BIO-INSPIRED HASHING FOR UNSUPERVISED SIMILARITY SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

The fruit fly Drosophila's olfactory circuit has inspired a new locality sensitive hashing (LSH) algorithm, `FlyHash`. In contrast with classical LSH algorithms that produce low dimensional hash codes, `FlyHash` produces sparse high-dimensional hash codes and has also been shown to have superior empirical performance compared to classical LSH algorithms in similarity search. However, `FlyHash` uses random projections and cannot *learn* from data. Building on inspiration from `FlyHash` and the ubiquity of sparse expansive representations in neurobiology, our work proposes a novel hashing algorithm `BioHash` that produces sparse high dimensional hash codes in a *data-driven* manner. We show that `BioHash` outperforms previously published benchmarks for various hashing methods. Since our learning algorithm is based on a local and biologically plausible synaptic plasticity rule, our work provides evidence for the proposal that LSH might be a computational reason for the abundance of sparse expansive motifs in a variety of biological systems. We also propose a convolutional variant `BioConvHash` that further improves performance. From the perspective of computer science, `BioHash` and `BioConvHash` are *fast*, *scalable* and yield compressed binary representations that are useful for similarity search.

1 INTRODUCTION

Sparse expansive representations are ubiquitous in neurobiology. Expansion means that a high-dimensional input is mapped to an even higher dimensional secondary representation. Such expansion is often accompanied by a sparsification of the activations: dense input data is mapped into a sparse code, where only a small number of secondary neurons respond to a given stimulus.

A classical example of the sparse expansive motif is the Drosophila fruit fly olfactory system. In this case, approximately 50 projection neurons send their activities to about 2500 Kenyon cells (1), thus accomplishing an approximately 50x expansion. An input stimulus typically activates approximately 50% of projection neurons, and less than 10% Kenyon cells (1), providing an example of significant sparsification of the expanded codes. Another example is the rodent olfactory circuit. In this system, dense input from the olfactory bulb is projected into piriform cortex, which has 1000x more neurons than the number of glomeruli in the olfactory bulb. Only about 10% of those neurons respond to a given stimulus (2). A similar motif is found in rat's cerebellum and hippocampus (3).

From the computational perspective, expansion is helpful for increasing the number of classification decision boundaries by a simple perceptron (4) or increasing memory storage capacity in models of associative memory (5). Additionally, sparse expansive representations have been shown to reduce intrastimulus variability and the overlaps between representations induced by distinct stimuli (6). Sparseness has also been shown to increase the capacity of models of associative memory (7).

The goal of the present work is to use this "biological" inspiration about sparse expansive motifs, as well as local Hebbian learning, for designing a novel hashing algorithm `BioHash` that can be used in similarity search. Below we describe the task, the algorithm, and demonstrate that `BioHash` improves retrieval performance on common benchmark datasets.

Similarity search and LSH. In similarity search, given a query $q \in \mathbb{R}^d$, a similarity measure $\text{sim}(q, x)$, and a database $X \in \mathbb{R}^{n \times d}$ containing n items, the objective is to retrieve a ranked list of R items from the database most similar to q. When data is high-dimensional (e.g. images/documents)

and the databases are large (millions or billions items), this is a computationally challenging problem. However, approximate solutions are generally acceptable, with Locality Sensitive Hashing (LSH) being one such approach (8). Similarity search approaches maybe unsupervised or supervised. Since labelled information for extremely large datasets is infeasible to obtain, our work focuses on the unsupervised setting. In LSH (9; 10), the idea is to encode each database entry x (and query q) with a binary representation $h(x)$ ($h(q)$ respectively) and to retrieve R entries with smallest Hamming distances $d_H(h(x), h(q))$. Intuitively, (see (10), for a formal definition), a hash function $h : \mathbb{R}^d \rightarrow \{-1, 1\}^m$ is said to be *locality sensitive*, if similar (dissimilar) items x_1 and x_2 are close by (far apart) in Hamming distance $d_H(h(x_1), h(x_2))$. LSH algorithms are of fundamental importance in computer science, with applications in similarity search, data compression and machine learning (11).

Drosophila olfactory circuit and FlyHash. In classical LSH approaches, the data dimensionality d is much larger than the embedding space dimension m, resulting in low-dimensional hash codes (8; 9; 10). In contrast, a new family of hashing algorithms has been proposed (3) where $m \gg d$, but the secondary representation is highly sparse with only a small number k of m units being active, see figure 1. We call this algorithm `FlyHash` in this paper, since it is motivated by the computation carried out by the fly's olfactory circuit. The expansion from the d dimensional input space into an m dimensional secondary representation is carried out using a random set of weights W (3; 12). The resulting high dimensional representation is sparsified by k-Winner-Take-All (k-WTA) feedback inhibition in the hidden layer resulting in top $\sim 5\%$ of units staying active (13; 14).

While `FlyHash` uses random synaptic weights, sparse expansive representations are not necessarily random (6), perhaps not even in the case of Drosophila (15; 16). Moreover, using synaptic weights that are learned from data might help to further improve locality sensitive property of `FlyHash`. Thus, it is important to investigate the role of learned synapses on the hashing performance. `BioHash` utilizes synaptic weights learned from the data and demonstrates improved performance compared to random weights used in `FlyHash`.

Not only "biological" inspiration can lead to improving hashing techniques, but the opposite might also be true. One of the statements of the present paper is that `BioHash` satisfies locality sensitive property, and, at the same time, utilizes a biologically plausible learning rule for synaptic weights (17). This provides evidence toward the proposal that the reason why sparse expansive representations are so common in biological organisms is because they perform locality sensitive hashing. In other words, they cluster similar stimuli together and push distinct stimuli far apart. Thus, our work provides evidence toward the proposal that LSH might be a fundamental computational principle utilized by the sparse expansive circuits Fig. 1 (right). Importantly, learning of synapses must be neurobiologically plausible (the synaptic plasticity rule should be local).

