Supplementary Information

We expand on the discussion of related work in section 5. We also include here some additional results. Specifically, evaluation on GloVe dataset is included in section 6, demonstrating strong performance of `BioHash`. In section 7, it is shown that `BioConvHash` enjoys robustness to intensity variations. In section 8, we show that the strong performance of `BioHash` is not specific to the particular choice of the architecture (in the main paper only VGG16 was used). In section 9, we include technical details about implementation and architecture. In section 10, the KL divergence is calculated between the distribution of the data and the induced distribution of the hash codes for the small-dimensional models discussed in section 2.2. Finally, training time is discussed in section 11.

5. Additional Discussion of 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 (Ganguli & Sompolinsky, 2012). Additionally, (Sompolinsky, 2014) 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` (Charikar, 2002), 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 (Dasgupta et al., 2018; Sharma & Navlakha, 2018; Li et al., 2018) of Locality Sensitive Hashing (LSH) algorithms. Of these, `FlyHash` (Dasgupta et al., 2017) and `DenseFly` (Sharma & Navlakha, 2018) are based on random projections and cannot *learn* from data. Sparse Optimal Lifting (`SOLHash`) (Li et al., 2018) 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

$$\underset{\substack{Y\in[-1,1]^{n\times m}\\ Y\mathbf{e}_d=(-m+2k)\mathbf{e}_m}}{\arg\min} \quad ||XX^\intercal - YY^\intercal||_F^2 + \gamma||Y||_p, \tag{14}$$

$\mathbf{e}_m$ 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

$$\underset{\substack{W\in[-1,1]^{d\times m}\\ W\mathbf{e}_d=(-m+2c)\mathbf{e}_m}}{\arg\min} \quad ||Y - XW||_F^2 + \beta||W||_p, \tag{15}$$

Here, c is the # of synapses with weight 1; the rest are -1. To optimize this objective, (Li et al., 2018) 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 (Li et al., 2018). Indeed, though (Li et al., 2018) uses the SIFT1M dataset, the discussed limitations limit training to only 5% of the training data. Nevertheless, we make a comparison to `SOLHash` in Table 6 and see that `BioHash` results in substantially improved performance.

In the present work, we took the biological plausibility as a primary since one of the goals of our work was to better understand the computational role of sparse expansive biological circuits. Yet from a practical perspective, our work suggests that this constraint of biological plausibility may be relaxed while keeping or even improving the performance benefits - potentially by explicitly training a hashing method end-to-end using kWTA in lieu of using it post-hoc or by relaxing the online learning constraint.

Other WTA approaches Previous hashing approaches (Yagnik et al., 2011; Chen & Shrivastava, 2018) have used WTA (like `BioHash` and `FlyHash`) but do not use dimensionality expansion and do not learn to adapt to the data manifold.

Deep LSH. A number of state-of-the-art approaches (Su et al., 2018; Jin, 2018; Do et al., 2017a; Lin et al., 2015) to unsupervised hashing for image retrieval are perhaps unsurprisingly, based on deep CNNs trained on ImageNet (Deng et al., 2009); A common approach (Su et al., 2018)

Table 6. mAP@100 (%) on MNIST, using Euclidean distance in pixel space as the ground truth, following protocol in (Li et al., 2018). BioHash demonstrates the best retrieval performance, substantially outperforming SOLHash.

	Hash Length (k)					
Method	2	4	8	16	32	64
BioHash	**39.57**	**54.40**	**65.53**	**73.07**	**77.70**	**80.75**
SOLHash	11.59	20.03	30.44	41.50	51.30	-

Table 7. mAP@100 (%) on GloVe ($d = 300$), ground truth based on *Euclidean distance*. 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	0.37	0.51	1.93	12.91	18.23	26.83
PCAHash	0.74	1.45	4.86	19.57	28.52	37.49
FlyHash	13.95	15.78	21.15	28.12	39.72	54.24
SH	0.81	1.31	4.81	19.16	27.44	35.65
ITQ	0.59	1.42	4.57	19.81	31.50	43.08
BioHash	**23.06**	**34.42**	**43.21**	**50.32**	**56.94**	**62.87**

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 (Yang et al., 2018), 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.

