.org/abs/2006.03741.

Sanjoy Dasgupta, Charles F. Stevens, and Saket Navlakha. A neural algorithm for a fundamental computing problem. *Science*, 358(6364):793–796, 2017. doi: 10.1126/science.aam9868. URL https://doi.org/10.1126/science.aam9868.

Sanjoy Dasgupta, Timothy C. Sheehan, Charles F. Stevens, and Saket Navlakha. A neural data structure for novelty detection. *Proceedings of the National Academy of Sciences*, 115(51): 13093–13098, 2018. URL https://doi.org/10.1073/pnas.1814448115.

Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B184E5qee.

Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. Memory-efficient transformers via top-k attention. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pp. 39–52, 2021. URL https://aclanthology.org/2021.sustainlp-1.5.

Yuchen Liang, Chaitanya Ryali, Benjamin Hoover, Leopold Grinberg, Saket Navlakha, Mohammed J Zaki, and Dmitry Krotov. Can a fruit fly learn word embeddings? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xfmSoxdxFCG.

Bruno A Olshausen and David J Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14(4):481–487, 2004. URL https://doi.org/10.1016/j.conb.2004.07.007.

Parikshit Ram and Kaushik Sinha. Federated nearest neighbor classification with a colony of fruitflies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8097–8105, 2022. URL https://doi.org/10.1609/aaai.v36i7.20775.

Chaitanya Ryali, John Hopfield, Leopold Grinberg, and Dmitry Krotov. Bio-inspired hashing for unsupervised similarity search. In *International conference on machine learning*, pp. 8295–8306. PMLR, 2020. URL https://proceedings.mlr.press/v119/ryali20a.html.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. ISBN 9781107057135.

Kaushik Sinha. Non-parametric classification via expand-and-sparsify representation. In *Advances in Neural Information Processing Systems*, pp. 29657–29696, 2024. URL https://doi.org/10.52202/079017-0933.

Kaushik Sinha and Parikshit Ram. Fruit-fly inspired neighborhood encoding for classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1470–1480, 2021. doi: 10.1145/3447548.3467246. URL https://doi.org/10.1145/3447548.3467246.

Kaushik Sinha and Christopher Tosh. Optimal rates for density and mode estimation with expand-and-sparsify representations. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2026. URL https://openreview.net/forum?id=kHuqnMOJwD.

Rupesh Kumar Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In *Advances in Neural Information Processing Systems*, 2013. URL https://dl.acm.org/doi/10.5555/2999792.2999870.

Dan D. Stettler and Richard Axel. Representations of odor in the piriform cortex. *Neuron*, 63(6): 854–864, 2009. URL https://doi.org/10.1016/j.neuron.2009.09.005.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34: 24261–24272, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/cba0a4ee5ccd02fda0fe3f9a3e7b89fe-Abstract.html.

Glenn C. Turner, Maxim Bazhenov, and Gilles Laurent. Olfactory representations by drosophila mushroom body neurons. *Journal of Neurophysiology*, 99(2):734–746, 2008. URL https://doi.org/10.1152/jn.01283.2007.

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://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, Christian Rupprecht, Daniel Cremers, Peter Vajda, and Jialiang Wang. Cache me if you can: Accelerating diffusion models through block caching. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6211–6220, 2024. URL https://doi.org/10.1109/CVPR52733.2024.00594.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–134, Feb 2004.

# A Related work

**Expand-and-sparsify representations.** An EaS representation maps $\mathbf{x} \in \mathbb{R}^d$ to a much higher-dimensional sparse code by applying a linear map $\mathbf{M} \in \mathbb{R}^{m \times d}$ with $m \gg d$ and retaining only the $k \ll m$ largest coordinates. The construction formalizes a circuit in the *Drosophila* olfactory system, in which a $d = 50$-dimensional signal is lifted to roughly $m = 2{,}000$ Kenyon cells through a connectivity pattern that appears random, after which feedback from a single inhibitory neuron suppresses all but the $5\%$ ($k/m = 0.05$) highest-firing cells (Dasgupta et al., 2017; 2018). Similar motifs are reported in the olfactory system of rodents, the visual cortex of cats and the electrosensory system of electric fish. Dasgupta & Tosh (2020) give the framework we adopt: the rows of $\mathbf{M}$ are drawn i.i.d. from a distribution $\nu$ over $\mathbb{R}^d$, and $\mathbf{Mx}$ is sparsified either by $k$-winner-take-all or by $k$-thresholding. Dasgupta & Tosh (2020, Theorem 3) establishes universal approximation under $k$-WTA with $\nu$ uniform on $\mathbb{S}^{d-1}$ (the $d$ dimensional unit sphere), provided $k \geq c_o(d \log m + \log 1/\delta)$; we restate their results, and the corresponding lower bounds on $k$ from Sinha (2024) and Sinha & Tosh (2026), in section D.

**Data-dependent expansions.** Dasgupta & Tosh (2020, §5) analyze what they call a data-dependent expansion mapping, in which the rows of $\mathbf{M}$ are drawn i.i.d. from a distribution $\nu$ whose support coincides with the support of the data distribution, so the rows of $\mathbf{M}$ take values over the same support as the inputs. Their Theorem 9 gives an approximation bound under this assumption with two properties the data-independent bounds lack: $k$ may be taken as small as desired, and there is no dependence on the ambient dimension $d$, *even in the constants*. They describe this as "mild data-dependence" and note it "leads to significantly better approximation", and their stated open problem is "more sophisticated forms of adaptivity, for instance based on a clustering of the data".

The subsequent literature nonetheless works in the data-independent setting. Sinha & Ram (2021) and Ram & Sinha (2022) take $\mathbf{M} \in \{0, 1\}^{m \times d}$ random with $s \ll d$ nonzero entries per row. Sinha (2024) draws the rows i.i.d. uniform on $\mathbb{S}^{d-1}$ or i.i.d. Gaussian $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, and closes by noting that "in both algorithms, the projection directions are chosen in a data-independent manner", with data-dependent choices left to future work. Sinha & Tosh (2026) likewise sample the rows from a fixed $\nu$.

**Sparsity level and expanded dimension, empirically.** Ram & Sinha (2022) tune over 70 OpenML datasets with $k \in [8, 256]$ and $m \in [2d, 2048d]$, and in a per-parameter study observe that for fixed $k$, increasing $m$ improves accuracy but only up to a bound, and that increasing $k$ helps "up to a point after which it can hurt performance unless $m$ is increased as well", since it reduces the sparsity of the code. Section 3.4 quantifies the same interaction against a ReLU baseline, which their study does not include.

