Estimating Transfer Entropy under Long Ranged Dependencies (Supplementary material)

Sahil Garg[1,*] **Umang Gupta**[2] **Yu Chen**[1,†] **Syamantak Datta Gupta**[1,†] **Yeshaya Adler**[1,†] **Anderson Schneider**[1] **Yuriy Nevmyvaka**[1]

[1]Department of Machine Learning Research, Morgan Stanley, New York, NY, USA,
[2]Department of Computer Science, University of Southern California, Los Angeles, CA, USA
[*]Corresponding Author: sahil.garg.cs@gmail.com, sahil.garg@morganstanley.com
[†]Equal contributions

A THEORETICAL PROOFS

A.1 CONSISTENCY OF REGULARIZATION WITH HASH CODES

Below, we present a proof of Thm 1, we can see LSH based data generation as sampling from a data-driven histogram. Using this insight and using results from Lugosi and Nobel [1996] and a proof technique similar to Rothfuss et al. [2019], we will demonstrate consistency of our sampling approach. We use this approach to estimate entropy, i.e., $\mathcal{H}(\boldsymbol{\mathcal{Y}}_t|\boldsymbol{\mathcal{Y}_{t-1}})$.

Theorem 1. *Suppose* $\lim_{n\to\infty} \frac{2^H}{n} \to 0$, $\lim_{n\to\infty} \frac{tH\log n}{n} \to 0$ *and the input space, i.e.,* $\boldsymbol{y} \in \mathbb{R}^t$ *is bounded. Consider any function,* $f : \boldsymbol{y} \to (0, \infty)$ *with* $\log f$ *having finite second order moment w.r.t to* p *and* $g_{n,H}$*, then*

$$\lim_{n\to\infty} |\mathbb{E}_p[-\log f(\boldsymbol{y})] - \mathbb{E}_g[-\log f(\boldsymbol{y})]| \to 0$$

Proof. Let $l(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{y}\sim p} \log f(\boldsymbol{y})$ and $l_n^{(g,H)}(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{y}\sim g_{n,H}} \log f(\boldsymbol{y})$. Here $g_{n,H}(\boldsymbol{y})$ denotes the perturbed distribution obtained by using H dimensional hashcode function and n samples. Consider,

$$\begin{aligned}
|l(\boldsymbol{\theta}) - l_n^{(g,H)}(\boldsymbol{\theta})|^2 &= \left|-\mathbb{E}_{\boldsymbol{y}\sim p} \log f(\boldsymbol{y}) + \mathbb{E}_{\boldsymbol{y}\sim g_{n,H}} \log f(\boldsymbol{y})\right|^2 \\
&= \left(\mathbb{E}_{\boldsymbol{y}\sim p} \log f(\boldsymbol{y}) - \mathbb{E}_{\boldsymbol{y}\sim g_{n,H}} \log f(\boldsymbol{y})\right)^2 \\
&= \left(\int d\boldsymbol{y} \log f(\boldsymbol{y}) \left(p(\boldsymbol{y}) - g_{n,H}(\boldsymbol{y})\right)\right)^2 \\
&= \left(\int d\boldsymbol{y} \left(\log f(\boldsymbol{y}) \left(p(\boldsymbol{y}) - g_{n,H}(\boldsymbol{y})\right)^{0.5}\right) \left(p(\boldsymbol{y}) - g_{n,H}(\boldsymbol{y})\right)^{0.5}\right)^2
\end{aligned}$$

Using Cauchy-Schwartz inequality, $\left(\int d\boldsymbol{y} a(\boldsymbol{y}) b(\boldsymbol{y})\right)^2 \leq = \int d\boldsymbol{y} |a(\boldsymbol{y})|^2 \int d\boldsymbol{y} |b(\boldsymbol{y})|^2$

$$\leq \left(\int d\boldsymbol{y} \left|\log f(\boldsymbol{y})\right|^2 \left|p(\boldsymbol{y}) - g_{n,H}(\boldsymbol{y})\right|\right) \left(\int d\boldsymbol{y} \left|p(\boldsymbol{y}) - g_{n,H}(\boldsymbol{y})\right|\right)$$

The first term will be bounded by $\mathbb{E}_p \log^2 f + \mathbb{E}_g \log^2 f$ which is some finite quantity by assumption. To bound the second term, we use Thm. 1 of Lugosi and Nobel [1996], which is restated below.