Discrete locality sensitive hash codes have also been used for modelling dialogues in (Garg et al., 2018).

6. Evaluation on GloVe

We include evaluation on GloVe embeddings (Pennington et al., 2014). We use the top 50,000 most frequent words. As in previous work (Dasgupta et al., 2017), we selected a random subset of 10,000 words as the database and each word in turn was used as a query; ground truth was based on nearest neighbors in the database. Methods that are trainable (e.g.BioHash, ITQ), are trained on the remaining 40,000 words. Results shown are averages over 10 random partitions; Activity $a = 0.01$. Results are shown for Euclidean distance in Table 7 and cosine distance in Table 8.

7. Robustness of BioConvHash to variations in intensity

Patch normalization is reminiscent of the canonical neural computation of divisive normalization (Carandini & Heeger, 2011) 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 6. 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.

Table 8. mAP@100 (%) on GloVe ($d = 300$), ground truth based on *cosine distance*. 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	0.41	0.65	2.23	13.91	30.30	32.60
PCAHash	0.65	1.71	7.18	25.87	40.07	53.13
FlyHash	15.06	17.09	24.64	34.12	50.96	72.37
SH	0.79	1.74	7.01	25.39	37.68	49.39
ITQ	0.76	1.84	6.84	27.64	44.47	61.15
BioHash	**38.13**	**54.22**	**66.85**	**76.30**	**84.05**	**89.78**

Table 9. Robustness to shadows. 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 substantially outperforms all the other methods, while still being competitive at higher hash lengths. Best results (second best) for each hash length are in **bold** (underlined).

	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 10. 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	13.16	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**	-

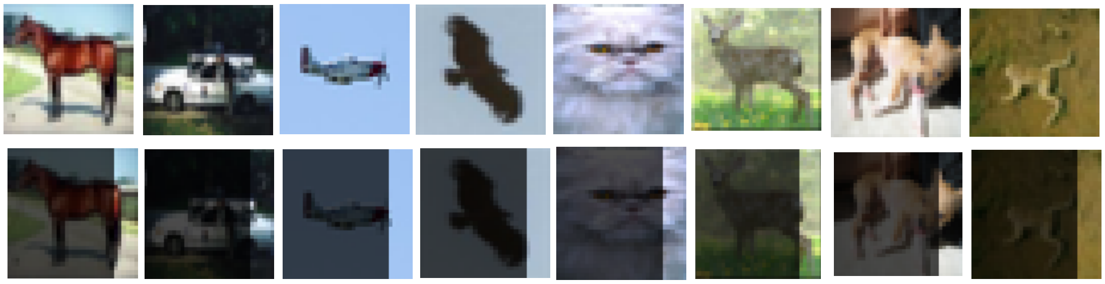

Figure 6. Examples of images with and without a "shadow". We modified the intensities in the query set of CIFAR-10 by multiplying 80% of each image by a factor of 0.3; such images largely remain discriminable to human perception.

Table 11. 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.

	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**	**50.13**	-

8. 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) (Ioffe & Szegedy, 2015) as well as AlexNet (Krizhevsky et al., 2012). Consistent with the evaluation using VGG16 reported in the main paper, BioHash consistently demonstrates the best retrieval performance, especially at small k.

9. 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.

  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 4 (main text). 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 4 (main text). We then retrained BioHash on the whole training data of size 50k and reported the performance on the query set.