**Competition as an activation function.** Srivastava et al. (2013) propose local winner-take-all layers, in which units are partitioned into blocks and only the maximum within each block propagates its activation. Their framework permits any block size and any activation, but in order to study competition in isolation they take the activation to be the identity and use blocks of size two throughout, which fixes the sparsity of the layer at exactly $50\%$. We are explicitly interested in the EaS setting where we first have an expansion to $m$ dimensions followed by a $k$-sparsification with $k \ll m$. Thus, our setting introduces the expansion, and the high sparsity regime, neither of which is considered in Srivastava et al. (2013). But we consider LWTA@k as a different sparsification scheme in addition to topk considered in existing literature on EaS representations for completeness.

# B What the sparsity is worth in practice

Section 3 measures accuracy. This appendix sets out what an extremely ($k \ll m$) and predictably (always exactly $k$ nonzeros) sparse expanded representation buys in compute and memory, which is the reason the accuracy question is worth asking at all.

**Loading the next layer's weights.** With $\mathbf{z} \in \mathbb{R}^m$ as a $k$-sparse EaS representation, only the $k$ nonzero coordinates of $\mathbf{z}$ contribute to $\mathbf{Wz}$, so only the $k$ columns of $\mathbf{W} \in \mathbb{R}^{l \times m}$ that multiply them need to be loaded. For a single input this is a factor $m/k$ fewer columns, which at our loosest setting is 10 and at our tightest is 1000. With a batch of $b$ inputs the union of the active coordinates is what

matters, so at most $bk$ columns are needed. That bound is attained only when no two inputs in the batch share an active coordinate. In practice inputs that are close to each other activate overlapping sets, so the number of columns actually loaded is smaller than $bk$, and how much smaller depends on how correlated the batch is. Overall, this allows for a reduction of the memory overhead from the naive $O(lm)$ to load the whole $\mathbf{W}$ to $O(lbk)$ where we just load the "batch-activated" rows of $\mathbf{W}$. Sparse linear algebra kernels can also reduce the naive computational complexity of $\mathbf{Wz}$ from $O(lm)$ to $O(lk)$ for each $\mathbf{z}$.

**Backpropagation and caching.** The same argument applies to the backward pass: gradients flow only through the $k$ nodes that were active, so the memory needed to backpropagate through the layer scales with $k$ rather than $m$ (Gupta et al., 2021). Where intermediate activations are cached rather than recomputed (Grave et al., 2017; Wimbauer et al., 2024), a representation with exactly $k$ nonzeros can be stored as $k$ index-value pairs instead of $m$ values, which is a factor $m/2k$ in the value-passing form and $m/k$ indices alone in the binary form considered in section 3.3.

**Binary codes.** A binary `EaS` representation is fixed entirely by which $k$ coordinates are active, so it can be stored as $k$ indices of $\lceil \log_2 m \rceil$ bits each and no values; at $m = 30{,}000$ that is 15 bits per active unit. The next layer then needs no multiplications, since $\mathbf{Wz}$ is the sum of the $k$ columns of $\mathbf{W}$ that $\mathbf{z}$ selects. A binarized `ReLU` layer shares the second property, but the number of units it activates varies from input to input, so its storage cannot be fixed in advance. Section 3.3 shows that on classification `EaS` gets these savings at no measurable cost in accuracy.

**Why predictability matters.** A representation that is merely sparse on average still has to be allocated for its worst case. Top-$k$ is sparse by construction, with the same count for every input, so the working set is known ahead of time and does not have to be sized for an unlucky batch. This is the property that separates `EaS` from sparsity that emerges from a `ReLU`, and it is why we hold $k$ fixed rather than letting the layer choose how many units to activate.

## C THE TUNING ASYMMETRY

We select $k$ on validation for top-$k$ and LWTA, while `ReLU`, GELU and SiLU have no comparable hyper-parameter, so part of what section 3.2 compares is the freedom to tune. Two things bound that freedom. Every setting we allow keeps at most a tenth of the coordinates, so the sparse activations are under a constraint the dense ones do not face even at their loosest. And section 3.4 repeats the comparison at fixed sparsity levels with no selection over $k$ at all, where the $90\%$ level matches `ReLU` at every expansion factor. A reader who wants the strictest reading should take that result rather than section 3.2 as the headline. Validation also tends to pick the loosest level we allow: the selected $k/m$ is the top of our grid (0.10) in about half of the classification configurations and in $59\%$ to $69\%$ of the regression ones, so the cap $k \leq 0.1m$ binds for the sparse activations rather than being a formality.

## D THEORETICAL GUARANTEES AND THE RESULTING CONSTRAINT ON $k$

This appendix collects the results referred to in section A. Nothing here is ours; we state it because the constraint it places on $k$ is what makes our sparsity grid a departure from the theory rather than an instance of it.

**Approximation under a data-independent projection.** Dasgupta & Tosh (2020) draw the rows of $\mathbf{M}$ i.i.d. from a distribution $\nu$ over $\mathbb{R}^d$ and sparsify $\mathbf{y} = \mathbf{Mx}$ either by $k$-winner-take-all or by *$k$-thresholding*, in which each coordinate has its own threshold $\tau_j$ set so that $z_j$ fires a $k/m$ fraction of the time. Their Theorem 3 establishes a universal approximation property under $k$-WTA with $\nu$ uniform on $\mathbb{S}^{d-1}$: for $\lambda$-Lipschitz $f : \mathbb{R}^d \to \mathbb{R}$, an explicitly specified linear function of $\mathbf{z} = \texttt{topk}(\mathbf{Mx})$, denoted by $\widehat{f} : \mathbb{R}^d \to \mathbb{R}$, attains $\sup_{\mathbf{x}} |\widehat{f}(\mathbf{x}) - f(\mathbf{x})| \leq c_d \lambda (4k/m)^{1/(d-1)}$ whenever $k \geq c_o(d \log m + \log 1/\delta)$. Their Theorem 4 shows that this scheme is *not* adaptive to manifold structure, since on a one-dimensional circle in $\mathbb{S}^{d-1}$ most $z_j$ are never activated, while their Theorem 8 shows that $k$-thresholding is.

