www.sciencemag.org/content/358/6364/793/suppl/DC1

Supplementary Materials for

A neural algorithm for a fundamental computing problem

Sanjoy Dasgupta, Charles F. Stevens, Saket Navlakha*

*Corresponding author. Email: navlakha@salk.edu

Published 10 November 2017, *Science* **358**, 793 (2017)
DOI: 10.1126/science.aam9868

This PDF file includes:

Materials and Methods

Datasets and pre-processing. Our empirical evaluations were performed on four benchmark datasets: SIFT (*31*) ($d = 128$), GLOVE (*32*) ($d = 300$), MNIST (*33*) ($d = 784$), and GIST (*31*) ($d = 960$). For each dataset, we selected a subset of size 10,000 inputs to efficiently perform the all-vs-all comparison in determining true nearest neighbors. For all datasets, each input vector was normalized to have the same mean.

Fixing the computational complexity for LSH and the fly to be the same. To perform a fair comparison between the fly's approach and LSH, we fixed the computational complexity of both algorithms to be the same (Fig. 1C). That is, the two approaches were fixed to use the same number of mathematical operations to generate a hash with length k (i.e., a vector with k non-zero values) for each input. LSH computes $m = k$ random projections per input, but each projection requires $2d$ operations — multiplying each entry of the d-dimensional input by an i.i.d. Gaussian random value, and then doing d summations. For the fly, each binary random projection only requires $0.1d$ operations to compute — summing the roughly 10% of the input indices sampled (6 out of 50) by each Kenyon cell. Thus, the fly can compute $m = 20k$ random projections, while incurring the same computational expense as LSH. The only additional expense for the fly is the sparsification step so that only k (of the $20k$) values are non-zero, as in the LSH tag.

Supplementary Text

Formal definition of a locality-sensitive hash function

The formal definition of a locality-sensitive hash function is as follows:

Definition 1. *A hash function $h : \mathbb{R}^d \rightarrow \mathbb{R}^m$ is called locality-sensitive if for any two points $p, q \in \mathbb{R}^d$, $\Pr[h(p) = h(q)] = sim(p, q)$, where $sim(p, q) \in [0, 1]$ is a similarity function defined on two input points.*

In practical applications for nearest-neighbors search, a second (traditional) hash function is used to place each m-dimensional point into a discrete bin so that all similar images lie in the same bin, for easy retrieval.

In this paper, we focus only on designs for the LSH function (h) and study how tags are generated and the computational properties of the tag. How the tag is subsequently used is the next natural question. Computationally, the binning step (placing each m-dimensional point into a discrete bin) is important because processing a query image then involves simply finding its bin and returning the most similar images that lie in the same bin, which takes sub-linear time. Biologically, the tag is used in the mushroom body for learning, which occurs by identifying which Kenyon cells respond to an odor (the tag), and modifying the strength of their synapses onto approach and avoidance circuits. How learning occurs algorithmically using this tag remains an open problem. Even if learning does not require a similar "binning" step, both problems require the same first step — forming the tag/hash of an input point — which is our focus here.

Theoretical analysis of the fly olfactory circuit

The mapping from projection neurons (PNs) to Kenyon cells (KCs) can be viewed as a bipartite connection matrix, with $d = 50$ PNs on the left and the $m = 2000$ KCs on the right. The nodes on the left take values $x_1, \ldots, x_d$ and those on the right are $y_1, \ldots, y_m$. Each value y_j is equal to the sum of a small number of the x_i's; we represent this relationship by an undirected edge connecting every such x_i with y_j. This bipartite graph can be summarized by an $m \times d$ adjacency matrix M:

$$M_{ji} = \begin{cases} 1 & \text{if } x_i \text{ connects to } y_j \\ 0 & \text{otherwise.} \end{cases}$$

Moving to vector notation, with $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and $y = (y_1, \ldots, y_m) \in \mathbb{R}^m$, we have[1]:

$$y = Mx.$$

After feedback inhibition from the APL neuron, only the k highest firing KCs retain their values; the rest are zeroed out. This winner-take-all mechanism produces a sparse vector $z \in \mathbb{R}^m$ (called the tag) with:

$$z_i = \begin{cases} y_i & \text{if } y_i \text{ is one of the } k \text{ largest entries in } y \\ 0 & \text{otherwise.} \end{cases}$$

A simple model of M is a sparse, binary random matrix: each entry M_{ij} is set independently with probability p. Choosing $p = 6/d$, for instance, would mean that each row of M has roughly 6 entries equal to 1 (and all of the other entries are 0), which matches experimental findings.

Below, we prove that the first two steps of the fly's circuitry produces tags that preserve ℓ_2 distances of input odors in expectation. The third step (winner-take-all) is then a simple method for sparsifying the representation while preserving the largest and most discriminative

[1]In practice, an additional quantization step is used for discretization: $y = \lfloor \frac{Mx}{w} \rfloor$, where w is a constant, and $\lfloor \cdot \rfloor$ is the floor operation.

coefficients (*34*). We also prove that when m is large enough (i.e., the number of random projections is $O(d)$), the variance $\|y\|^2$ is tightly concentrated around its expected value.