Theorem 2 (Theorem 1 of Lugosi and Nobel [1996])**.** *Let* $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots$ *be IID random vectors in* $\mathbb{R}^t$ *and* $\boldsymbol{y}^{(i)} \sim p(\boldsymbol{y})$*. Let* $\Pi = \{\pi_1, \pi_2 \ldots\}$ *be a fixed partitioning scheme for* $\mathbb{R}^t$ *and let* $\mathcal{A}_n$ *be the collection of all possible partitions associated with the rule* π_n*, and* $\pi_n[\boldsymbol{y}]$ *denotes the partition in which* $\boldsymbol{y}$ *lies. Define* $m(\mathcal{A}) = \max_{\pi\in\mathcal{A}} |\pi|$ *as the maximum size of partition,* $\Delta^*(\mathcal{A}, n)$ *is the maximum number of distinct partitions of any* n *points in* $\mathbb{R}^t$ *induced by* $\mathcal{A}$*. If as* $n \to \infty$*,*

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

- $n^{-1}m(\mathcal{A}_n) \to 0$
- $n^{-1}\log \Delta^*(\mathcal{A}_n, n) \to 0$
- $P(\{\boldsymbol{y} : diam(\pi_n[\boldsymbol{y}]) > \gamma\}) \to 0$*, i.e., the size of each bin with significant probability mass goes to zero.*

Then the histogram density estimates, $g_{n,H}$ are strongly consistent in L_1, i.e., with probability 1

$$\int |p(\boldsymbol{y}) - g_{n,H}(\boldsymbol{y})| d\boldsymbol{y} \to 0$$

Since, we used a H dimensional binary hashcode in this work, $m(\mathcal{A}_n) \leq 2^H$. Now each bit of hashcode function can be seen as partitioning the input space (or transformed input space) by hyperplanes. We know that n points can be split by hyperplanes in n^{t^*} ways [Cover, 1965], where t^* is the effective-input dimension. Assuming each hash code function is linear, there are only n^t possible ways in N points can be split. Thus, $\Delta^*(\mathcal{A}_n, n) \leq (n^t)^H$. We can see that the first two conditions are satisfied due to the regularity conditions.

Since input space is bounded and H can be increased as n increases, the diameter of significant probability bins would shrink and tend to 0 eventually. Hence,

$$\int |p(\boldsymbol{y}) - g_{n,H}(\boldsymbol{y})| d\boldsymbol{y} \to 0$$

. □

A.2 VARIANCE OF THE ESTIMATOR

Theorem 3. *For some data distribution p and conditional model distribution q and $-\log q(\boldsymbol{y}_t|(.)) \in [-Q, Q]$. Let $\hat{T}^q_{X\to Y}$ denote the n-sample estimate of transfer entropy. Then with probability $1-\delta$ ($\delta > 0$), we have*

$$\left|\hat{\mathcal{T}}^q_{X\to Y} - \mathcal{T}^q_{X\to Y}\right| \leq 2Q\sqrt{\frac{2}{n}\ln\frac{4}{\delta}} \tag{1}$$

Proof. To estimate transfer entropy, we use difference of empirical estimate of two conditional entropy terms, i.e,

$$\mathcal{T}^q_{X\to Y} = \mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}) - \mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}, \boldsymbol{\mathcal{X}}_{t-1})$$