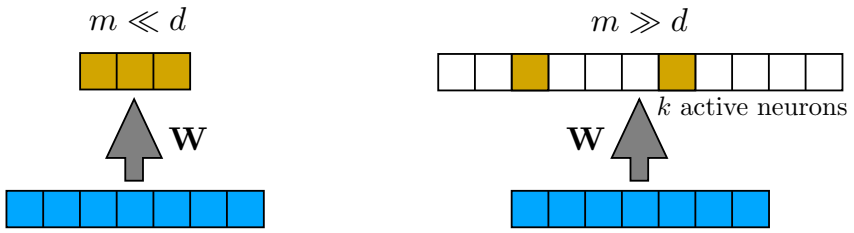

Figure 1: It is convenient to think about different hashing algorithms in terms of representational contraction (large dimension of input is mapped into a smaller dimensional latent space), or expansion (large dimensional input is mapped into an even larger dimensional latent space). The projections can be random or data driven.

Contributions. Building on inspiration from `FlyHash` and more broadly the ubiquity of sparse, expansive representations in neurobiology, our work proposes a novel hashing algorithm `BioHash` that produces sparse high dimensional hash codes in a *data-driven* (as opposed to random synapses in `FlyHash`) manner; with learning of synapses in a neurobiologically plausible way, we provide an existence proof for the proposal that LSH maybe a fundamental computational principle in neural circuits (3) in the context of learned synapses. We incorporated convolutional structure into `BioHash`, resulting in a hashing with improved performance compared to previously published benchmarks. From the perspective of computer science, we show that `BioHash` is simple, scalable

to large datasets and demonstrates good performance for similarity search. Interestingly, `BioHash` outperforms a number of recent SOTA deep hashing methods trained via backpropogation.

2 Approximate Similarity Search via BioHashing

Formally, if we denote a data point as $x \in \mathbb{R}^d$, we seek a binary hash code $y \in \{-1, 1\}^m$. We define the hash length of a binary code as k, if the exact Hamming distance computation is $O(k)$. Below we present our bio-inspired hashing algorithm.

2.1 Bio-inspired Hashing (BioHash)

We adopt a biologically plausible unsupervised algorithm for representation learning from (17). Denote the synapses from the input layer to the hash layer as $\mathbf{W} \in \mathbb{R}^{m \times d}$. The learning dynamics for the synapses of an individual neuron μ, denoted by $W_{\mu i}$, is given by

$$\tau \frac{dW_{\mu i}}{dt} = g\Big[\text{Rank}\big(\langle W_\mu, x\rangle_\mu\big)\Big]\Big(x_i - \langle W_\mu, x\rangle_\mu W_{\mu i}\Big), \tag{1}$$

where

$$g(\mu) = \begin{cases} 1, & \mu = 1 \\ -\Delta, & \mu = r \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

and $\langle x, y\rangle_\mu = \sum_{i,j} \eta^\mu_{i,j} x_i y_j$, with $\eta^\mu_{i,j} = |W_{\mu i}|^{p-2}\delta_{ij}$, where δ_{ij} is Kronecker delta and τ is the time scale of the learning dynamics. It can be shown that the synapses converge to a unit ($p-$norm) sphere (17). The training dynamics can be shown to minimize the following energy function

$$E = -\sum_A \sum_{\mu=1}^m g\Big[\text{Rank}\big(\langle W_\mu, x^A\rangle_\mu\big)\Big] \frac{\langle W_\mu, x^A\rangle_\mu}{\langle W_\mu, W_\mu\rangle_\mu^{\frac{p-1}{p}}}, \tag{3}$$

where A indexes the training example. Note that the training dynamics do not perform gradient descent, i.e $\dot{W}_\mu \neq \nabla_{W_\mu} E$. However, time derivative of the energy function under dynamics (1) is always negative (we show this for the case $\Delta = 0$ below),

$$\begin{aligned} \tau \frac{dE}{dt} &= -\sum_A \frac{\tau(p-1)}{\langle W_{\hat{\mu}}, W_{\hat{\mu}}\rangle_{\hat{\mu}}^{\frac{p-1}{p}+1}} \Big[\langle \frac{dW_{\hat{\mu}}}{dt}, x^A\rangle_{\hat{\mu}} \langle W_{\hat{\mu}}, W_{\hat{\mu}}\rangle_{\hat{\mu}} - \langle W_{\hat{\mu}}, x^A\rangle_{\hat{\mu}} \langle \frac{dW_{\hat{\mu}}}{dt}, W_{\hat{\mu}}\rangle_{\hat{\mu}}\Big] = \\ &= -\sum_A \frac{\tau(p-1)}{\langle W_{\hat{\mu}}, W_{\hat{\mu}}\rangle_{\hat{\mu}}^{\frac{p-1}{p}+1}} \Big[\langle x^A, x^A\rangle_{\hat{\mu}} \langle W_{\hat{\mu}}, W_{\hat{\mu}}\rangle_{\hat{\mu}} - \langle W_{\hat{\mu}}, x^A\rangle_{\hat{\mu}}^2\Big] \leq 0, \end{aligned} \tag{4}$$

where Cauchy-Schwartz inequality is used. For every training example A the index of the activated hidden unit is defined as

$$\hat{\mu} = \arg\max_\mu \big[\langle W_\mu, x^A\rangle_\mu\big]. \tag{5}$$

Thus, the energy function decreases during learning. A similar result can be shown for $\Delta \neq 0$.

After the learning-phase is complete, the hash code is generated, as in FlyHash, via WTA sparsification: for a given query x we generate a hash code $y \in \{-1, 1\}^m$ as

$$y_\mu = \begin{cases} 1, & \langle W_\mu, x\rangle_\mu \text{ is in top } k \\ -1, & \text{otherwise.} \end{cases} \tag{6}$$

Thus, the hyperparameters of the method are p, r, m and Δ. Note that the synapses are updated based only on pre- and post-synaptic activations resulting in Hebbian or anti-Hebbian updates. Many "unsupervised" learning to hash approaches provide a sort of "weak supervision" in the form of similarities evaluated in the feature space of deep CNNs trained on ImageNet (18) to achieve good performance. `BioHash` does not assume such information is provided and is completely unsupervised.