Distance-preserving properties of sparse binary projections

Here we establish that sparse binary random projections, of the type outlined above, preserve neighborhood structure if the number of projections m is sufficiently large. A key determiner of how well distances are preserved is the *sparsity* of the vectors x.

Fix any $x \in \mathbb{R}^d$ denoting the activations of the projection neurons. Let M_j denote the jth row of matrix M, so that $Y_j = M_j \cdot x$ is the value of the jth Kenyon cell. Let's first compute the first and second moments of Y_j.

Lemma 1. *Fix any $x \in \mathbb{R}^d$ and define $Y = (Y_1, \ldots, Y_m) = Mx$. For any $1 \leq j \leq m$,*

$$\begin{aligned} \mathbb{E}Y_j &= p(\mathbf{1} \cdot x) \\ \mathbb{E}Y_j^2 &= p(1-p)\|x\|^2 + p^2(\mathbf{1} \cdot x)^2 \end{aligned}$$

where $\mathbf{1}$ is the all-ones vector (and thus $\mathbf{1} \cdot x$ is the sum of the entries of x). For the squared Euclidean norm of Y, namely $\|Y\|^2 = Y_1^2 + \cdots + Y_m^2$, this implies

$$\mathbb{E}\|Y\|^2 = mp\left((1-p)\|x\|^2 + p(\mathbf{1} \cdot x)^2\right).$$

Likewise, if two inputs $x, x' \in \mathbb{R}^d$ get projected to $Y, Y' \in \mathbb{R}^m$, respectively, we have

$$\mathbb{E}\|Y - Y'\|^2 \;=\; mp\left((1-p)\|x - x'\|^2 + p(\mathbf{1} \cdot (x - x'))^2\right) \tag{1}$$

In the fly, the second (bias) term, $p^2(\mathbf{1} \cdot (x - x'))^2 \approx 0$, because x and x' have roughly the same total activation level. This is because all odors are represented as an exponential distribution of firing rates with the same mean, for all odors and all odor concentrations. Thus, the bias term is negligible, and the random projection $x \mapsto Y$ preserves ℓ_2 distances.

The result (1) is only a statement about *expected* distances. The reality could be very different if the variance of $\|Y\|^2$ is high. However, we will see that when m is large enough, $\|Y\|^2$ is tightly concentrated around its expected value, in the sense that

$$(1-\epsilon)\mathbb{E}\|Y\|^2 \ \leq\ \|Y\|^2 \ \leq\ (1+\epsilon)\mathbb{E}\|Y\|^2,$$

with high probability, for small $\epsilon > 0$. The required m depends on how sparse x is.

It is useful to look at two extremal cases in more detail:

1. x is *very sparse.*

   Let's say the only non-zero coordinate of x is x_1. Then $Y_j = M_j \cdot x$ has the following distribution:
   $$Y_j = \begin{cases} x_1 & \text{with probability } p \\ 0 & \text{otherwise} \end{cases}$$
   This is usually zero, and if not, then $\|x\|$.

2. x is *uniformly spread.*

   If $x = (x_o, x_o, \ldots, x_o)$, then Y_j has mean $pdx_o = c\|x\|/\sqrt{d}$. The distribution of Y_j/x_o is roughly Poisson.

Thus individual Y_j can have a fairly large spread of possible values if x is sparse. How large must m be for $\|Y\|^2$ to be tightly concentrated around its expected value? It turns out that it is always sufficient to take $m = O(d)$, and that this upper bound is also necessary for sparse x. For x closer to uniform, $m = O(1)$ is sufficient.

Lemma 2. *Fix any $x \in \mathbb{R}^d$ and pick $0 < \delta, \epsilon < 1$. If we take*

$$m \geq \frac{5}{\epsilon^2\delta}\left(2c + \frac{d\|x\|_4^4}{\|x\|^4}\right),$$

then with probability at least $1-\delta$*, we have* $(1-\epsilon)\mathbb{E}\|Y\|^2 \leq \|Y\|^2 \leq (1+\epsilon)\mathbb{E}\|Y\|^2$.

Here $\|x\|_4$ is the 4-norm of x, so

$$\|x\|_4^4 = \sum_{i=1}^{d} x_i^4.$$

The ratio $\|x\|_4^4/\|x\|^4$ lies in the range $[1/d, 1]$. It is 1 when x is very sparse and $1/d$ when x is uniformly spread out. We now show that this ratio is roughly $6/d$ when the individual coordinates of x are independent draws from the same exponential distribution.