**Approximation under a data-supported projection.** Theorem 9 of Dasgupta & Tosh (2020) gives an optimal $k$ of $d_o/2$ under their data-dependent sampling, an absolute quantity fixed by the intrinsic dimension $d_o$. We do not use this as a prescription: $d_o$ is not known in practice, and parameterizing by $k = s \cdot m$ has the practical advantage of controlling the sparsity of the representation directly, which is the quantity we want to vary.

**The lower bounds on $k$.** The guarantees above constrain $k$ from below. Dasgupta & Tosh (2020) require $k \geq c_o(d \log m + \log 1/\delta)$ for their cell-diameter bound; Theorem 3.12 of Sinha (2024) carries the same condition, and its empirical section accordingly sets $k = d \log m$; Lemma 5.2 of Sinha & Tosh (2026) requires $k > C_\delta^2\, d \log m$. These are lower bounds ensuring the response regions are well estimated, not prescriptions for an optimal $k$, and they sit inside asymptotic regimes far from what is reachable in practice: Corollary 3.13 of Sinha (2024) sets $m = k\, n^{(d-1)/(2\beta+d-1)}$, which is exponentially large in $d$. We therefore parameterize by $k = s \cdot m$ and report results as functions of $s$. Since our $m$ is not exponential in $d$, the extreme sparsity regime we work in, $k \leq 0.1m$, is outside the range these bounds cover, and our results should be read as evidence about that regime rather than as a test of the theory.

## E WINS, TIES AND LOSSES AT EVERY EXPANSION FACTOR

The tables below give every comparison of section 3 at every expansion factor: table 2 the classification comparisons against `ReLU` summarized in table 1, table 4 the same comparisons and the binary ones on the regression tasks, table 3 the binary codes on classification, and table 5 the fixed sparsity levels of section 3.4.

Table 2: Table 1 at every expansion factor. Each cell gives wins/ties/losses and, below it, the Wilcoxon $p$-value; a star marks $p < 0.05$.

| activation | $m/d = 1$ | $m/d = 2$ | $m/d = 4$ | $m/d = 8$ | $m/d = 16$ | $m/d = 64$ | $m/d = 128$ | $m/d = 256$ |
|---|---|---|---|---|---|---|---|---|
| top-$k$ | 17/15/25<br>(.198) | 22/13/31<br>(.686) | 27/9/35<br>(.585) | 22/12/37<br>(.178) | 24/12/35<br>(.727) | 27/9/35<br>(.800) | 30/12/28<br>(.574) | 31/9/27<br>(.321) |
| LWTA | 19/14/24<br>(.517) | 20/15/31<br>(.370) | 25/9/37<br>(.425) | 24/8/39<br>(.105) | 25/11/35<br>(.373) | 21/16/34<br>(.725) | 31/8/31<br>(.757) | 28/13/26<br>(.616) |
| GELU | 25/3/29<br>(.899) | 31/5/30<br>(.922) | 23/4/44*<br>(.023) | 22/7/42*<br>(.014) | 19/10/42*<br>(.002) | 21/8/42*<br>(.005) | 23/11/36*<br>(.044) | 22/8/37<br>(.101) |
| SiLU | 21/6/30<br>(.255) | 28/3/35<br>(.321) | 21/4/46*<br>(.001) | 20/4/47*<br>(.002) | 15/6/50*<br>(<.001) | 21/6/44*<br>(.002) | 17/8/45*<br>(.003) | 20/11/36*<br>(.007) |

Table 3: Binary codes (section 3.3). The first two rows compare each binarized sparse code against the binarized dense layer; the last three give what binarizing costs each activation, as its binary form minus its own value-passing form. Each cell gives the median difference in accuracy points, wins/ties/losses over datasets and the two-sided Wilcoxon $p$-value; a star marks $p < 0.05$. $M$ is learned on both sides.

| comparison | $m/d = 1$ | $m/d = 2$ | $m/d = 4$ | $m/d = 8$ | $m/d = 16$ | $m/d = 64$ | $m/d = 128$ | $m/d = 256$ |
|---|---|---|---|---|---|---|---|---|
| top-$k$ bin. − `ReLU` bin. | 0.00<br>26/13/18<br>(.821) | 0.00<br>26/11/29<br>(.709) | 0.00<br>30/8/33<br>(.301) | +0.37<br>38/5/28<br>(.274) | +0.58<br>40/4/27*<br>(.043) | +0.56<br>44/5/22*<br>(.023) | +0.62<br>48/5/17*<br>(<.001) | +0.50<br>43/2/22*<br>(.002) |
| LWTA bin. − `ReLU` bin. | 0.00<br>22/12/23<br>(.710) | 0.00<br>28/10/28<br>(.920) | 0.00<br>30/8/33<br>(.513) | 0.00<br>31/9/31<br>(.709) | +0.26<br>36/9/26<br>(.189) | +0.52<br>44/8/19*<br>(.011) | +0.42<br>42/5/23*<br>(.034) | +0.52<br>44/3/20*<br>(.005) |
| `ReLU` bin. − `ReLU` | 0.00<br>21/17/35<br>(.226) | 0.00<br>27/14/31<br>(.860) | −0.05<br>27/9/36<br>(.669) | −0.12<br>26/9/36<br>(.291) | 0.00<br>28/9/34<br>(.356) | −0.29<br>29/6/36<br>(.703) | −0.43<br>21/10/39*<br>(.027) | −0.39<br>16/8/43*<br>(.004) |
| top-$k$ bin. − top-$k$ | 0.00<br>20/18/19<br>(.816) | −0.05<br>20/14/32<br>(.680) | 0.00<br>26/11/34<br>(.465) | 0.00<br>28/12/31<br>(.987) | 0.00<br>30/11/30<br>(.507) | 0.00<br>33/9/29<br>(.229) | 0.00<br>30/13/27<br>(.565) | 0.00<br>25/16/26<br>(.978) |
| LWTA bin. − LWTA | 0.00<br>12/18/27<br>(.074) | 0.00<br>23/13/30<br>(1.000) | 0.00<br>30/9/32<br>(.886) | 0.00<br>26/16/29<br>(.666) | 0.00<br>30/14/27<br>(.438) | +0.13<br>36/16/19*<br>(.025) | 0.00<br>25/18/27<br>(.779) | 0.00<br>29/12/26<br>(.476) |