2.2 INTUITION BEHIND THE LEARNING ALGORITHM

An intuitive way to think about the learning algorithm is to view the hidden units as particles that are attracted to local peaks of the density of the data, and that simultaneously repel each other. To demonstrate this, it is convenient to think about input data as randomly sampled points from a continuous distribution. Consider the case when $p = 2$ and $\Delta = 0$. In this case, the energy function can be written as (since for $p = 2$ the inner product does not depend on the weights, we drop the subscript μ of the inner product)

$$E = -\frac{1}{n}\sum_A \frac{\langle W_{\hat{\mu}}, x^A\rangle}{\langle W_{\hat{\mu}}, W_{\hat{\mu}}\rangle^{\frac{1}{2}}} = -\int \prod_i dv_i \frac{1}{n}\sum_A \delta(v_i - x_i^A)\frac{\langle W_{\hat{\mu}}, v\rangle}{\langle W_{\hat{\mu}}, W_{\hat{\mu}}\rangle^{\frac{1}{2}}} = \\ = -\int \prod_i dv_i \rho(v)\frac{\langle W_{\hat{\mu}}, v\rangle}{\langle W_{\hat{\mu}}, W_{\hat{\mu}}\rangle^{\frac{1}{2}}}, \tag{7}$$

where we introduced a continuous density of data $\rho(v)$. Furthermore, consider the case of $d = 2$, and imagine that the data lies on a unit circle. In this case the density of data can be parametrized by a single angle φ. Thus, the energy function can be written as

$$E = -\int_{-\pi}^{\pi} d\varphi \rho(\varphi)\cos(\varphi - \varphi_{\hat{\mu}}), \quad \text{where} \quad \hat{\mu} = \arg\max_{\mu}\big[\cos(\varphi - \varphi_\mu)\big] \tag{8}$$

It is instructive to solve a simple case when the data follows an exponential distribution concentrated around zero angle with the decay length σ (α is a normalization constant),

$$\rho(\varphi) = \alpha e^{-\frac{|\varphi|}{\sigma}}. \tag{9}$$

In this case, the energy (8) can be calculated exactly for any number of hidden units m. However, minimizing over the position of hidden units cannot be done analytically for general m. To further simplify the problem consider the case when the number of hidden units m is small. For $m = 2$ the energy is equal to

$$E = -\alpha\frac{\sigma(1 + e^{-\frac{\pi}{\sigma}})}{1 + \sigma^2}\Big(\cos(\varphi_1) + \sigma\sin(\varphi_1) + \cos(\varphi_2) - \sigma\sin(\varphi_2)\Big) \tag{10}$$

Thus, in this simple case the energy is minimized when

$$\varphi_{1,2} = \pm\arctan(\sigma) \tag{11}$$

In the limit when the density of data is concentrated around zero angle ($\sigma \to 0$) the hidden units are attracted to the origin and $|\varphi_{1,2}| \approx \sigma$. In the opposite limit ($\sigma \to \infty$) the data points are uniformly distributed on the circle. The resulting hidden units are then organized to be on the opposite sides of the circle $|\varphi_{1,2}| = \frac{\pi}{2}$, due to mutual repulsion.

Another limit when the problem can be solved analytically is the uniform density of the data $\rho = 1/(2\pi)$ for arbitrary number m of hidden units. In this case the hidden units span the entire circle homogeneously - the angle between two consecutive hidden units is $\Delta\varphi = 2\pi/m$. Intuitively, similar data activate similar hidden units, resulting in similar representations.

2.3 COMPUTATIONAL COMPLEXITY AND METABOLIC COST.

In classical LSH algorithms (10; 9), typically, $k = m$ and $m \ll d$, entailing a storage cost of k bits per database entry and $O(k)$ computational cost to compute Hamming distance. In `BioHash` (and in `FlyHash`), typically $m \gg k$ and $m > d$ entailing storage cost of $k\log_2 m$ bits per database entry and $O(k)$ [1] computational cost to compute Hamming distance. Note that while there is additional storage/lookup overhead over classical LSH in maintaining pointers, this is not unlike the storage/lookup overhead incurred by quantization methods like Product Quantization (PQ) (19), which stores a lookup table of distances between every pair of codewords for each product space. From a neurobiological perspective, a highly sparse representation such as the one produced by `BioHash` keeps the same metabolic cost (20) as a dense low-dimensional ($m \ll d$) representation, such as in classical `LSH` methods. At the same time, as we empirically show below, it better preserves similarity information.

[1] If we maintain sorted pointers to the locations of 1s, we have to compute the intersection between 2 ordered lists of length k, which is $O(k)$.

Figure 2: Examples of queries and top 15 retrievals using `BioHash` ($k = 16$) on VGG16 fc7 features of CIFAR-10. Retrievals have a green (red) border if the image is in the same (different) semantic class as the query image. We show some success (top 4) and failure (bottom 2) cases. However, it can be seen that even the failure cases are reasonable.

	Hash Length (k)					
Method	2	4	8	16	32	64
`LSH` (10)	12.45	13.77	18.07	20.30	26.20	32.30
`PCAHash` (24)	19.59	23.02	29.62	26.88	24.35	21.04
`FlyHash` (3)	18.94	20.02	24.24	26.29	32.30	38.41
`SH` (25)	20.17	23.40	29.76	28.98	27.37	24.06
`ITQ` (24)	<u>21.94</u>	<u>28.45</u>	<u>38.44</u>	41.23	43.55	44.92
`DH` (26)	-	-	-	43.14	44.97	<u>46.74</u>
`UH-BNN` (27)	-	-	-	<u>45.38</u>	<u>47.21</u>	-
`BioHash`	**44.38**	**49.32**	**53.42**	**54.92**	**55.48**	-

Table 1: mAP@All (%) on MNIST (higher is better). Best results (second best) for each hash length are in **bold** (<u>underlined</u>). `BioHash` demonstrates the best retrieval performance, substantially outperforming other methods including deep hashing methods `DH` and `UH-BNN`, especially at small k. Performance for `DH` and `UH-BNN` is unavailable for some k, since it is not reported in the literature.

2.4 Convolutional BioHash

In order to take advantage of the spatial statistical structure present in images, we use the dynamics in 1 to learn convolutional filters by training on image patches as in (21). Convolutions in this case are unusual since the patches of the images are normalized to be unit vectors before calculating the inner product with the filters. Differently from (21), we use *cross channel inhibition* to suppress the activities of the hidden units that are weakly activated. Specifically, if there are F convolutional filters, then only the top k_{CI} of F activations are kept active per spatial location.

Patch normalization is reminiscent of the canonical computation of divisive normalization (22) and performs local intensity normalization. This is not unlike divisive normalization in the fruit fly's projection neurons. Divisive normalization has also been found to be beneficial (23) in CNNs trained end-to-end by the backpropogation algorithm on a supervised task. We found that the cross channel inhibition is important for a good hashing performance. Post cross-channel inhibition, we use a max-pooling layer, followed by a `BioHash` layer as in sec 2.1.