Lemma 3. *Suppose* $X = (X_1, \ldots, X_d)$*, where the* X_i *are i.i.d. draws from an exponential distribution (with any mean parameter).*

(a)
$$\frac{\mathbb{E}\|X\|_4^4}{(\mathbb{E}\|X\|_2^2)^2} = \frac{6}{d}.$$

(b) Moreover, $\|X\|_2^2$ *and* $\|X\|_4^4$ *are tightly concentrated around their expectations. In particular, for any positive integer c, and any* $0 < \delta < 1$*, we have that with probability at least* $1-\delta$*,*

$$\|X\|_c^c = \mathbb{E}(\|X\|_c^c)\left(1 \pm \frac{2^c}{\sqrt{d\delta}}\right).$$

Proof of Lemma 1

Fix any $x \in \mathbb{R}^d$ and $1 \leq j \leq m$. For any $i \neq i'$ we have

$$\mathbb{E}M_{ji} = p$$
$$\mathbb{E}(M_{ji}M_{ji'}) = p^2$$

The expressions for $\mathbb{E}Y_j$ and $\mathbb{E}Y_j^2$ then follow immediately, using linearity of expectation:

$$\begin{aligned}
\mathbb{E}Y_j &= \mathbb{E}\left(\sum_i M_{ji}x_i\right) = p\sum_i x_i \\
\mathbb{E}Y_j^2 &= \mathbb{E}\left(\sum_i M_{ji}x_i\right)^2 = \sum_{i,i'} \mathbb{E}(M_{ji}M_{ji'})x_i x_{i'} \\
&= p\sum_i x_i^2 + p^2 \sum_{i \neq i'} x_i x_{i'} \; = \; p\|x\|^2 + p^2((\mathbf{1}\cdot x)^2 - \|x\|^2)
\end{aligned}$$

Proof of Lemma 2

Applying Chebyshev's bound, we have that for any $t > 0$,

$$\begin{aligned}
\Pr\left(\left|\|Y\|^2 - \mathbb{E}\|Y\|^2\right| \geq t\right) &\leq \frac{\mathrm{var}(\|Y\|^2)}{t^2} \\
&= \frac{\mathrm{var}(Y_1^2 + \cdots + Y_m^2)}{t^2} \; = \; \frac{m \cdot \mathrm{var}(Y_1^2)}{t^2}.
\end{aligned}$$

Using $t = \epsilon\mathbb{E}\|Y\|^2 = \epsilon m \mathbb{E}Y_1^2$ then gives

$$\Pr\left(\left|\|Y\|^2 - \mathbb{E}\|Y\|^2\right| \geq \epsilon\mathbb{E}\|Y\|^2\right) \; \leq \; \frac{1}{\epsilon^2 m} \cdot \frac{\mathrm{var}(Y_1^2)}{(\mathbb{E}Y_1^2)^2} \tag{2}$$

It remains to bound the last ratio. We already have $\mathbb{E}Y_1^2$ from Lemma 1. To compute $\text{var}(Y_1^2)$, we begin with $\mathbb{E}Y_1^4$:

$$\begin{aligned}
\mathbb{E}Y_1^4 &= \mathbb{E}(M_{11}x_1 + \cdots + M_{1d}x_d)^4 \\
&= \sum_i \mathbb{E}[M_{1i}^4 x_i^4] + 4\sum_{i\neq j}\mathbb{E}[M_{1i}x_i M_{1j}^3 x_j^3] + 3\sum_{i\neq j}\mathbb{E}[M_{1i}^2 x_i^2 M_{1j}^2 x_j^2] \\
&\quad + 6\sum_{i\neq j\neq k}\mathbb{E}[M_{1i}^2 x_i^2 M_{1j}x_j M_{1k}x_k] + \sum_{i\neq j\neq k\neq \ell}\mathbb{E}[M_{1i}x_i M_{1j}x_j M_{1k}x_k M_{1\ell}x_\ell] \\
&= p\sum_i x_i^4 + 4p^2\sum_{i\neq j} x_i x_j^3 + 3p^2\sum_{i\neq j} x_i^2 x_j^2 + 6p^3\sum_{i\neq j\neq k} x_i^2 x_j x_k + p^4\sum_{i\neq j\neq k\neq\ell} x_i x_j x_k x_\ell.
\end{aligned}$$

This is maximized when all the x_i are positive, so

$$\begin{aligned}
\mathbb{E}Y_1^4 &\le p\sum_i x_i^4 + 4p^2\sum_{i,j} x_i x_j^3 + 3p^2\sum_{i,j} x_i^2 x_j^2 + 6p^3\sum_{i,j,k} x_i^2 x_j x_k + p^4\sum_{i,j,k,\ell} x_i x_j x_k x_\ell \\
&= p\|x\|_4^4 + 4p^2\sum_{i,j} x_i x_j^3 + 3p^2\|x\|^4 + 6p^3\|x\|^2(\mathbf{1}\cdot x)^2 + p^4(\mathbf{1}\cdot x)^4 \\
&\le p\|x\|_4^4 + 4p^2 d\|x\|_4^4 + 3p^2\|x\|^4 + 6p^3\|x\|^2(\mathbf{1}\cdot x)^2 + p^4(\mathbf{1}\cdot x)^4 \\
&\le p(1+4c)\|x\|_4^4 + 3p^2(1+2c)\|x\|^4 + p^4(\mathbf{1}\cdot x)^4,
\end{aligned}$$

where we have twice invoked the inequality $2ab \le a^2 + b^2$ to get

$$\sum_{i,j} x_i x_j^3 = \frac{1}{2}\sum_{i,j}(x_i x_j^3 + x_j x_i^3) = \frac{1}{2}\sum_{i,j} x_i x_j(x_i^2 + x_j^2) \le \frac{1}{2}\sum_{i,j}\frac{1}{2}(x_i^2+x_j^2)^2 \le \frac{1}{2}\sum_{i,j}(x_i^4+x_j^4) = d\sum_i x_i^4.$$

and used the Cauchy-Schwarz inequality to get $(\mathbf{1}\cdot x)^2 \le d\|x\|^2$. Continuing,

$$\text{var}(Y_1^2) \;=\; \mathbb{E}Y_1^4 - (\mathbb{E}Y_1^2)^2 \;\le\; 5cp\|x\|_4^4 + 9cp^2\|x\|^4.$$