Table 4: The comparisons of table 2 and table 3 on the 48 regression tasks, where the metric is $R^2$, reported raw, with $k$ chosen on the raw validation score. Each cell gives the median difference, wins/ties/losses and the Wilcoxon $p$-value; a star marks $p < 0.05$. The top block scores every activation on the same datasets within each expansion factor, 32 at $m = d$, 48 at $m = 8d$ and $m = 16d$, and 46 at $m = 256d$; the bottom block pairs each comparison on its own datasets. With $R^2$ floored at zero every verdict is the same except two, both at $m = 256d$: SiLU becomes significantly behind `ReLU` and binarizing `ReLU` is no longer significant.

| comparison | $m/d = 1$ | $m/d = 2$ | $m/d = 4$ | $m/d = 8$ | $m/d = 16$ | $m/d = 64$ | $m/d = 128$ | $m/d = 256$ |
|---|---|---|---|---|---|---|---|---|
| top-$k$ | $-0.038$<br>6/0/26$^*$<br>(<.001) | $-0.030$<br>8/1/34$^*$<br>(<.001) | $-0.008$<br>16/1/29<br>(.116) | $-0.014$<br>15/1/32<br>(.070) | $-0.007$<br>15/1/32<br>(.332) | $-0.001$<br>22/1/24<br>(.747) | $-0.001$<br>21/3/23<br>(.522) | $0.002$<br>26/3/17<br>(.186) |
| LWTA | $-0.030$<br>6/0/26$^*$<br>(.001) | $-0.025$<br>8/0/35$^*$<br>(<.001) | $-0.011$<br>14/1/31$^*$<br>(.023) | $-0.011$<br>15/1/32<br>(.082) | $-0.007$<br>15/2/31<br>(.120) | $0.001$<br>23/3/21<br>(.861) | $0.001$<br>23/4/20<br>(.273) | $0.002$<br>25/1/20<br>(.251) |
| GELU | $-0.008$<br>10/1/21$^*$<br>(.025) | $-0.010$<br>15/0/28$^*$<br>(.017) | $-0.006$<br>16/4/26<br>(.068) | $-0.003$<br>16/2/30$^*$<br>(.025) | $-0.004$<br>19/2/27<br>(.155) | $-0.001$<br>20/3/24<br>(.214) | $-0.001$<br>19/4/24<br>(.210) | $-0.006$<br>15/1/30$^*$<br>(.024) |
| SiLU | $-0.015$<br>6/1/25$^*$<br>(.003) | $-0.018$<br>14/1/28$^*$<br>(.002) | $-0.007$<br>12/3/31$^*$<br>(.004) | $-0.009$<br>15/3/30$^*$<br>(.003) | $-0.009$<br>15/3/30$^*$<br>(.017) | $-0.003$<br>15/4/28$^*$<br>(.030) | $-0.001$<br>21/2/24<br>(.218) | $-0.003$<br>15/1/30<br>(.080) |
| top-$k$ bin. − `ReLU` bin. | $-0.035$<br>11/0/21<br>(.243) | $-0.007$<br>20/0/23<br>(.518) | $0.010$<br>27/1/18<br>(.132) | $0.009$<br>28/0/20<br>(.102) | $0.014$<br>30/0/18$^*$<br>(.030) | $0.028$<br>35/0/12$^*$<br>(.001) | $0.017$<br>29/3/15$^*$<br>(.009) | $0.016$<br>26/0/20<br>(.068) |
| LWTA bin. − `ReLU` bin. | $-0.039$<br>10/0/22<br>(.102) | $-0.010$<br>18/2/23<br>(.254) | $0.006$<br>24/0/22<br>(.519) | $-0.003$<br>23/1/24<br>(.397) | $0.009$<br>25/2/21<br>(.165) | $0.027$<br>30/2/15$^*$<br>(.012) | $0.009$<br>27/2/18$^*$<br>(.046) | $0.012$<br>28/0/18<br>(.082) |
| `ReLU` bin. − `ReLU` | $-0.068$<br>7/0/41$^*$<br>(<.001) | $-0.051$<br>5/0/43$^*$<br>(<.001) | $-0.045$<br>9/0/39$^*$<br>(<.001) | $-0.023$<br>11/0/37$^*$<br>(<.001) | $-0.025$<br>16/1/31$^*$<br>(.002) | $-0.024$<br>13/0/34$^*$<br>(.003) | $-0.021$<br>18/0/29$^*$<br>(.012) | $-0.015$<br>16/0/30$^*$<br>(.027) |
| top-$k$ bin. − top-$k$ | $-0.037$<br>6/0/26$^*$<br>(<.001) | $-0.037$<br>7/0/36$^*$<br>(<.001) | $-0.028$<br>6/1/39$^*$<br>(<.001) | $-0.021$<br>10/0/38$^*$<br>(<.001) | $-0.022$<br>4/2/42$^*$<br>(<.001) | $-0.011$<br>10/3/34$^*$<br>(.001) | $-0.010$<br>8/3/36$^*$<br>(<.001) | $-0.013$<br>5/2/39$^*$<br>(<.001) |
| LWTA bin. − LWTA | $-0.055$<br>4/0/28$^*$<br>(<.001) | $-0.033$<br>7/1/35$^*$<br>(<.001) | $-0.029$<br>6/2/38$^*$<br>(<.001) | $-0.025$<br>6/1/41$^*$<br>(<.001) | $-0.021$<br>9/0/39$^*$<br>(<.001) | $-0.016$<br>9/0/38$^*$<br>(<.001) | $-0.013$<br>7/0/40$^*$<br>(<.001) | $-0.011$<br>8/4/34$^*$<br>(<.001) |

Table 5: Fixed sparsity levels (section 3.4): top-$k$ at a fixed $k/m$, with no selection over $k$, minus `ReLU` at the same expanded dimension. Each cell gives the median difference in accuracy points, wins/ties/losses over datasets and the two-sided Wilcoxon $p$-value; a star marks $p < 0.05$. $M$ is learned on both sides.