Now, we can write the error as:

$$\begin{aligned}\left|\hat{\mathcal{T}}^q_{X\to Y} - \mathcal{T}^q_{X\to Y}\right| &= \left|\left(\mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}) - \hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1})\right) - \left(\mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}, \boldsymbol{\mathcal{X}}_{t-1}) - \hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}, \boldsymbol{\mathcal{X}}_{t-1})\right)\right| \\ &\leq \left|\mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}) - \hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1})\right| + \left|\mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}, \boldsymbol{\mathcal{X}}_{t-1}) - \hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}, \boldsymbol{\mathcal{X}}_{t-1})\right|\end{aligned}$$

We will bound the probability of error in transfer entropy estimates by bounding the error of both terms in both R.H.S terms of above expression using additive-chernoff bounds (similar to McAllester and Stratos [2020]). We can write,

$$\begin{aligned}&P\left(\left|\hat{\mathcal{T}}^q_{X\to Y} - \mathcal{T}^q_{X\to Y}\right| \leq \epsilon\right) \\ &\geq \sum_{e=0}^{\epsilon} P\left(\left|\mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}) - \hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1})\right| \leq e\right) P\left(\left|\mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}, \boldsymbol{\mathcal{X}}_{t-1}) - \hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}, \boldsymbol{\mathcal{X}}_{t-1})\right| \leq \epsilon - e\right) \\ &\geq P\left(\left|\mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}) - \hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1})\right| \leq \epsilon/2\right) P\left(\left|\mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}, \boldsymbol{\mathcal{X}}_{t-1}) - \hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}, \boldsymbol{\mathcal{X}}_{t-1})\right| \leq \epsilon/2\right)\end{aligned} \tag{2}$$

Consider, $\hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}) = \frac{1}{n}\sum_{i=1}^n \log q(y_t^{(i)}|\boldsymbol{y}_{t-1}^{(i)})$, and, $\mathbb{E}\ \hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}) = \mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1})$. Since $\log q \in [-Q, Q]$, using additive chernoff bounds, we have

$$P\left(\left|\mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1}) - \hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1})\right| \geq \epsilon/2\right) \leq 2e^{-\frac{2n(\epsilon/2)^2}{(2Q)^2}} = 2e^{-\frac{n\epsilon^2}{8Q^2}}$$

and similarly,

$$P\left(\left|\mathcal{H}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1},\boldsymbol{\mathcal{X}}_{t-1})-\hat{\mathcal{H}}(\mathcal{Y}_t|\boldsymbol{\mathcal{Y}}_{t-1},\boldsymbol{\mathcal{X}}_{t-1})\right|\geq\epsilon/2\right)\leq 2e^{-\frac{n\epsilon^2}{8Q^2}}$$

Substituting in 2, we have

$$P\left(\left|\hat{\mathcal{T}}^q_{X\to Y}-\mathcal{T}^q_{X\to Y}\right|\leq\epsilon\right)\leq\left(1-2e^{-\frac{n\epsilon^2}{8Q^2}^2}\right)^2$$

or,

$$P\left(\left|\hat{\mathcal{T}}^q_{X\to Y}-\mathcal{T}^q_{X\to Y}\right|\geq\epsilon\right)\leq 4e^{-\frac{n\epsilon^2}{8Q^2}}$$

Setting, $4e^{-\frac{n\epsilon^2}{8Q^2}}=\delta$, we get $\epsilon=2Q\sqrt{\frac{2}{n}\ln\frac{4}{\delta}}$ □

Remark: Using above results, it becomes straight-forward to compute the bounds on mean square error of the estimator or variance as below:

$$\begin{aligned}\mathbb{E}\left(\hat{\mathcal{T}}^q_{X\to Y}-\mathcal{T}^q_{X\to Y}\right)^2 &\leq \left(2Q\sqrt{\frac{2}{n}\ln\frac{4}{\delta}}\right)^2 P\left(\left|\hat{\mathcal{T}}^q_{X\to Y}-\mathcal{T}^q_{X\to Y}\right|\leq 2Q\sqrt{\frac{2}{n}\ln\frac{4}{\delta}}\right)+\\ &\quad P\left(\left|\hat{\mathcal{T}}^q_{X\to Y}-\mathcal{T}^q_{X\to Y}\right|\geq 2Q\sqrt{\frac{2}{n}\ln\frac{4}{\delta}}\right)(Q-(-Q))^2\\ &=(1-\delta)\left(\frac{8Q^2}{n}\ln\frac{4}{\delta}\right)+4Q^2\delta\\ &=4Q^2\left((1-\delta)\frac{2}{n}\ln\frac{4}{\delta}+\delta\right)\end{aligned}$$