3 Similarity Search

In this section, we empirically evaluate `BioHash`, investigate the role of sparsity in the latent space, and compare our results with previously published benchmarks. We consider two settings for evaluation: a) the training set contains unlabeled data, and the labels are only used for the evaluation

	Hash Length (k)					
Method	2	4	8	16	32	64
LSH (10)	11.73	12.49	13.44	16.12	18.07	19.41
PCAHash (24)	12.73	14.31	16.20	16.81	17.19	16.67
FlyHash (3)	14.62	16.48	18.01	19.32	21.40	23.35
SH (25)	12.77	14.29	16.12	16.79	16.88	16.44
ITQ (24)	12.64	14.51	17.05	18.67	20.55	21.60
BioHash	<u>20.47</u>	<u>21.61</u>	<u>22.61</u>	<u>23.35</u>	<u>24.02</u>	-
BioConvHash	**26.94**	**27.82**	**29.34**	**29.74**	**30.10**	-

Table 2: mAP@1000 (%) on CIFAR-10 (higher is better). Best results (second best) for each hash length are in **bold** (<u>underlined</u>). BioHash demonstrates the best retrieval performance, especially at small k.

of the performance of the hashing algorithm and b) where supervised pretraining on a different dataset is permissible. Features extracted from this pretraining are then used for hashing. In both settings BioHash outperforms previously published benchmarks for various hashing methods.

3.1 Evaluation Metric

Following (28; 29; 30), we use Mean Average Precision (mAP) as the evaluation metric, a single figure that averages precision over different recall. Specifically, given a query set Q, we evaluate mAP as

$$\text{mAP}(Q) = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{1}{\sum \text{Rel}(k)} \sum_{k=1}^{R} \text{Precision}(k)\text{Rel}(k),$$

where R is number of retrievals, $\text{Rel}(k) = \mathbb{1}(\text{document } k \text{ is relevant})$, $\text{Precision}(k)$ is precision of top k retrievals.

3.2 Datasets and Protocol

To make our work comparable with recent related work, we used common benchmark datasets: a) MNIST (31), a dataset of 70k grey-scale images (size 28 x 28) of hand-written digits with 10 classes of digits ranging from "0" to "9", b) CIFAR-10 (32), a dataset containing 60k images (size 32x32x3) from 10 classes (e.g: car, bird).

Following the protocol in (26; 33), on MNIST we randomly sample 100 images from each class to form a query set of 1000 images. We use the rest of the 69k images as the training set for BioHash as well as the database for retrieval post training. Similarly, on CIFAR-10, following previous work (34; 33; 35), we randomly sampled 1000 images per class to create a query set containing 10k images. The remaining 50k images were used for both training as well as the database for retrieval as in the case of MNIST. Ground truth relevance for both dataset is based on class labels. Following previous work (33; 36; 18), we use mAP@1000 for CIFAR-10 and mAP@All for MNIST. It is common to benchmark the performance of hashing methods at hash lengths $k \in \{16, 32, 64\}$. However, it was observed (3) that the regime in which FlyHash outperformed LSH was in the regime of low hash lengths $k \in \{2, 4, 8, 16, 32\}$. Accordingly, we evaluate performance for $k \in \{2, 4, 8, 16, 32, 64\}$.

3.3 Baselines

As baselines we include random hashing methods FlyHash (3), classical LSH (LSH (10)), and data-driven hashing methods PCAHash (24), Spectral Hashing (SH (25)), Iterative Quantization (ITQ (24)). As in (3), for FlyHash we set the sampling rating from PNs to KCs to be 0.1 and $m = 10d$. Additionally, where appropriate, we also compare performance of BioHash to deep hashing methods: DeepBit (36), DH (26), USDH (18), UH−BNN (27), SAH (37) and GreedyHash (34).

3.4 Results and Discussion

The performance of BioHash on MNIST is shown in table 1. BioHash demonstrates the best retrieval performance, substantially outperforming other methods, including deep hashing methods

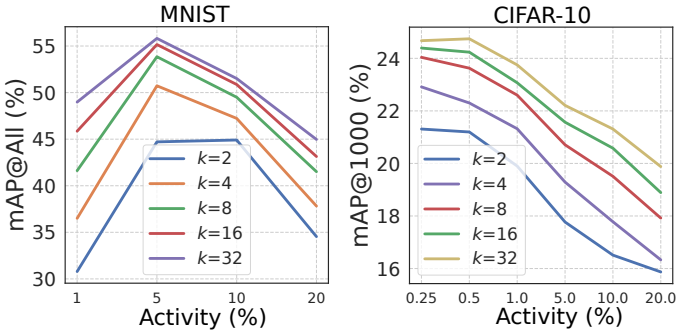

Figure 3: Effect of varying sparsity (activity): optimal activity % for MNIST and CIFAR-10 are 5% and 0.25%. Since the improvement in performance is small from 0.5 % to 0.25 %, we use 0.5% for CIFAR-10 experiments. The change of activity is accomplished by changing m at fixed k.

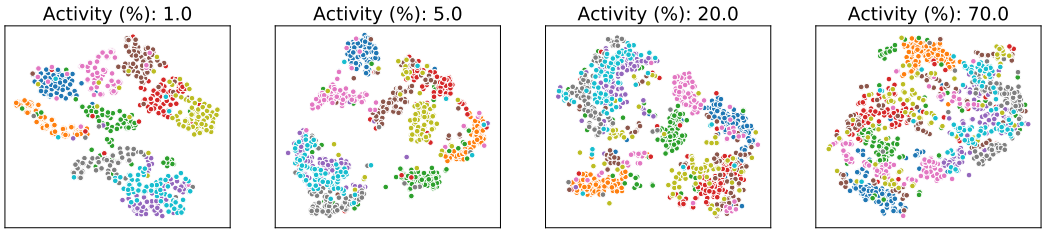

Figure 4: tSNE embedding of MNIST as the activity is varied for a fixed $m = 160$ (the change of activity is accomplished by changing k at fixed m). When the sparsity of activations decreases (activity increases), some clusters merge together, though highly dissimilar clusters (e.g. orange and blue in the lower left) stay separated.