Plugging this into (2) then gives the bound.

Proof of Lemma 3

Suppose $X_1, \ldots, X_d$ are i.i.d. draws from an exponential distribution with parameter λ. It is well-known that for any positive integer k,

$$\mathbb{E}X_1^k = \frac{k!}{\lambda^k}.$$

Thus:

$$\mathbb{E}\|X\|_2^2 = \mathbb{E}(X_1^2 + \cdots + X_d^2) = d\,\mathbb{E}X_1^2 = \frac{2d}{\lambda^2}$$
$$\mathbb{E}\|X\|_4^4 = \mathbb{E}(X_1^4 + \cdots + X_d^4) = d\,\mathbb{E}X_1^4 = \frac{24d}{\lambda^4}$$

Part (a) of the lemma follows immediately.

Pick any positive integer c. To show that $\|X\|_c^c = X_1^c + \cdots + X_d^c$ is concentrated around its expected value, we use Chebyshev's inequality. First, we compute the variance of X_1^c,

$$\text{var}(X_1^c) = \mathbb{E}X_1^{2c} - (\mathbb{E}X_1^c)^2 = \frac{(2c)!}{\lambda^{2c}} - \left(\frac{c!}{\lambda^c}\right)^2 = \frac{(2c)! - (c!)^2}{\lambda^{2c}},$$

so that $\text{var}(\|X\|_c^c) = \text{var}(X_1^c + \cdots + X_d^c) = d\,\text{var}(X_1^c)$ is exactly d times this. Thus, for any $\epsilon > 0$,

$$\begin{aligned}\Pr\left(|\|X\|_c^c - \mathbb{E}\|X\|_c^c| \geq \epsilon\,\mathbb{E}\|X\|_c^c\right) \leq \frac{\text{var}(\|X\|_c^c)}{\epsilon^2(\mathbb{E}\|X\|_c^c)^2} &= \frac{d\,\text{var}(X_1^c)}{\epsilon^2(d\,\mathbb{E}X_1^c)^2} \\ &= \frac{d((2c)! - (c!)^2)}{\lambda^{2c}} \cdot \frac{\lambda^{2c}}{\epsilon^2 d^2 (c!)^2} \leq \frac{2^{2c}}{\epsilon^2 d}.\end{aligned}$$

Part (b) of the lemma follows by choosing a value of ϵ that makes this expression $\leq \delta$.

Varying the density of the binary random projection

We varied the number of projection neurons (PNs) each Kenyon cell (KC) samples from and evaluated its effect on nearest-neighbors retrieval performance (Fig. S1). In the fly, each KC samples from roughly 10% of the PNs (6 out of 50). We tried setting this value to 1% and to 50%. For some datasets, 1% sufficed, though this is likely more sensitive to noise. Across all datasets, the most consistent performance was obtained when sampling 10%, with no improvement in performance at 50%. Sampling 10% thus achieved the best trade-off between computational efficiency and performance.

See Litwin-Kumar et al. (*35*) for a perspective of how sampling affects associative learning.

Empirical analysis on the GIST dataset

We tested the fly algorithm in even higher dimensions ($d = 960$, GIST image dataset (*31*)) and found a similar trend in performance (Fig. S2). Thus, although designed biologically for $d = 50$, the fly algorithm is scalable.

Binary locality-sensitive hashing

We used the fly algorithm to implement binary locality-sensitive hashing (*36*), where the LSH function $h : \mathbb{R}^d \rightarrow \mathbb{Z}^m$. In other words, instead of using the values of the top k Kenyon cells as the tag, we used their indices, setting those indices to 1 and the remaining to 0. For LSH, binary hashes are typically computed by: $y = \mathrm{sgn}(Mx)$, where M is a dense, i.i.d. Gaussian random matrix, and x is the input. If the $(i, j)^{\text{th}}$ element of Mx is greater than 0, y_{ij} is set to 1, and 0 otherwise. In other words, each Kenyon cell is binarized to 0 or 1 based on whether its value is ≤ 0 or > 0, respectively.

For binary hashing, the fly algorithm performed better than traditional binary LSH across all four datasets (Fig. S3).