B MORE ALGORITHMIC DETAILS

In this section, we introduce more details about unsupervised learning of hash functions. The main idea behind locality sensitive hashing (LSH) is to have a set of hash functions, $\boldsymbol{h}(.)=\{h_1(.),\cdots,h_H(.)\}$, with each one, $h_l(.)$, randomly splitting the input space into two parts; $h_l(\boldsymbol{y})\in\{0,1\}$. Despite the randomness of an individual hash function, putting multiple hash functions together ensures that the data points belonging to same hashcode bin are similar to each other [Indyk and Motwani, 1998, Zhao et al., 2014, Wang et al., 2017].

In a data-driven LSH approach to learning a hash function $h_l(.)$, a subset of data points $\boldsymbol{Y}^{(l)}$ is subsampled from the superset of data points, $\boldsymbol{Y}=\{\boldsymbol{y}^{(1)},\cdots,\boldsymbol{y}^{(n)}\}$, for which the hash functions (hashcodes) are optimized. A binary split of $\boldsymbol{Y}^{(l)}$ into two subsets can be generalized as a split of the entire input space (so for $\boldsymbol{Y}$) through a binary classification model, and that model is the resulting hash function $h_l(.)$ [Joly and Buisson, 2011, Kulis and Grauman, 2012, Garg et al., 2019b]. Garg et al. [2019a] proposed to find the optimal split of $\boldsymbol{Y}^{(l)}$ such that it generalizes across both training and test (nearly-unsupervised setting). They also proposed to sample the subset $\boldsymbol{Y}^{(l)}$ locally from a hashcode bin so as to capture more fine-grained differences between data points with in the local neighborhoods.

Here we extend their approach from a nearly-unsupervised setting to an unsupervised learning setting. We learn hash functions in a greedy fashion with pseudo code shown in Alg. 1. There are two key steps in the greedy approach for optimizing $h_l(.)$ (see Fig. 1 for a visual illustration). First, as per the $l-1$ number of hash functions optimized so far, we obtain hashcodes based binning of all the data points in $\boldsymbol{Y}$, then select a hashcode bin to sample $\boldsymbol{Y}^l$ from within the selected bin. If the maximal size of a bin is above a certain threshold, it suffices to just select the bin with the maximum size. Otherwise we *select the hashcode bin with the highest entropy* of data points, as shown below,

$$\max_{\boldsymbol{c}}\mathcal{H}(\boldsymbol{\mathcal{Y}}|\boldsymbol{\mathcal{C}}=\boldsymbol{c}), \tag{3}$$

Algorithm 1 Unsupervised Learning of LSH functions

Require: $\boldsymbol{Y} = (\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(n)}), H.$

for $l = 1 \rightarrow H$ **do**

  $\boldsymbol{Y}^{(l)} \leftarrow$ sampleFromHashcodeBinOfHighEntropy$(\boldsymbol{Y}, \boldsymbol{C})$

  $h_l(.) \leftarrow$optimizeSplit($\boldsymbol{Y}^{(l)}, \boldsymbol{Y}, \boldsymbol{C}$)

  $\boldsymbol{C}(:, l) \leftarrow h_l(\boldsymbol{Y}\ )$

end for

Return $h_1(.), \cdots, h_H(.), \boldsymbol{C}.$

wherein $\mathcal{H}(\boldsymbol{\mathcal{Y}}|\boldsymbol{\mathcal{C}} = \boldsymbol{c})$ is entropy of $\boldsymbol{\mathcal{Y}}$ within a particular hashcode bin $\boldsymbol{c}$; the entropy term can be computed in terms of nearest neighbor distances within the bin itself using KL estimator due to Kozachenko and Leonenko [1987].[1]