DH and UH−BNN, especially at small k. Indeed, even at a very short hash length of $k = 2$, the performance of BioHash is comparable to or better than DH for $k \in \{16, 32\}$, while at $k = 4$, the performance of BioHash is better than the DH and UH−BNN for $k \in \{16, 32, 64\}$. The performance of BioHash saturates around $k = 16$, showing only a small improvement from $k = 8$ to $k = 16$ and an even smaller improvement from $k = 16$ to $k = 32$; accordingly, we do not evaluate performance at $k = 64$. We note that while SOLHash also evaluated retrieval performance on MNIST and is a data-driven hashing method inspired by Drosophila's olfactory circuit, the ground truth in their experiment was top 100 nearest neighbors of a query in the database, based on Euclidean distance between pairs of images in pixel space and thus cannot be directly compared[2].

The performance of BioHash on CIFAR-10 is shown in table 2. Similar to the case of MNIST, BioHash demonstrates the best retrieval performance, substantially outperforming other methods, especially at small k. Even at $k \in \{2, 4\}$, the performance of BioHash is comparable to other methods with $k \in \{16, 32, 64\}$. This suggests that BioHash is a particularly good choice when short hash lengths are required.

Effect of sparsity For a given hash length k, we parametrize the total number of neurons m in the hash layer as $m \times a = k$, where a is the activity i.e the fraction of active neurons. For each hash length k, we varied % of active neurons and evaluated the performance on a validation set (see appendix for details), see figure 3. There is an optimal level of activity for each dataset. For MNIST and CIFAR-10, a was set to 0.05 and 0.005 respectively for all experiments. We visualize tSNE embeddings for different settings of activity levels in figure 4. Interestingly, at lower sparsity levels, dissimilar images may become nearest neighbors though highly dissimilar images stay apart. This is reminiscent of an experimental finding (13) in Drosophila. Sparsification of Kenyon cells in Drosophila is controlled by feedback inhibition from the anterior paired lateral neuron. Disrupting this feedback inhibition

[2]Due to missing values of the hyperparameters we are unable to reproduce the performance of SOLHash to enable a direct comparison.

	Hash Length (k)			
k_{CI}	2	4	8	16
1	26.94	27.82	29.34	29.74
5	24.92	25.94	27.76	28.90
10	23.06	25.25	27.18	27.69
25	20.30	22.73	24.73	26.20
100	17.84	18.82	20.51	23.57

Table 3: Effect of Channel Inhibition: mAP@1000 on CIFAR-10. The number of channels that are kept active is denoted by k_{CI}. Hashing performance improves, as the number of active channels decreases.

	Hash Length (k)					
Method	2	4	8	16	32	64
LSH*	13.25	17.52	25.00	30.78	35.95	44.49
PCAHash*	21.89	31.03	36.23	37.91	36.19	35.06
FlyHash*	25.67	32.97	39.46	44.42	50.92	54.68
SH*	22.27	31.33	36.96	38.78	39.66	37.55
ITQ*	23.28	32.28	41.52	47.81	51.90	55.84
DeepBit (36)	-	-	-	19.4	24.9	27.7
USDH (18)	-	-	-	26.13	36.56	39.27
SAH (37)	-	-	-	41.75	45.56	47.36
GreedyHash (34)	10.56	23.94	34.32	44.8	47.2	50.1
BioHash*	**57.33**	**59.66**	**61.87**	**63.47**	**64.61**	-

Table 4: mAP@1000 on CIFAR-10CNN. Best results (second best) for each hash length are in **bold** (underlined). BioHash demonstrates the best retrieval performance, substantially outperforming other methods including deep hashing methods GreedyHash, SAH, DeepBit and USDH, especially at small k. Performance for DeepBit and USDH is unavailable for some k, since it is not reported in the literature. * denotes the corresponding hashing method using representations from VGG16 fc7.

leads to denser representations, resulting in fruit flies being able to discriminate between dissimilar odors but not similar odors.

Convolutional BioHash: CIFAR-10 We trained 400 convolutional filters (21) of kernel sizes $K = 3, 4$ and 10. Convolutional variant of BioHash, which we call BioConvHash shows further improvement over BioHash on CIFAR-10. Channel Inhibition is important for performance of BioConvHash, see table 3. Hashing performance improves as the number of active channels decreases.

Hashing using deep CNN features State-of-the-art hashing methods generally adapt deep CNNs trained on ImageNet (34; 18; 33; 38). These approaches derive large performance benefits from the semantic information learned in pursuit of the classification goal on ImageNet (39). To make a fair comparison with our work, we trained BioHash on features extracted from fc7 layer of VGG16 (40), since previous work (34; 36; 33) has often adapted this pre-trained network. BioHash demonstrates substantially improved performance over recent deep unsupervised hashing methods with mAP@1000 of 63.29 for $k = 16$; example retrievals are shown in figure 2. For performance of other methods and performance at varying hash lengths see table 4.

It is worth remembering that while exact Hamming distance computation is $O(k)$ for all the methods under consideration, unlike classical hashing methods, BioHash (and also FlyHash) incurs a storage cost of $k \log_2 m$ instead of k per database entry. In the case of MNIST (CIFAR-10), BioHash at $k = 2$ corresponds to $m = 40$ ($m = 400$) entailing a storage cost of 12 (18) bits respectively. Even in scenarios where storage is a limiting factor, BioHash at $k = 2$ compares favorably to other methods at $k = 16$, yet Hamming distance computation remains cheaper for BioHash.

4 CONCLUSIONS, DISCUSSION, AND FUTURE WORK

Inspired by the recurring motif of sparse expansive representations in neural circuits, we introduced a new hashing algorithm, `BioHash`. In contrast with previous work (3), `BioHash` is a data-driven algorithm that demonstrates strong empirical results outperforming recent unsupervised deep hashing methods. Moreover, `BioHash` is an online algorithm that has a reasonable degree of biological plausibility, thus supporting the proposal that LSH might be a general computational function (41) of the neural circuits featuring sparse expansive representations. From the perspective of computer science `BioHash` is easy to train and scalable to large datasets.

Compressed sensing/sparse coding have also been suggested as computational roles of sparse expansive representations in biology (42). These ideas, however, require that the input be reconstructable from the sparse latent code. This is a much stronger assumption than LSH - downstream tasks might not require such detailed information about the inputs, e.g: novelty detection (28).