- BioConvHash A convolutional filter of kernel size K is learned by dividing the training set into patches of sizes $K \times K$ and applying the learning dynamics. In the case of MNIST, we trained 500 filters of kernel sizes $K = 3, 4$. The filters were trained with $p = 2$, $r = 2$, $\Delta = 0.1$; $\epsilon_0 = 10^{-3}$. In the case of CIFAR-10, we trained 400 filters of kernel sizes $K = 3, 4, 10$ (corresponding $\Delta = 0.1, 0.2, 0.2$; for all filters $p = 2, r = 2$; $\epsilon_0 = 10^{-4}$). For both datasets, we used a stride of 1 in the convolutional layers. We set $k_{\text{CI}} = 10$ for MNIST and $k_{\text{CI}} = 1$ for CIFAR-10 during hashing. Hyperparameters were set cross-validation. The effect of channel inhibition is shown in Table 3 (main text) for the query set. $k_{\text{CI}} = 1$ means that only the largest activation across channels per spatial location was kept, while the rest are set to 0. This was followed by $2d$ max-pooling with a stride of 2 and kernel size of 7. This was followed by a fully connected layer (the "hash" layer).

- FlyHash Following (Dasgupta et al., 2017), 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 (Gong & Lazebnik, 2011), 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 VGG16 with BN and also using AlexNet (Krizhevsky et al., 2012), see Tables 10, 11.

- `GreedyHash` replaces the softmax layer of VGG16 with a hash layer and is trained end-to-end via backpropogation using a custom objective function, see (Su et al., 2018) for more details. We use the code [5] provided by the authors to measure performance at $k = 2, 4, 8$, since these numbers were not reported in (Su et al., 2018). We used the default parameters: mini-batch size of 32, learning rate of 1×10^{-4} and trained for 60 epochs.

10. Distribution of the data in the hash space

A distribution of the data in the input space induces a distribution over all possible hash codes. In this section the analysis of the small dimensional toy model examples from section (2.2) is expanded to compare the properties of these two distributions. Specifically, consider a data distribution $\rho(\varphi)$ described by equation (9), and assume that only $m = 3$ hidden units are available. Similarly to the case of $m = 2$, considered in the main text, an explicit expression for the energy function can be derived and the three angles corresponding to the positions of the hidden units can be calculated (see Figure 7). The angle ψ is determined as a solution to the following equation

$$\big(\sigma\cos\psi - \sin\psi\big)e^{-\frac{\pi}{\sigma}} + \big(\sigma\cos\frac{\psi}{2} - \sin\frac{\psi}{2}\big)e^{-\frac{\psi}{2\sigma}} = 0. \quad (16)$$

It can be easily solved in the limiting cases: $\sigma \to 0$ with $\psi \to 2\sigma$, and $\sigma \to \infty$ with $\psi = \frac{2\pi}{3}$. Notice an extra factor of 2 in the former case compared with $\psi = |\varphi_{1,2}| \approx \sigma$ in the case of $m = 2$ (see the main text). This extra factor of 2 reflects an additional force of repulsion from the middle hidden unit exerted onto the flanking hidden units. As a result of this additional force the flanking hidden units are positioned (twice) further away from the mean of the data distribution than in the case of $m = 2$, which does not have a hidden unit in the middle.

For $m = 3$, two possible choices of the hash lengths can be made: a) $k = 1$, for every data point a nearest hidden unit is activated, and b) $k = 2$, two nearest hidden units are activated. The corresponding distributions over the hash codes are denoted as $P_{k=1}$ and $P_{k=2}$. It is possible to calculate a KL divergence between the original distribution and the induced distributions in the hash space. For $k = 1$, we obtained:

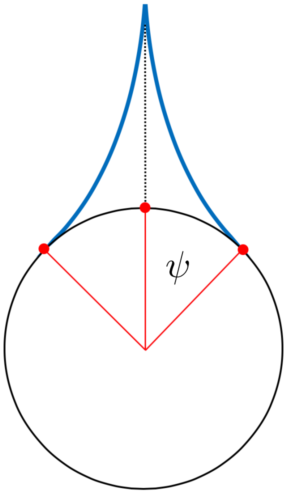

Figure 7. Positions of the $m = 3$ hidden units (shown in red) relative to the density of the data described by (9) (shown in blue). The angle between the middle hidden unit and one of the flanking hidden units is denoted by ψ.