Discussion

Why not use the projection neurons (PNs) as the tag, as opposed to the sparse Kenyon cells (KCs)? Biologically, the answer is that the point of the mushroom body (where the Kenyon cells lie) is to learn odors, and the only synapses whose strength can be modified are the ones that are active. Because most PNs fire in response to most odors, if the PN signal were used for learning, each odor would modify the synaptic strength associated with most other odors and so a specific odor could not be discriminated. The only solution is to have non-overlapping KC tags so that just the synapses associated with one odor can be modified without modifying synapses associated with other odors.

Algorithmically, one could similarly ask, why not use the input data itself as the hash tag? The answer is that random projections provide better theoretical guarantees and better bounds. Moreover, in LSH applications, it is often necessary to build multiple hash tables to boost recall. Some randomization is thus critical because it allows construction of multiple independent hash functions.

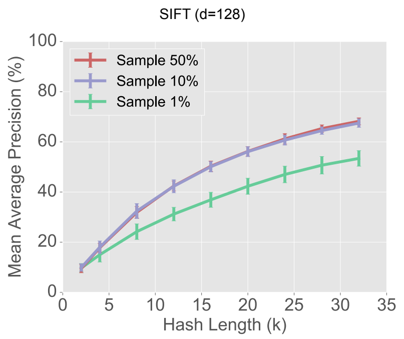
Additional related work in theoretical computer science. The theoretical computer science community has studied sparse random projections in some related contexts. For example, Kane and Nelson *(18)* study matrices whose entries are both positive and negative, with mean zero. The resulting statistics are different from those we encounter; for instance, their estimators are unbiased. Shi et al. (*37*) study a process similar to the one here, in the context of speeding up computations relating to kernel support vector machines. They also use a sparse binary matrix, with the additional condition that in each consecutive block of 1/p rows, there is exactly one non-zero entry per column. Their goal is to bound the effect of this projection on dot products (rather than distances), and they obtain error bounds that are qualitatively different from ours: additive rather than multiplicative, for instance, and not tuned to the sparsity of the input signal.

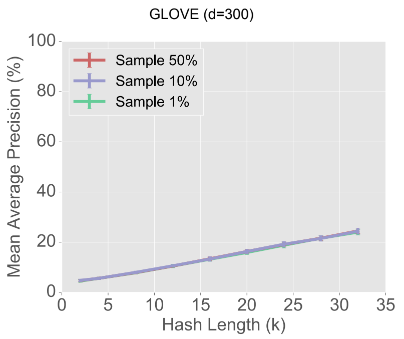
Andoni et al. (27) use sparse random linear maps as a preprocessing step to reduce the dimensionality of sparse data; this contrasts with the fly's use of these maps, which has the effect of producing a sparse, high-dimensional representation. To our knowledge, LSH families themselves have not been based on sparse binary projections.

When does the fly's algorithm perform best? Empirically, we found that the fly's algorithm works best when the distribution of feature values for each input has a high-firing rate tail (e.g., a Gaussian or exponential). Kenyon cells that sample PNs at the tail of the distribution are least probable to fire at the same rate for a different input, and these KCs end up constituting the tag following the winner-take-all step. Thus, using these KCs as the tag serves as a strong discriminator between different inputs, and a strong indicator for similarity if the inputs are indeed very similar. Interestingly, such a distribution is exactly what the PNs in the brain produce: an exponential distribution of firing rates with a high-firing rate tail.

How are winner-take-all (WTA) networks implemented in the brain? In the fly olfactory circuit, the WTA network requires just one neuron (APL), which receives feed-forward excitation from Kenyon cells, and provides proportional feed-back inhibition to silence the slowest-firing neurons. This method of sparsification (using inhibition) appears to be more robust than an alternative where KCs apply a threshold to PN input *(4)*. In the mouse olfactory circuit, the analogs of the Kenyon cells (called semilunar cells) also form a sparse tag, but there are at least five classes of inhibitory neurons that help generate this sparse tag (*38*); it remains unclear computationally how this is accomplished. More generally, generating biologically-plausible WTA networks is an active area of research, with many proposals (*39,40*) though there is debate about how well these models match with actual neural anatomy and cell types.

Learning and data-dependent hashing. Following the fly, we focused on *data-independent* hashing; that is, hash functions that do not learn from prior data nor use prior data in any way when deriving the tag. Recently, many classes of *data-dependent* LSH families have been proposed, including principal components analysis hashing (PCA hashing (*41*)), spectral hashing (*42*), semantic hashing (*43*), deep hashing (*44*), and others (*45*) (reviewed by Wang et al (*46*)). Biologically, learning also plays an important role, especially in the mammalian olfactory circuit, where plasticity and neurogenesis both occur in early processing stages (*47,48*). In the fly, learning occurs primarily in the mushroom body at Kenyon cell synapses onto mushroom body output neurons *(2)*, though there is some evidence that learning can also occur in the glomeruli (*49*). Understanding the different contributions to learning made by each of these regions remains an open biological problem. Computationally, understanding how hash functions can be learned and modified over time based on online experience, as opposed to using a fixed offline database as is often the assumption made by data-dependent hashing algorithms, remains an important question.