In this work, we limited ourselves to linear scan using fast Hamming distance computation for image retrieval, like much of the relevant literature (3; 34; 36; 35). Yet, there is potential for improvement. One line of future inquiry would be to speed up retrieval using multi-probe methods, perhaps via psuedo-hashes (30). Another line of inquiry would be to adapt `BioHash` for Maximum Inner Product Search (43; 44).

REFERENCES

[1] Turner, G. C., Bazhenov, M. & Laurent, G. Olfactory representations by Drosophila mushroom body neurons. *Journal of neurophysiology* **99**, 734–746 (2008).

[2] Mombaerts, P. *et al.* Visualizing an Olfactory Sensory Map. *Cell* **87**, 675–686 (1996).

[3] Dasgupta, S., Stevens, C. F. & Navlakha, S. A neural algorithm for a fundamental computing problem. *Science* 5 (2017).

[4] Cover, T. M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers* 326–334 (1965).

[5] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences* **79**, 2554–2558 (1982).

[6] Sompolinsky, B. Sparseness and Expansion in Sensory Representations. *Neuron* **83**, 1213–1226 (2014).

[7] Tsodyks, M. V. & Feigel'man, M. V. The enhanced storage capacity in neural networks with low activity level. *EPL (Europhysics Letters)* **6**, 101 (1988).

[8] Wang, J., Shen, H. T., Song, J. & Ji, J. Hashing for Similarity Search: A Survey. *arXiv:1408.2927 [cs]* (2014).

[9] Indyk, P. & Motwani, R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing - STOC '98*, 604–613 (ACM Press, Dallas, Texas, United States, 1998).

[10] Charikar, M. S. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, 380–388 (ACM, New York, NY, USA, 2002).

[11] Andoni, A. & Indyk, P. Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commun. ACM* **51**, 117–122 (2008).

[12] Caron, S. J. C., Ruta, V., Abbott, L. F. & Axel, R. Random convergence of olfactory inputs in the *Drosophila* mushroom body. *Nature* **497**, 113–117 (2013).

[13] Lin, A. C., Bygrave, A. M., de Calignon, A., Lee, T. & Miesenböck, G. Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nature Neuroscience* **17**, 559–568 (2014).

[14] Stevens, C. F. A statistical property of fly odor responses is conserved across odors. *Proceedings of the National Academy of Sciences* **113**, 6737–6742 (2016).

[15] Gruntman, E. & Turner, G. C. Integration of the olfactory code across dendritic claws of single mushroom body neurons. *Nature Neuroscience* **16**, 1821–1829 (2013).

[16] Zheng, Z. *et al.* A Complete Electron Microscopy Volume of the Brain of Adult Drosophila melanogaster. *Cell* **174**, 730–743.e22 (2018).

[17] Krotov, D. & Hopfield, J. J. Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences* **116**, 7723–7731 (2019).

[18] Jin, S., Yao, H., Sun, X. & Zhou, S. Unsupervised semantic deep hashing. *Neurocomputing* **351**, 19–25 (2019).

[19] Jégou, H., Douze, M. & Schmid, C. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**, 117–128 (2011).

[20] Levy, W. B. & Baxter, R. A. Energy efficient neural codes. *Neural Computation* **8**, 531–543 (1996).

[21] Grinberg, L., Hopfield, J. & Krotov, D. Local Unsupervised Learning for Image Analysis. *arXiv:1908.08993 [cs, q-bio, stat]* (2019).

[22] Carandini, M. & Heeger, D. J. Normalization as a canonical neural computation. *Nature reviews. Neuroscience* **13**, 51–62 (2011).

[23] Ren, M., Liao, R., Urtasun, R., Sinz, F. H. & Zemel, R. S. Normalizing the Normalizers: Comparing and Extending Network Normalization Schemes. *International Conference on Learning Representations* (2016).

[24] Gong, Y. & Lazebnik, S. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR 2011*, 817–824 (IEEE, Colorado Springs, CO, USA, 2011).

[25] Weiss, Y., Torralba, A. & Fergus, R. Spectral Hashing. In Koller, D., Schuurmans, D., Bengio, Y. & Bottou, L. (eds.) *Advances in Neural Information Processing Systems 21*, 1753–1760 (Curran Associates, Inc., 2009).

[26] Lu, J., Liong, V. E. & Zhou, J. Deep hashing for scalable image search. *IEEE transactions on image processing* **26**, 2352–2367 (2017).

[27] Do, T.-T., Doan, A.-D. & Cheung, N.-M. Learning to hash with binary deep neural network. In *European Conference on Computer Vision*, 219–234 (Springer, 2016).

[28] Dasgupta, S., Sheehan, T. C., Stevens, C. F. & Navlakha, S. A neural data structure for novelty detection. *Proceedings of the National Academy of Sciences* **115**, 13093–13098 (2018).

[29] Li, W., Mao, J., Zhang, Y. & Cui, S. Fast Similarity Search via Optimal Sparse Lifting. In Bengio, S. *et al.* (eds.) *Advances in Neural Information Processing Systems 31*, 176–184 (Curran Associates, Inc., 2018).

[30] Sharma, J. & Navlakha, S. Improving Similarity Search with High-dimensional Locality-sensitive Hashing. *arXiv:1812.01844 [cs, stat]* (2018).

[31] Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**, 2278–2324 (1998).

[32] Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Tech. Rep. (2009).

[33] Chen, J., Cheung, W. K. & Wang, A. Learning Deep Unsupervised Binary Codes for Image Retrieval. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 613–619 (International Joint Conferences on Artificial Intelligence Organization, Stockholm, Sweden, 2018).

[34] Su, S., Zhang, C., Han, K. & Tian, Y. Greedy hash: Towards fast optimization for accurate hash coding in CNN. In *Advances in Neural Information Processing Systems*, 798–807 (2018).

[35] Jin, S. Unsupervised Semantic Deep Hashing. *arXiv:1803.06911 [cs]* (2018).

[36] Lin, K., Yang, H.-F., Hsiao, J.-H. & Chen, C.-S. Deep learning of binary hash codes for fast image retrieval. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 27–35 (IEEE, Boston, MA, USA, 2015).

[37] Do, T., Tan, D. L., Pham, T. T. & Cheung, N. Simultaneous Feature Aggregating and Hashing for Large-Scale Image Search. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4217–4226 (2017).