| comparison | $m/d = 1$ | $m/d = 2$ | $m/d = 4$ | $m/d = 8$ | $m/d = 16$ | $m/d = 64$ | $m/d = 128$ | $m/d = 256$ |
|---|---|---|---|---|---|---|---|---|
| 90% ($k/m = 0.1$) | $0.00$<br>18/19/27<br>(.328) | $0.00$<br>22/14/33<br>(.391) | $0.00$<br>25/13/34<br>(.350) | $-0.02$<br>23/14/34<br>(.518) | $0.00$<br>21/18/32<br>(.324) | $-0.01$<br>27/12/32<br>(.592) | $0.00$<br>32/12/26<br>(.324) | $+0.07$<br>31/9/27<br>(.211) |
| 97% ($k/m = 0.03$) | $-0.26$<br>10/5/15<br>(.237) | $-0.36$<br>13/11/24<br>(.111) | $-0.59$<br>16/12/38$^*$<br>(.002) | $-0.53$<br>18/9/44$^*$<br>(.003) | $-0.33$<br>22/7/42$^*$<br>(.033) | $-0.14$<br>22/12/37<br>(.171) | $-0.35$<br>25/7/38<br>(.230) | $-0.01$<br>27/8/32<br>(.434) |
| 99% ($k/m = 0.01$) | $-3.28$<br>1/2/5<br>(.076) | $0.00$<br>6/7/10<br>(.118) | $-2.42$<br>6/7/28$^*$<br>(<.001) | $-2.06$<br>5/7/41$^*$<br>(<.001) | $-2.00$<br>5/6/54$^*$<br>(<.001) | $-0.99$<br>16/5/50$^*$<br>(<.001) | $-0.61$<br>16/9/45$^*$<br>(.008) | $-0.35$<br>18/10/39$^*$<br>(.039) |

## F DATASET COUNTS PER EXPANSION FACTOR

The number of datasets entering each comparison changes with $m/d$, for three separate reasons, and we list them here so that no count in the main text is unexplained.

First, the integer-$k$ floor. The smallest sparsity level an integer $k$ can reach is $k/m = 1/m$, so a dataset admits a sparse method at $k \leq 0.1m$ only when $m \geq 10$. At $m = d$ this excludes the eight datasets with $d < 10$, taking the pool from 73 to 65. It does not account for the whole of the first column: the observed count is 57, eight lower again. That remainder is coverage of our sweep rather

Table 6: Datasets supporting each group of methods at each expansion factor.

| | $m/d$=1 | 2 | 4 | 8 | 16 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|---|
| dense activations | 73 | 72 | 72 | 71 | 71 | 71 | 70 | 67 |
| sparse activations | 57 | 66 | 71 | 71 | 71 | 71 | 70 | 67 |
| data-supported $M$ | 57 | 66 | 68 | 71 | 71 | 71 | 70 | 67 |

than a property of the method, and it is visible on its own at $m = 2d$, where the floor binds on no dataset in our pool yet the sparse methods are supported by 66 rather than 73. From $m = 4d$ onwards the two rows agree to within one dataset.

Second, the $m \leq 30{,}000$ cap, which is what reduces both rows from 71 to 67 between $m = 64d$ and $m = 256d$. This affects every method equally and so does not bias any comparison.

Third, two gaps that are not about expansion factors at all. Three of the 76 admissible classification tasks are absent because they one-hot encode to an input dimension of roughly $10{,}000$, which made every expansion too expensive for our budget. And our two sweeps used different sparsity grids, so the $95\%$ level is defined over only part of the expansion range and we do not report it.

The regression pool behaves the same way. The paired comparison against `ReLU` rests on 32 of the 48 tasks at $m = d$, 43 at $m = 2d$, 46 at $m = 4d$ and all 48 from $m = 8d$ onwards. The integer-$k$ floor admits 41, 48, 48 and 48 at those four expansion factors, so here too it sets the shape of the narrow end without being the whole of it, and the residual is again sweep coverage.

Table 7 gives the number of datasets behind every point of fig. 3, where a dataset counts only if all three constructions of $M$ are present at that expansion factor and sparsity level. A dash marks a point supported by fewer than 10 datasets, which the figure omits.

Table 7: Datasets behind each point of fig. 3, by task type, sparsity level and expansion factor.

| task | $k/m$ | $m/d$=1 | 2 | 4 | 8 | 16 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|---|---|
| binary | 0.01 | – | 12 | 26 | 28 | 37 | 37 | 37 | 35 |
| | 0.1 | 27 | 34 | 34 | 35 | 37 | 37 | 37 | 35 |
| | tuned | 30 | 34 | 35 | 37 | 37 | 37 | 37 | 35 |
| multi-class | 0.01 | – | 14 | 19 | 28 | 34 | 34 | 33 | 32 |
| | 0.1 | 22 | 30 | 31 | 33 | 34 | 34 | 33 | 32 |
| | tuned | 27 | 32 | 33 | 34 | 34 | 34 | 33 | 32 |
| regression | 0.01 | – | – | 16 | 27 | 33 | 32 | 32 | 31 |
| | 0.1 | 17 | 19 | 31 | 32 | 33 | 32 | 32 | 31 |
| | tuned | 19 | 29 | 32 | 33 | 33 | 32 | 32 | 31 |

**How many units `LWTA@k` keeps.** `LWTA@k` keeps one unit per block of size $\mathrm{round}(m/k)$, which is $\lceil m/\mathrm{round}(m/k) \rceil$ units: exactly $k$ when $k$ divides $m$, and otherwise usually $k$ or $k + 1$. Among the configurations selected on validation, $63\%$ keep exactly $k$ units, $33\%$ keep $k + 1$, $4\%$ keep between $k + 2$ and $k + 8$, and two keep $k - 1$. The realised sparsity then exceeds $k/m = 0.105$ in about $4\%$ of the classification configurations and $7\%$ of the regression ones, all at $m \leq 16d$. Removing them changes no classification conclusion, and on regression it removes only the significance of `LWTA@k`'s deficit at $m = 4d$.