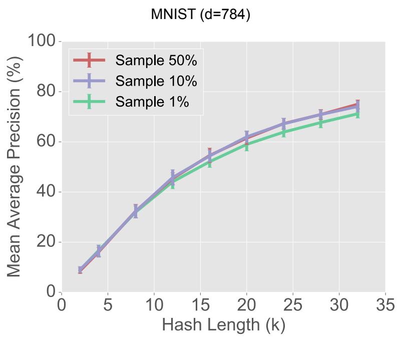
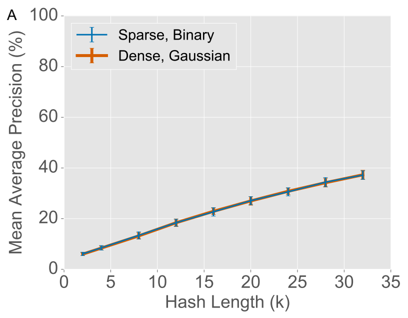
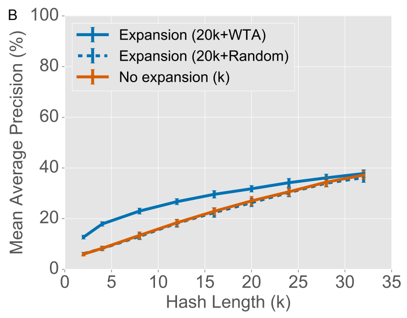


Figure S1: **Comparison of different sampling levels in the sparse, binary random projection.**





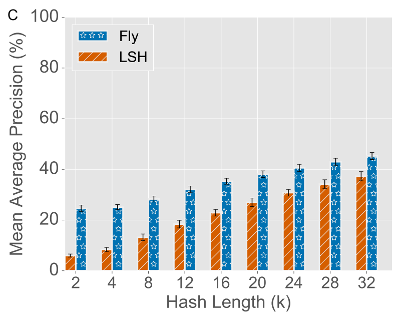


Figure S2: **Analysis of the GIST dataset.** (A) Similar performance of sparse, binary compared to dense, Gaussian random projections. (B) Performance gains using winner-take-all compared to random tag selection. (C) Further performance gains for the fly algorithm with a $10d$ expansion compared to a $20k$ expansion in (B).