[38] Lin, J., Li, Z. & Tang, J. Discriminative Deep Hashing for Scalable Face Image Retrieval. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2266–2272 (International Joint Conferences on Artificial Intelligence Organization, Melbourne, Australia, 2017).

[39] Deng, J. *et al.* Imagenet: A large-scale hierarchical image database. In *In CVPR* (2009).

[40] Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[41] Valiant, L. G. What must a global theory of cortex explain? *Current Opinion in Neurobiology* **25**, 15–19 (2014).

[42] Ganguli, S. & Sompolinsky, H. Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis. *Annual review of neuroscience* **35**, 485–508 (2012).

[43] Shrivastava, A. & Li, P. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D. & Weinberger, K. Q. (eds.) *Advances in Neural Information Processing Systems 27*, 2321–2329 (Curran Associates, Inc., 2014).

[44] Neyshabur, B. & Srebro, N. On Symmetric and Asymmetric LSHs for Inner Product Search 9.

[45] Yang, E., Deng, C., Liu, T., Liu, W. & Tao, D. Semantic Structure-based Unsupervised Deep Hashing. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 1064–1070 (International Joint Conferences on Artificial Intelligence Organization, Stockholm, Sweden, 2018).

[46] Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]* (2015).

[47] Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L. & Weinberger, K. Q. (eds.) *Advances in Neural Information Processing Systems 25*, 1097–1105 (Curran Associates, Inc., 2012).

SUPPLEMENTARY MATERIAL

We expand on the discussion of related work in section 5. We also include here some additional results. In section 6, we show that BioConvHash enjoys robustness to intensity variations. In section 7, we show that the strong empirical performance of BioHash is not specific to the choice of VGG16. Finally, we include technical details about implementation and architecture in section 8.

5 APPENDIX A: RELATED WORK

Sparse High-Dimensional Representations in Neuroscience. Previous work has explored the nature of sparse high-dimensional representations through the lens of sparse coding and compressed sensing (42). Additionally, (6) has examined the computational role of sparse expansive representations in the context of categorization of stimuli as appetitive or aversive. They studied the case of random projections as well as learned/"structured" projections. However, structured synapses were formed by a Hebbian-like association between each cluster center and a corresponding fixed, randomly selected pattern from the cortical layer; knowledge of cluster centers provides a strong form a "supervision"/additional information, while BioHash does not assume access to such information. To the best of our knowledge no previous work has systematically examined the proposal that LSH maybe a computational principle in the brain in the context of structured synapses learned in a biologically plausible manner.

Classical LSH. A classic LSH algorithm for angular similarity is SimHash (10), which produces hash codes by $h(x) = \text{sign}(W^\intercal x)$, where entries of $W \in \mathbb{R}^{m \times d}$ are i.i.d from a standard normal distribution and sign() is element-wise. While LSH is a property and consequently is sometimes used to refer to hashing methods in general, when the context is clear we refer to SimHash as LSH following previous literature.

Fruit Fly inspired LSH. The fruit fly's olfactory circuit has inspired research into new families (28; 30; 29) of Locality Sensitive Hashing (LSH) algorithms. Of these, FlyHash (3) and DenseFly (30) are based on random projections and cannot *learn* from data. Sparse Optimal Lifting (SOLHash) (29) is based on learned projections and results in improvements in hashing performance. SOLHash attempts to learn a sparse binary representation $Y \in \mathbb{R}^{n \times m}$, by optimizing

$$\underset{\substack{Y \in [-1,1]^{n \times m} \\ Y\mathbf{e}_d = (-m+2k)\mathbf{e}_m}}{\arg\min} \quad ||XX^\intercal - YY^\intercal||_F^2 + \gamma ||Y||_p, \tag{12}$$

$\mathbf{e}_m$ is an all 1's vector of size m. Note the relaxation from a binary $Y \in \{-1, 1\}^{n \times m}$ to continuous $Y \in [-1, 1]^{n \times m}$. After obtaining a Y, queries are hashed by learning a linear map from X to Y by minimizing

$$\underset{\substack{W \in [-1,1]^{d \times m} \\ W\mathbf{e}_d = (-m+2c)\mathbf{e}_m}}{\arg\min} \quad ||Y - XW||_F^2 + \beta ||W||_p, \tag{13}$$

Here, c is the # of synapses with weight 1; the rest are -1. To optimize this objective, (29) resorts to Franke-Wolfe optimization, wherein every learning update involves solving a constrained linear program involving all of the training data, which is biologically unrealistic. In contrast, BioHash is neurobiologically plausible involving only Hebbian/Anti-Hebbian updates and inhibition.

From a computer science perspective, the scalability of SOLHash is highly limited; not only does every update step invoke a constrained linear program but the program involves pairwise similarity matrices, which can become intractably large for datasets of even modest size. This issue is further exacerbated by the fact that $m \gg d$ and $YY^\intercal$ is recomputed at every step (29). Indeed, though (29) uses the SIFT1M dataset, the discussed limitations limit training to only 5% of the training data.

Deep LSH. A number of state-of-the-art approaches (34; 35; 37; 36) to unsupervised hashing for image retrieval are perhaps unsurprisingly, based on deep CNNs trained on ImageNet (39); A common approach (34) is to adopt a pretrained DCNN as a backbone, replace the last layer with a custom hash layer and objective function and to train the network by backpropogation. Some other approaches (45), use DCNNs as feature extractors or to compute a measure of similarity in it's feature space, which is then used as a training signal. While deep hashing methods are not the purpose of our work, we include them here for completeness.

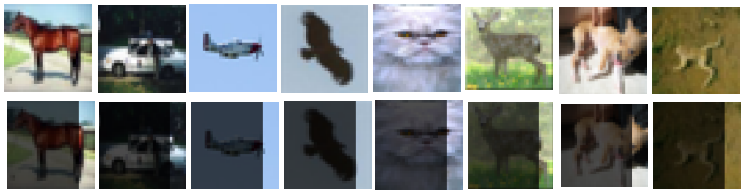

Figure 5: Examples of images with and without a "shadow". We modified the intensities in the query set of CIFAR-10 by multiplying 80% of each image by a factor of 0.3; such images largely remain discriminable to human perception.