## G Distinct rows of the data-supported expansion

The difference between the two comparisons in section 3.1 has a cause we can measure. The rows of a data-supported $M$ are training points, so it can never hold more than $n$ distinct directions however large $m$ becomes, while the other two constructions hold $m$. The fraction of $M$ that is distinct falls from $97\%$ at $m = d$ to $76\%$ at $m = 16d$ and $12\%$ at $m = 256d$ (table 8). Compared against a data-independent $M$, the sampled one wins at $m = d$, $2d$ and $4d$, by a median 1.64, 1.55 and 0.91 points, all significant; is indistinguishable at $m = 8d$ and $m = 16d$; and *loses* from

Table 8: How many of the $m$ rows of a data-supported $M$ are distinct. The rows are drawn from the training inputs with replacement, so the count is bounded by $n$ and the fraction falls as $m$ grows. $D$ is the number of (dataset, sparsity level) configurations entering the column. Each row is a separate median over those configurations, so the last row is the median of the per-configuration fraction and not the quotient of the two rows above it.

| | $m/d$=1 | 2 | 4 | 8 | 16 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|---|
| configurations $D$ | 120 | 166 | 192 | 207 | 237 | 275 | 285 | 276 |
| median $m$ | 58 | 106 | 184 | 296 | 592 | 2304 | 4096 | 6912 |
| median distinct rows | 54 | 102 | 158 | 244 | 326 | 622 | 665 | 693 |
| median fraction distinct | 96.9% | 94.7% | 89.9% | 85.2% | 75.6% | 38.5% | 21.6% | 12.0% |

$m = 64d$, by $0.41$, $0.38$ and $0.77$ points, again all significant. The sign changes in exactly the range where the fraction of distinct rows falls through three quarters, which is what we would expect if its advantage comes from the directions being real data and its disadvantage from there not being enough of them. Regression shows the same crossover at the same expansion factors: the sampled $\mathbf{M}$ is ahead of a data-independent one at $m \leq 4d$, by a median $0.044$, $0.024$ and $0.013$ in $R^2$ ($p \leq 0.025$), indistinguishable at $m = 8d$ and $m = 16d$, and behind from $m = 64d$, by $0.019$ to $0.024$ ($p < 0.0002$), with $R^2$ raw or floored.

The saturation is a property of sampling with replacement rather than of our data: once $m$ exceeds the training set size, no further distinct rows are available to draw. Its consequence is the crossover above: identical rows give identical pre-activations, so a top-$k$ over a saturated $M$ is choosing among ties, and the expansion has far fewer distinct directions than its nominal size.

## H Approximation and generalization results

In this section we bound the excess Bayes risk when $M \in \mathbb{R}^{m \times d}$ is random and $\mathbf{w} \in \mathbb{R}^m$ is learned via SGD using binary cross entropy (`bce`) loss in the binary classification setting. SGD learns a function from the following function class:

$$\mathcal{F} = \{f(\mathbf{x}; \mathbf{w}) = W \cdot \sigma(\mathbf{M}\mathbf{x}) : \mathbf{w} \in \mathbb{R}^m, \|\mathbf{w}\|_2 \leq R\}$$

where $\sigma$ is the $k$ winners-take-all activation such that for any $\mathbf{z} \in \mathbb{R}^m$, $\sigma(\mathbf{z})$ sets the largest $k$ entries of $\mathbf{z}$ to 1 and the rest to zero.

We first show that the `bce` loss and `logistic` loss are mathematically equivalent as this will help us work with `logistic` loss instead of `bce` loss. For any $f \in \mathcal{F}$, let $z = f(\mathbf{x}; \mathbf{w})$. Then the `bce` loss is defined as:

$$\texttt{bce}(y, z) = -y \ln(s(z)) - (1 - y) \ln(1 - s(z)) \tag{2}$$

where $s(z) = 1/(1 + e^{-z})$ is the sigmoid function. Note that $y \in \{0, 1\}$. Next, the `logistic` loss function is defined as:

$$\texttt{logistic}(y', z) = \log_2(1 + e^{-y'z}) = \log_2 e \cdot \ln(1 + e^{-y'z}) \tag{3}$$

Note that here $y' \in \{-1, 1\}$. It is easy to use the transformation $y' = 2y - 1$ and write the `logistic` loss in terms of $y$ as follows:

$$\texttt{logistic}(y, z) = \log_2(1 + e^{-(2y-1)z}) = \log_2 e \cdot \ln(1 + e^{-(2y-1)z}) \tag{4}$$

**Lemma H.1.** *The `bce` loss given in equation 2 and the `logistic` loss given in equation 4 are equivalent.*

*Proof.* For $y = 1$,

$$\texttt{bce}(1, z) \quad = \quad -\ln s(z) = \ln(1 + e^{-z}) = \left(\frac{1}{\log_2 e}\right) \cdot \texttt{logistic}(1, z).$$

Similarly, for $y = 0$,

$$\begin{aligned}\texttt{bce}(0, z) &= -\ln(1 - s(z)) = -\ln\left(\frac{e^{-z}}{1 + e^{-z}}\right) = \ln(1 + e^{-z}) + z \\ &= \ln\left(\frac{1 + e^{z}}{e^{z}}\right) + z = \ln(1 + e^{z}) = \left(\frac{1}{\log_2 e}\right) \cdot \texttt{logistic}(0, z).\end{aligned}$$

□

From now on, we will assume that the labels come from the set $\{1, -1\}$ and because of the equivalency above, use the `logistic` loss given in equation 3.

To formalize the learning problem, we introduce a probabilistic setting, and let $(\mathbf{X}, Y)$ be an $\mathcal{X} \times \{1, -1\}$ - valued random pair, modeling observation and its corresponding class. The distribution of the random pair $(\mathbf{X}, Y)$ may be described by the probability distribution of $\mathbf{X}$ (given by the probabilities $\Pr(\mathbf{X} \in A) = \mu(A)$ for all measurable subsets $A \subset \mathcal{X}$) and conditional probability, $\eta(\mathbf{x}) = \Pr(Y = 1 | \mathbf{X} = x)$. For any classifier $g : \mathcal{X} \to \{1, -1\}$, its probability of error (also known as risk) is given by

$$L(g) = \Pr(g(\mathbf{X}) \neq Y) \tag{5}$$

and given a training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ sampled i.i.d. from $\mathcal{X} \times \{1, -1\}$, its empirical version is given by:

$$L_n(g) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{\{g(\mathbf{x}_i) \neq y_i\}} \tag{6}$$

Given $\eta$, one may easily construct a classifier with minimal probability of error, known as the Bayes optimal classifier as follows:

$$g^*(\mathbf{x}) = \begin{cases} 1, & \text{if } \eta(\mathbf{x}) \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{7}$$

The minimal risk $L^* \triangleq L(g^*)$. Given a training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ sampled i.i.d. from $\mathcal{X} \times \{1, -1\}$, we learn a function $f_n \in \mathcal{F}$ by minimizing empirical `logistic` loss using SGD and we aim to bound $L(f_n) - L^*$.