Having sampled $\boldsymbol{Y}^l$ from a hashcode bin with the highest entropy (Fig. 1(a)), the second step is to optimize its split. In Fig. 1(b), a split of the sampled points corresponds to a split of the entire 2-D input space (dashed line). Between the two choices of splits, the one with vertical dashed line (Navy colored) is optimal since it leads to better splits of the other hash bins (including self). Mathematically, we characterize this criterion of splitting the hashcode bins as,

$$\max_{h_l(.)} \mathcal{H}(h_l(\boldsymbol{y})|h_1(\boldsymbol{y}), \cdots, h_{l-1}(\boldsymbol{y})). \tag{4}$$

It ensures that hash function $h_l(.)$ is disentangled (not redundant) w.r.t. the hash functions optimized previously. It is cheap to compute empirically, by simply counting the fraction of ones from the output of $h_l(.)$ for each of the bins. We refine this criterion further for a hashcode bin that is of a relatively small size (number of data points in it) and has high entropy of $h_l(.)$ (less-biased proportion of ones). For any such hashcode bin $\boldsymbol{c}$, we propose to maximize the KL-divergence between the data distribution from the two partitions of the bin, emerging from $h_l(.)$:

$$\max_{h_l(.)} \mathcal{D}^c_{KL}(p(\boldsymbol{y}|h_l = 0)||p(\boldsymbol{y}|h_l = 1)). \tag{5}$$

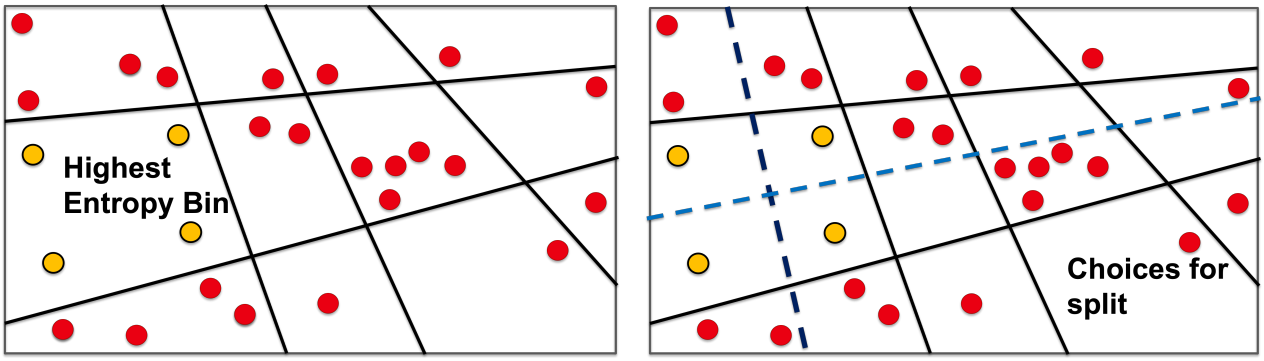
We empirically estimate $\mathcal{D}^c_{KL}(.)$ from the ratio of the nearest neighbor distances within and across the two partitions of the bin [Zhao and Lai, 2020].

We eliminate a large fraction of candidates for a split of $\boldsymbol{Y}^l$ through constraints based on info-theoretic clustering along with a divide and conquer procedure. The algorithm is parallelizable, and is overall compute efficient. While the proposed LSH algorithm has various applications, we evaluate it only for the purpose of robust estimation of transfer entropy.