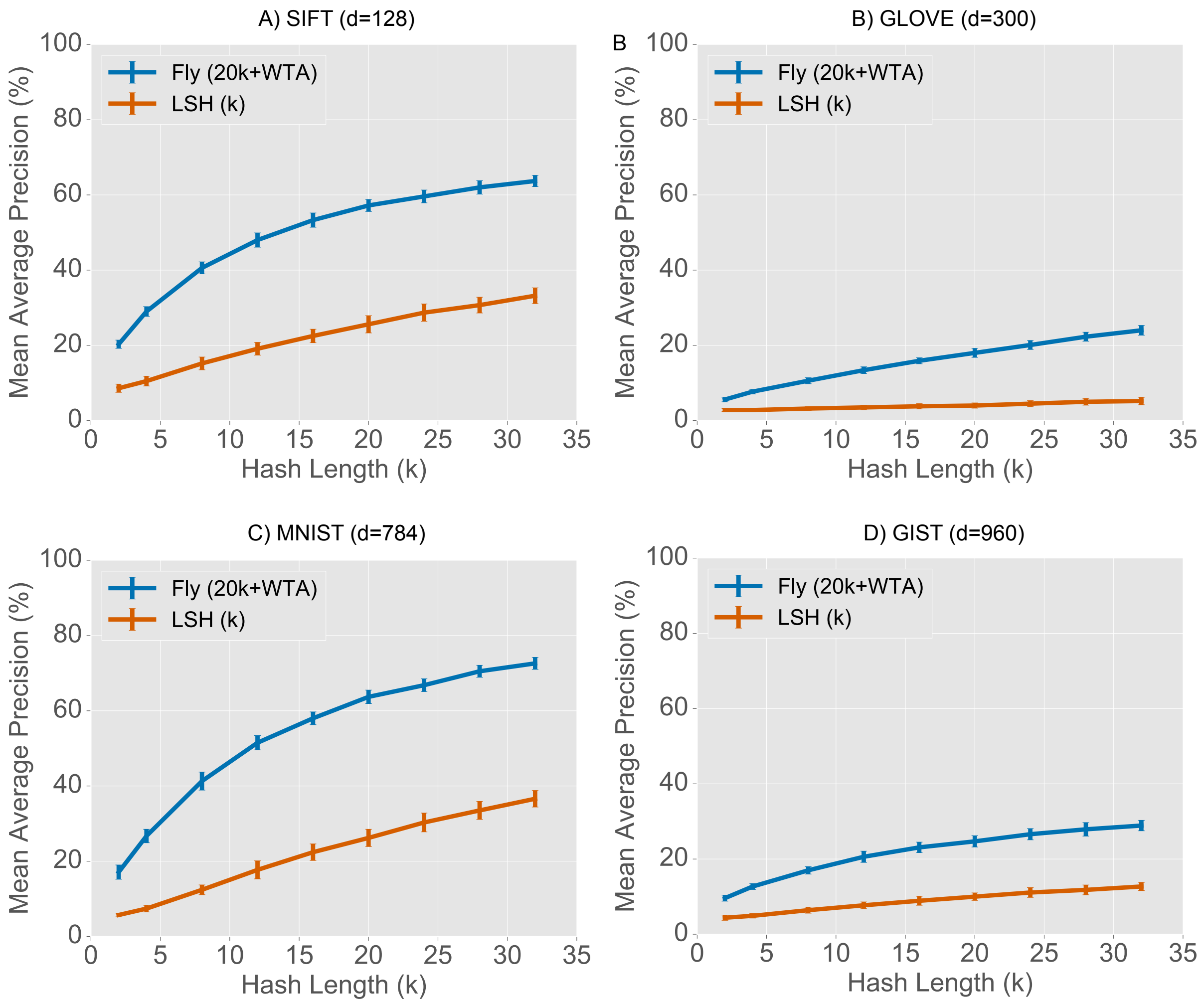


Figure S3: **The fly versus LSH using binary locality-sensitive hashing.**

References and Notes

1. C. F. Stevens, What the fly's nose tells the fly's brain. *Proc. Natl. Acad. Sci. U.S.A.* **112**, 9460–9465 (2015). doi:10.1073/pnas.1510103112 Medline

2. D. Owald, S. Waddell, Olfactory learning skews mushroom body output pathways to steer behavioral choice in *Drosophila*. *Curr. Opin. Neurobiol.* **35**, 178–184 (2015). doi:10.1016/j.conb.2015.10.002 Medline

3. G. C. Turner, M. Bazhenov, G. Laurent, Olfactory representations by *Drosophila* mushroom body neurons. *J. Neurophysiol.* **99**, 734–746 (2008). doi:10.1152/jn.01283.2007 Medline

4. A. C. Lin, A. M. Bygrave, A. de Calignon, T. Lee, G. Miesenböck, Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nat. Neurosci.* **17**, 559–568 (2014). doi:10.1038/nn.3660 Medline

5. M. Papadopoulou, S. Cassenaer, T. Nowotny, G. Laurent, Normalization for sparse encoding of odors by a wide-field interneuron. *Science* **332**, 721–725 (2011). doi:10.1126/science.1201835 Medline

6. E. A. Hallem, J. R. Carlson, Coding of odors by a receptor repertoire. *Cell* **125**, 143–160 (2006). doi:10.1016/j.cell.2006.01.050 Medline

7. C. F. Stevens, A statistical property of fly odor responses is conserved across odors. *Proc. Natl. Acad. Sci. U.S.A.* **113**, 6737–6742 (2016). doi:10.1073/pnas.1606339113 Medline

8. S. R. Olsen, V. Bhandawat, R. I. Wilson, Divisive normalization in olfactory population codes. *Neuron* **66**, 287–299 (2010). doi:10.1016/j.neuron.2010.04.009 Medline

9. S. J. Caron, V. Ruta, L. F. Abbott, R. Axel, Random convergence of olfactory inputs in the *Drosophila* mushroom body. *Nature* **497**, 113–117 (2013). doi:10.1038/nature12063 Medline

10. A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* **51**, 117 (2008). doi:10.1145/1327452.1327494

11. A. Gionis, P. Indyk, R. Motwani, in *VLDB'99, Proceedings of the 25th International Conference on Very Large Data Bases*, M. P. Atkinson *et al.*, Eds. (Morgan Kaufman, 1999), pp. 158–529.

12. H. Samet, *Foundations of Multidimensional and Metric Data Structures* (Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, Morgan Kaufmann, 2005).

13. Materials and methods are available as supplementary materials.

14. W. Johnson, J. Lindenstrauss, in *Conference on Modern Analysis and Probability*, R. Beals, A. Beck, A. Bellow, A. Hajian, Eds., vol. 26 of *Contemporary Mathematics* (American Mathematical Society, 1984), pp. 189–206.

15. S. Dasgupta, A. Gupta, An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures Algorithms* **22**, 60–65 (2003). doi:10.1002/rsa.10073

16. D. Achlioptas, Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *J. Comput. Syst. Sci.* **66**, 671–687 (2003). doi:10.1016/S0022-0000(03)00025-4

17. Z. Allen-Zhu, R. Gelashvili, S. Micali, N. Shavit, Sparse sign-consistent Johnson-Lindenstrauss matrices: Compression with neuroscience-based constraints. *Proc. Natl. Acad. Sci. U.S.A.* **111**, 16872–16876 (2014). doi:10.1073/pnas.1419100111 Medline

18. D. Kane, J. Nelson, Sparser Johnson-Lindenstrauss transforms. *J. Assoc. Comput. Mach.* **61**, 4 (2014). doi:10.1145/2559902

19. Y. Lin, R. Jin, D. Cai, S. Yan, X. Li, in *2013 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE Computer Society, 2013), pp. 446–451.

20. M. S. Charikar, in *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, *STOC '02* [Association for Computing Machinery (ACM), 2002], pp. 380–388.

21. C. Pehlevan, D. B. Chklovskii, in *NIPS'15, Proceedings of the 28th International Conference on Neural Information Processing Systems* (MIT Press, 2015), pp. 2269–2277.