6 Appendix B: Robustness of BioConvHash to variations in intensity

Patch normalization is reminiscent of the canonical neural computation of divisive normalization (22) and performs local intensity normalization. This makes `BioConvHash` robust to variations in light intensity. To test this idea, we modified the intensities in the query set of CIFAR-10 by multiplying 80% of each image by a factor of 0.3; such images largely remain discriminable to human perception, see figure 5. We evaluated the retrieval performance of this query set with "shadows", while the database (and synapses) remain unmodified. We find that `BioConvHash` performs best at small hash lengths, while the performance of other methods except `GreedyHash` is almost at chance. These results suggest that it maybe beneficial to incorporate divisive normalization into DCNNs architectures to increase robustness to intensity variations.

	Hash Length (k)					
Method	2	4	8	16	32	64
`LSH`	10.62	11.82	11.71	11.25	11.32	11.90
`PCAHash`	10.61	10.60	10.88	11.33	11.79	11.83
`FlyHash`	11.44	11.09	11.86	11.89	11.45	11.44
`SH`	10.64	10.45	10.45	11.70	11.26	11.30
`ITQ`	10.54	10.68	11.65	11.00	10.95	10.94
`BioHash`	11.05	11.50	11.57	11.33	11.59	-
`BioConvHash`	**26.84**	**27.60**	**29.31**	29.57	29.95	-
`GreedyHash`	10.56	21.47	25.21	**30.74**	**30.16**	**37.63**

Table 5: mAP@1000 (%) on CIFAR-10 (higher is better), when query set has "shadows". Performance of other hashing methods drops substantially, while the performance of `BioConvHash` remains largely unchanged due to patch normalization. For small k, `BioConvHash` performs best. Best results (second best) for each hash length are in **bold** (underlined).

7 Appendix C: Evaluation using VGG16BN and AlexNet

The strong empirical performance of `BioHash` using features extracted from VGG16 fc7 is not specific to choice of VGG16. To demonstrate this, we evaluated the performance of `BioHash` using VGG16 with batch normalization (BN) (46) and AlexNet (47). Consistent with the evaluation using VGG16 reported in the main paper, `BioHash` consistently demonstrates the best retrieval performance, especially at small k.

8 Appendix D: Implementation details

- `BioHash`: The training /retrieval database was centered. Queries were also centered using mean computed on the training set. Weights were initialized by sampling from the standard normal distribution. For simplicity we used $p = 2, \Delta = 0$. We set initial learning rate $\epsilon_0 = 2 \times 10^{-2}$, which was decayed as $\epsilon_t = \epsilon_0(1 - \frac{t}{T_{\text{max}}})$, where t is epoch number and T_{max} is maximum number of epochs. We used $T_{\text{max}} = 100$ and a mini-batch size of 100. The criterion for convergence was average norm of synapses was < 1.06. Convergence usually took < 20 epochs.

	Hash Length (k)					
Method	2	4	8	16	32	64
LSH	13.16	15.86	20.85	27.59	38.26	47.97
PCAHash	21.72	34.05	38.64	40.81	38.75	36.87
FlyHash	27.07	34.68	39.94	46.17	52.65	57.26
SH	21.76	34.19	38.85	41.80	42.44	39.69
ITQ	23.02	34.04	44.57	51.23	55.51	58.74
BioHash	**60.56**	**62.76**	**65.08**	**66.75**	**67.53**	-

Table 6: mAP@1000 on CIFAR-10CNN, VGG16BN. Best results (second best) for each hash length are in **bold** (underlined). BioHash demonstrates the best retrieval performance, especially at small k.

	Hash Length (k)					
Method	2	4	8	16	32	64
LSH	13.25	12.94	18.06	23.28	25.79	32.99
PCAHash	17.19	22.89	27.76	29.21	28.22	26.73
FlyHash	18.52	23.48	27.70	30.58	35.54	38.41
SH	16.66	22.28	27.72	28.60	29.27	27.50
ITQ	17.56	23.94	31.30	36.25	39.34	42.56
BioHash	**44.17**	**45.98**	**47.66**	**49.32.43**	**50.13**	-

Table 7: mAP@1000 on CIFAR-10CNN, AlexNet. Best results (second best) for each hash length are in **bold** (underlined). BioHash demonstrates the best retrieval performance, especially at small k.

In order to set the activity level, we performed cross-validation. In the case of MNIST, we separated 1k random samples (100 from each class) from the training set, to create a training set of size 68k and validation set of 1k images. Activity level with highest mAP@All on the validation set was determined to be 5%, see figure 3. We then retrained BioHash on the whole training data of size 69k and reported the performance on the query set. Similarly for CIFAR-10, we separated 1k samples (100 images per class) to create a training set of size 49k and validation set of 1k. We set the activity level to be 0.5%, see figure 3. We then retrained BioHash on the whole training data of size 50k and reported the performance on the query set.

- BioConvHash We trained convolutional filters of kernel sizes $K = 3, 4, 10$ on CIFAR-10 training set of 50k images, stride of 1. The convolutional filters were trained by dividing the training set into patches of sizes $K = 3 \times 3, 4 \times 4, 10 \times 10$ and applying the training dynamics 1. We set $k_{\text{CI}} = 1$, by cross-validation in a procedure similar to how sparsity was set. The effect of channel inhibition is shown in table 3. $k_{\text{CI}} = 1$ means that only the largest activation across channels per spatial location was kept, while the rest are set to 0. We employed separate 2d max-pooling layers for each set of filters (corresponding to $K = 3, 4, 10$) with a common stride of 2. This was followed by a fully connected layer(the "hash" layer).
- FlyHash Following (3), we set $m = 10d$ for all hash lengths k and each neuron in the hashing layer ("Kenyon" cell) sampled from 0.1 dimensions of input data (Projection neurons). Following (24), ITQ employed 50 iterations.
- To extract representations from VGG16 fc7, CIFAR-10 images were resized to 224×224 and normalized using default values: $[0.485, 0.456, 0.406], [0.229, 0.224, 0.225]$. To make a fair comparison we used the pre-trained VGG16 model (without BN), since this model is frequently employed by deep hashing methods. We also evaluated the performance using a VGG16 with BN and also on AlexNet (47), see tables 6, 7.
- GreedyHash, replaces the softmax layer of VGG16 with a hash layer and is trained end-to-end in via backpropogation using a custom objective function, see (34) for more details. We that we use the code provided by the authors to measure performance at $k = 2, 4, 8$. We used the default parameters: mini-batch size of 32, learning rate of 1×10^{-4} and trained for 60 epochs.