C MORE DETAILS OF NEUROSCIENCE DATA

The experiments used multiple high-density extracellular electrophysiology probes to simultaneously record spiking activity from a wide variety of areas in the mouse brain, ranging from the subcortical region, such as the thalamus, to multiple the visual cortices, such as primary visual cortex (V1), lateral medial visual area (LM), rostrolateral visual area (RL), anterolateral visual are (AL), anteromedial visual area (AM), etc. The neural activities were recorded while the animals were head-fixed and were passively presented with visual stimuli. The details of the experimental setup can be found in [Siegle et al., 2021, Institute, 2020]. We used animal with session-id `798911424`. One experiment contained a mixture of many stimulus types, such as natural movies, flashes, Gabor filters, drifting gratings, etc. We selected recording trials of drifting gratings as they could strongly elicit neural responses. The visual stimulus of each trial lasted for two seconds with one-second rest in between without any visual stimulus. We randomly subsampled 100 trials without replacement from 13 conditions of drifting gratings with 15 repeated trials each, so totally out of 195 trials. The condition ids are: `275`, `268`, `270`, `284`, `274`, `249`, `261`, `278`, `280`, `256`, `260`, `257`, `281`. The first 500 ms of each trial after stimulus onset was

[1] KL estimator should be accurate for empirical estimates of entropy within local neighborhoods, even in high dimensions, since a small value of k is optimal in such case.



(a) Select highest entropy bin.

(b) Optimize split.

Figure 1: This figure illustrates the learning of a hash function. In 1(a) and 1(b), black lines refer to hash functions learned previously. The intersection of the lines correspond to hashcode bins, with data points (red dots) dispersed across these bins. If data points are highly dispersed within a bin, as is the case for the yellow points, it is a good candidate for sampling data points from it, so as to learn a new hash function which would split the selected bin to reduce the dispersion within it, and potentially split the other bins as well. Mathematically, dispersion of data in high dimensions can be characterized as entropy of the data distribution, see the highest entropy bin in 1(a). In 1(b), there are multiple ways of splitting the high entropy cluster (dashed lines), and the optimal choice is one which splits the other bins as well (Navy colored vertical dashed line).

extracted, as the early visual activities mainly involved feedforward interactions, thus could better reflect the hierarchy of the visual system. The number of neurons in visual cortical areas recorded by one probe was roughly around 100. The raw data was composed of sequences of action potential timestamps or counting processes of each neuron. We time-binned the timestamps with 0.1 ms resolution, then averaged the time series across all neurons for each brain region. The time lags of all TE estimators are all 20 ms, with 40 time steps and the length of each time step is 0.5 ms.

References

Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, 1965.

Sahil Garg, Aram Galstyan, Greg Ver Steeg, and Guillermo A Cecchi. Nearly-unsupervised hashcode representations for biomedical relation extraction. In *Proc. of EMNLP*, 2019a.

Sahil Garg, Aram Galstyan, Greg Ver Steeg, Irina Rish, Guillermo Cecchi, and Shuyang Gao. Kernelized hashcode representations for relation extraction. In *Proc. of AAAI*, 2019b.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. of STOC*, 1998.

Allen Institute. *Visual Coding - Neuropixels*, 2020.

Alexis Joly and Olivier Buisson. Random maximum margin hashing. In *Proc. of CVPR*, 2011.

L. F. Kozachenko and N. N Leonenko. A statistical estimate for the entropy of a random vector. *Problems of Information Transmission*, 1987.

Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

Gábor Lugosi and Andrew Nobel. Consistency of data-driven histogram methods for density estimation and classification. *The Annals of Statistics*, 1996.

David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *Proc. of AISTATS*, 2020.

Jonas Rothfuss, Fabio Ferreira, Simon Walther, et al. Conditional density estimation with neural networks: Best practices and benchmarks. *arXiv preprint arXiv:1903.00954*, 2019.

Joshua H Siegle, Xiaoxuan Jia, Séverine Durand, et al. Survey of spiking in the mouse visual system reveals functional hierarchy. *Nature*, 2021.

Jingdong Wang, Ting Zhang, Nicu Sebe, et al. A survey on learning to hash. *PAMI*, 2017.

Kang Zhao, Hongtao Lu, and Jincheng Mei. Locality preserving hashing. In *Proc. of AAAI*, 2014.

Puning Zhao and Lifeng Lai. Analysis of k nearest neighbor kl divergence estimation for continuous distributions. In *ISIT*, 2020.