22. R. Spring, A. Shrivastava, Scalable and sustainable deep learning via randomized hashing. arXiv:1602.08194 [stat.ML] (26 February 2016).

23. M. Slaney, Y. Lifshits, J. He, Optimal parameters for locality-sensitive hashing. *Proc. IEEE* **100**, 2604–2623 (2012). doi:10.1109/JPROC.2012.2193849

24. Q. Lv, W. Josephson, Z. Wang, M. Charikar, K. Li, in *VLDB '07, Proceedings of the 33rd International Conference on Very Large Data Bases* (ACM, 2007), pp. 950–961.

25. P. Li, M. Mitzenmacher, A. Shrivastava, in *Proceedings of the 31st International Conference on Machine Learning* (Proceedings of Machine Learning Research, 2014), pp. 676–684.

26. A. Dasgupta, R. Kumar, T. Sarlos, in *KDD '11, The 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2011), pp. 1073–1081.

27. A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, L. Schmidt, in *NIPS'15, Proceedings of the 28th International Conference on Neural Information Processing Systems* (MIT Press, 2015), pp. 1225–1233.

28. A. Broder, in *Proceedings of the Compression and Complexity of Sequences 1997* (IEEE Computer Society, 1997), p. 21.

29. J. Yagnik, D. Strelow, D. A. Ross, R.-s. Lin, in *2011 International Conference on Computer Vision* (IEEE Computer Society, 2011), pp. 2431–2438.

30. L. G. Valiant, What must a global theory of cortex explain? *Curr. Opin. Neurobiol.* **25**, 15–19 (2014). doi:10.1016/j.conb.2013.10.006 Medline

31. H. Jégou, M. Douze, C. Schmid, Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**, 117–128 (2011). doi:10.1109/TPAMI.2010.57 Medline

32. J. Pennington, R. Socher, C. D. Manning, in *EMNLP 2014: The 2014 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, 2014), pp. 1532–1543.

33. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998). doi:10.1109/5.726791

34. D. L. Donoho, Compressed sensing. *IEEE Trans. Inf. Theory* **52**, 1289–1306 (2006). doi:10.1109/TIT.2006.871582

35. A. Litwin-Kumar, K. D. Harris, R. Axel, H. Sompolinsky, L. F. Abbott, Optimal degrees of synaptic connectivity. *Neuron* **93**, 1153–1164.e7 (2017). doi:10.1016/j.neuron.2017.01.030 Medline

36. J. Wang, H. T. Shen, J. Song, J. Ji, Hashing for similarity search: A survey. arXiv:1408.2927 [cs.DS] (13 August 2014).

37. Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, S. V. N. Vishwanathan, Hash kernels for structured data. *J. Mach. Learn. Res.* **10**, 2615–2637 (2009).

38. J. M. Bekkers, N. Suzuki, Neurons and circuits for odor processing in the piriform cortex. *Trends Neurosci.* **36**, 429–438 (2013). doi:10.1016/j.tins.2013.04.005 Medline

39. Y. Chen, Mechanisms of winner-take-all and group selection in neuronal spiking networks. *Front. Comput. Neurosci.* **11**, 20 (2017). doi:10.3389/fncom.2017.00020 Medline

40. N. A. Lynch, C. Musco, M. Parter, Computational tradeoffs in biological neural networks: Self-stabilizing winner-take-all networks. arXiv:1610.02084 [cs.NE] (6 October 2016).

41. Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: A Procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 2916–2929 (2013). doi:10.1109/TPAMI.2012.193 Medline

42. Y. Weiss, A. Torralba, R. Fergus, in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, L. Bottou, Eds. (Curran Associates, 2009), pp. 1753–1760.

43. R. Salakhutdinov, G. Hinton, Semantic hashing. *Int. J. Approx. Reason.* **50**, 969–978 (2009). doi:10.1016/j.ijar.2008.11.006

44. H. Zhu, M. Long, J. Wang, Y. Cao, in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (AAAI Press, 2016), pp. 2415–2421.

45. K. Zhao, H. Lu, J. Mei, in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence* (AAAI Press, 2014), pp. 2874–2880.

46. J. Wang, T. Zhang, J. Song, N. Sebe, H. T. Shen, A survey on learning to hash. arXiv:1606.00185 [cs.CV] (1 June 2017).

47. D. A. Wilson, R. M. Sullivan, Cortical processing of odor objects. *Neuron* **72**, 506–519 (2011). doi:10.1016/j.neuron.2011.10.027 Medline

48. P. M. Lledo, M. Alonso, M. S. Grubb, Adult neurogenesis and functional plasticity in neuronal circuits. *Nat. Rev. Neurosci.* **7**, 179–193 (2006). doi:10.1038/nrn1867 Medline

49. D. Yu, A. Ponomarev, R. L. Davis, Altered representation of the spatial code for odors after olfactory classical conditioning; memory trace formation by synaptic recruitment. *Neuron* **42**, 437–449 (2004). doi:10.1016/S0896-6273(04)00217-X Medline