$$D(\rho||P_{\text{k=1}}) = -\frac{\sigma - (\pi+\sigma)e^{-\pi/\sigma}}{\sigma(1-e^{-\pi/\sigma})} - \frac{1-e^{-\psi/2\sigma}}{1-e^{-\pi/\sigma}}\ln\Big[\frac{2\sigma(1-e^{-\psi/2\sigma})}{\psi}\Big] - \frac{e^{-\psi/2\sigma}-e^{-\pi/\sigma}}{1-e^{-\pi/\sigma}}\ln\Big[\frac{\sigma\big(e^{-\psi/2\sigma}-e^{-\pi/\sigma}\big)}{\pi-\psi/2}\Big]. \quad (17)$$

For $k = 2$, the following expression holds:

$$D(\rho||P_{\text{k=2}}) = -\frac{\sigma - (\pi+\sigma)e^{-\pi/\sigma}}{\sigma(1-e^{-\pi/\sigma})} - \frac{e^{-\pi/\sigma}\big(-1+e^{\psi/2\sigma}\big)}{1-e^{-\pi/\sigma}}\ln\Big[\frac{2\sigma e^{-\pi/\sigma}\big(-1+e^{\psi/2\sigma}\big)}{\psi}\Big] - \frac{1-e^{\psi/2\sigma-\pi/\sigma}}{1-e^{-\pi/\sigma}}\ln\Big[\frac{\sigma\big(1-e^{\psi/2\sigma-\pi/\sigma}\big)}{\pi-\psi/2}\Big]. \quad (18)$$

For sufficiently smooth distributions of the data $\sigma \to \infty$, both divergences approach zero. Thus, in this limiting case the original and the induced distributions match exactly. For finite values of σ the divergence of the original and induced distributions is quantified by the expressions above.

As with almost any representation learning algorithm (e.g. deep neural nets) it is difficult to provide theoretical guarantees in generality. It is possible, however, to calculate the probability of false negatives (probability that similar data points are assigned different hash codes) for our hashing algorithm analytically on the circle in the limit $\sigma \to \infty$. Assuming hash length $k = 1$ and a given cosine similarity

[5] https://github.com/ssppp/GreedyHash

Table 12. Training time for the best variant of BioHash, and the next best method for MNIST.

	Hash Length (k)				
Method	2	4	8	16	32
BioHash	∼1.7 *s*	∼1.7*s*	∼1.7 *s*	∼3.4 *s*	∼ 5 *s*
BioConvHash	∼3.5 *m*	∼3.5 *m*	∼3.5 *m*	∼5 *m*	∼5 *m*

Table 13. Training time for the best variant of BioHash and the next best method for CIFAR-10. Both models are based on VGG16.

	Hash Length (k)				
CIFAR-10	2	4	8	16	32
BioHash	∼ 4.2 *s*	∼7.6 *s*	∼11.5 *s*	∼22 *s*	∼35 *s*
GreedyHash	∼1.2 *hrs*	∼1.2 *hrs*	∼1.3 *hrs*	∼1.4 *hrs*	∼1.45 *hrs*

between two data points $\theta = \arccos(x, y)$, the probability that they have different hash codes is equal to

$$P = \begin{cases} \frac{m\theta}{2\pi}, & \text{for } \theta \geq \frac{2\pi}{m} \\ 1, & \text{for } \theta > \frac{2\pi}{m}. \end{cases}$$

11. Training time

Here we report the training times for the best performing (having the highest corresponding mAP@R) variant of our algorithm: BioHash, BioConvHash, or BioHash on top of VGG16 representations. For the case of MNIST, the best performing variant is BioConvHash, and for CIFAR-10 it is BioHash on top of VGG16 representations. We also report the training time of the next best method for each dataset. This is GreedyHash in the case of CIFAR-10, and BioHash in the case of MNIST. In the case of MNIST, the best method that is not a variant of BioHash is UH-BNN. Training time for UH-BNN is unavailable, since it is not reported in literature. All experiments were run on a single V100 GPU to make a fair comparison.