**Theorem H.2.** *Let $f_n \in \mathcal{F}$ be the solution for the setting where $M$ is random and $W$ is obtained by minimizing empirical `logistic` loss using SGD. For the `EaS` representation, let $k \leq 0.1m$. Fix any $\delta \in (0, 1)$. Then with probability at least $1 - 2\delta$,*

$$L(f_n) - L^* \leq \underbrace{O\left(\left(\frac{mk}{n}\right)^{1/4}\right)}_{\textit{estimation error}} + \underbrace{O\left(\left(\frac{d \log m}{m}\right)^{1/2(d-1)}\right)}_{\textit{approximation error}}$$

*Proof.* Consider the logit cost function $\phi(x) = \log_2(1 + e^x)$, which is a non-negative non-decreasing convex function satisfying $\lim_{x \to -\infty} \phi(x) = 0$ and $\phi(0) = 1$. For any function $f : \mathcal{X} \to \{1, -1\}$, note that cost functional $A(f) = \mathbb{E}\phi(-Yf(\mathbf{X}))$ is the expected logistic loss and its empirical version is $A_n(f) = \frac{1}{n} \sum_{i=1}^{n} \phi(-y_i f(\mathbf{x}_i))$. It is easy to check that $L(f) \leq A(f)$ and $L_n(f) \leq A_n(f)$. Note that for any $x \in \mathcal{X}$, we have

$$\mathbb{E}[\phi(-Yf(\mathbf{X})) | \mathbf{X} = \mathbf{x}] = \eta(\mathbf{x})\phi(-f(\mathbf{x})) + (1 - \eta(\mathbf{x}))\phi(f(\mathbf{x}))$$

For each $\eta(\mathbf{x}) \in [0, 1]$, define $h_{\eta(\mathbf{x})}(\alpha) = \eta(\mathbf{x})\phi(-\alpha) + (1 - \eta(\mathbf{x}))\phi(\alpha)$. Therefore, the function $f^*$ that minimizes the cost functional $A(f)$ is given by

$$f^*(\mathbf{x}) = \arg\min_{\alpha} h_{\eta(\mathbf{x})}(\alpha)$$

Since $h_\eta$ is strictly convex and differentiable, $f^*$ is well defined and the minimum is achieved for the value of $\alpha$ for which $h'_\eta(\alpha) = 0$, that is when,

$$(1 - \eta)\phi'(\alpha) - \eta\phi'(-\alpha) = 0$$

Since $\phi'(\alpha) = e^{\alpha}/(1+e^{\alpha})$ and $\phi'(-\alpha) = e^{-\alpha}/(1+e^{-\alpha}) = 1/(1+e^{\alpha})$, plugging in these values and simplifying yields $\alpha = \ln\left(\frac{\eta}{1-\eta}\right)$, that is $f^*(\mathbf{x}) = \ln\left(\frac{\eta(\mathbf{x})}{1-\eta(\mathbf{x})}\right)$. Note that $f^*(\mathbf{x})$ is positive if and only if $\eta(\mathbf{x}) > 1/2$. Therefore, the function $g^*(\mathbf{x}) = 2\mathbb{I}_{\{f^*(\mathbf{x})>0\}} - 1$ is the Bayes optimal classifier.

Let $A^* = A(f^*) = \inf_f A(f)$. Now we want to bound the excess probability of error $L(f) - L^*$ in terms of $A(f) - A^*$. Towards this end we use an inequality from (Zhang, 2004) which says that if the function $H : [0,1] \to \mathbb{R}$ defined by $H(\eta) = \inf_\alpha h_\eta(\alpha)$ and cost function $\phi$ is such that for some positive constants $s \geq 1$ and $c \geq 0$ it satisfies

$$(1/2 - \eta)^s \leq c^s(1 - H(\eta)), \eta \in [0,1] \tag{8}$$

then, for any function $f : \mathcal{X} \to \mathbb{R}$ it holds that

$$L(f) - L^* \leq 2c\,(A(f) - A^*)^{1/s}$$

In our case, since $\alpha = \ln\left(\frac{\eta}{1-\eta}\right)$ minimizes $h_\eta(\alpha)$, plugging in this value and simplifying yields $H(n) = 2\eta(1-\eta)$. Therefore, equation 8 holds with $s = 1$ and $c = 1/\sqrt{2}$.

Let $f_n \in \mathcal{F}$ be the function obtained by minimizing empirical `logistic` loss using SGD. Let $f^\dagger \in \mathcal{F}$ whose weight vector $\mathbf{w} = (w_1, \ldots, w_m)$ is specified by $w_j = f^*(C_j) = (1/\mu(C_j)) \int_{C_j} f^* d\mu$, where $C_j = \{\mathbf{x} \in \mathcal{X} : z_j = 1, \mathbf{z} = \sigma(Mx)\}$ (see (Dasgupta & Tosh, 2020) for details). Then we have,

$$\begin{aligned}
L(f_n) - L^* &\leq \sqrt{2(A(f_n) - A^*)} \\
&= \sqrt{2(A(f_n) - \inf_{f\in\mathcal{F}} A(f) + \inf_{f\in\mathcal{F}} A(f) - A(f^\dagger) + A(f^\dagger) - A^*)} \\
&\leq \sqrt{2|A(f_n) - \inf_{f\in\mathcal{F}} A(f)|} + \sqrt{2|A(f^\dagger) - \inf_{f\in\mathcal{F}} A(f)|} + \sqrt{2|A(f^\dagger) - A*|} \\
&\overset{a}{\leq} \sqrt{4\sup_{f\in\mathcal{F}} |A(f) - A_n(f)|} + \sqrt{4\sup_{f\in\mathcal{F}} |A(f) - A_n(f)|} + \sqrt{2|A(f^\dagger) - A^*|} \\
&= 4\sqrt{\sup_{f\in\mathcal{F}} |A(f) - A_n(f)|} + \sqrt{2|A(f^\dagger) - A^*|} \\
&\overset{b}{\leq} 4\sqrt{c\sqrt{\frac{mk}{n}} + B\sqrt{\frac{2\log(2/\delta)}{n}}} + \sqrt{2|A(f^\dagger) - A^*|} \\
&\overset{c}{\leq} 4\sqrt{c\sqrt{\frac{mk}{n}} + B\sqrt{\frac{2\log(2/\delta)}{n}}} + \sqrt{2c_d\lambda\left(\frac{2k}{m} + \frac{2c_0(d\log m + \log(1/\delta))}{m}\right)^{1/(d-1)}} \\
&\overset{d}{\leq} 4\sqrt{c\sqrt{\frac{mk}{n}} + B\sqrt{\frac{2\log(2/\delta)}{n}}} + \sqrt{2c_d\lambda\left(\frac{2c_0(d\log m + \log(1/\delta))}{m}\right)^{1/(d-1)}}
\end{aligned}$$

where, inequality $a$ follows using elementary inequality (for example, see (2) of (Boucheron et al., 2005)) and inequality $b$ follows from Lemma H.3. Inequality $c$ is due to Lemma H.4, and inequality $d$ is due to our choice of $k \leq 0.1m$. □

**Lemma H.3.** *(Estimation error) Let $B$ denote a uniform upper bound on $\phi(-f(x)y)$ and let $L_\phi = 1$ be the Lipschitz constant of the logit cost function $\phi(x) = \log_2(1 + e^x)$. Fix any $\delta \in (0,1)$. Then there exists a positive constant $c > 0$ such that with probability at least $1 - \delta$, the following holds:*

$$\sup_{f\in\mathcal{F}} |A(f) - A_n(f)| \leq c\sqrt{\frac{mk}{n}} + B\sqrt{\frac{2\log(2/\delta)}{n}}.$$

*Proof.* Let $\mathcal{H}$ be the class of functions $\mathcal{X} \times \{1, -1| \to \mathbb{R}$ of the form $-f(\mathbf{x})y, f \in \mathcal{F}$. For the training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, let $\mathbf{z}_1^n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n))$ and $\mathbf{x}_1^n = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Let

$\mathcal{F}(\mathbf{x}_1^n) = \{((f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)) : f \in \mathcal{F}\}$ and $\mathcal{H}(\mathbf{z}_1^n) = \{((-f(\mathbf{x}_1)y_1, \ldots, -f(\mathbf{x}_n)y_n) : f \in \mathcal{F}\}$, where $\mathbf{z}_1, \ldots, \mathbf{z}_n \triangleq (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$. Using Theorem 3.2 of (Boucheron et al., 2005), with probability at least $1 - \delta$,

$$\begin{aligned}
\sup_{f \in \mathcal{F}} |A(f) - A_n(f)| &\leq 2\mathcal{R}(\phi \circ \mathcal{H}(\mathbf{z}_1^n)) + B\sqrt{\frac{2\log(2/\delta)}{n}} \\
&\overset{a}{\leq} 2L_\phi \mathcal{R}(\mathcal{H}(\mathbf{z}_1^n)) + B\sqrt{\frac{2\log(2/\delta)}{n}} \\
&= 2\mathcal{R}(\mathcal{H}(\mathbf{z}_1^n)) + B\sqrt{\frac{2\log(2/\delta)}{n}} \\
&\overset{b}{=} 2\mathcal{R}(\mathcal{F}(\mathbf{x}_1^n)) + B\sqrt{\frac{2\log(2/\delta)}{n}} \\
&\overset{c}{\leq} \frac{2R\max_i \|\sigma(\mathbf{M}\mathbf{x}_i)\|_2}{\sqrt{n}} + B\sqrt{\frac{2\log(2/\delta)}{n}} \\
&\overset{d}{\leq} c\sqrt{\frac{mk}{n}} + B\sqrt{\frac{2\log(2/\delta)}{n}}
\end{aligned}$$

where, inequality $a$ is due to contraction principle (Theorem 3.3 of (Boucheron et al., 2005), equality $b$ is due to the fact that Radamacher complexity $\mathcal{R}(\mathcal{H}(\mathbf{z}_1^n)) = \mathcal{R}(\mathcal{F}(\mathbf{z}_1^n))$ (again, due to Theorem 3.3 of (Boucheron et al., 2005), and inequality $c$ is due to Theorem 26.10 of (Shalev-Shwartz & Ben-David, 2014). Finally, inequality $d$ follows from the fact that $w$ beining a vector in $\mathbb{R}^m$, the quantity $R$ which is an upper bound on $\|W\|_2$ must be $O(\sqrt{m})$. Also, for any $\mathbf{x} \in \mathbb{R}^d$, since $\sigma(\mathbf{M}\mathbf{x})$ is a $k$-sparse vector the quantity $\max_i \|\sigma(\mathbf{M}\mathbf{x}_i\|_2 = O(\sqrt{k})$. □

**Lemma H.4.** *(Approximation error) Let $f^\dagger \in \mathcal{F}$ and $f^* \in \mathcal{F}$ be as defined above. Assume $f^*$ to be $\lambda$ Lipschitz. Pick any $\delta \in (0, 1)$. Then with probability at least $1 - \delta$, over the choice of $M$, we have,*

$$|A(f^\dagger) - A^*| \leq c_d \lambda \left(\frac{2k}{m} + \frac{2c_0(d\log m + \log(1/\delta))}{m}\right)^{1/(d-1)}$$

*where $c_d$ is a constant that depends on $d$ and $c_0$ is a positive constant.*

*Proof.*

$$\begin{aligned}
|A(f^\dagger) - A^*| &= |A(f^\dagger) - A(f^*)| \\
&= \left|\mathbb{E}\left(\log_2(1 + e^{-yf^\dagger(\mathbf{x})}\right) - \mathbb{E}\left(\log_2\left(1 + e^{-yf^*(\mathbf{x})}\right)\right)\right| \\
&= \left|\mathbb{E}\left(\log_2(1 + e^{-yf^\dagger(\mathbf{x})} - \log_2\left(1 + e^{-yf^*(\mathbf{x})}\right)\right)\right| \\
&\overset{a}{\leq} \mathbb{E}\left|\left(\log_2(1 + e^{-yf^\dagger(\mathbf{x})} - \log_2\left(1 + e^{-yf^*(\mathbf{x})}\right)\right)\right| \\
&\overset{b}{\leq} \mathbb{E}\left|yf^\dagger(\mathbf{x}) - yf^*(\mathbf{x})\right| = \mathbb{E}\left|f^\dagger(\mathbf{x}) - f^*(\mathbf{x})\right| \\
&\overset{c}{\leq} c_d\lambda\left(\frac{2k}{m} + \frac{2c_0(d\log m + \log(1/\delta))}{m}\right)^{1/(d-1)}
\end{aligned}$$

where, inequality $a$ is due to Jensen's inequality, inequality $b$ follows from the fact that the function $f(t) = \log_2(1 + e^{-t})$ is 1-Lipschitz. Inequality $c$ holds with probability at least $1 - \delta$ and follows from Theorem 3 of (Dasgupta & Tosh, 2020). □