Annals of Operations Research (2020) 289:5–32
https://doi.org/10.1007/s10479-019-03464-z

S.I. : OR IN NEUROSCIENCE II

HNCcorr: combinatorial optimization for neuron identification

Roberto Asín Achá[1] · Dorit S. Hochbaum[2] · Quico Spaen[2]

Published online: 16 November 2019

Abstract
We present a combinatorial algorithm for cell detection in two-photon calcium imaging. Calcium imaging is a modern technique used by neuroscientists for recording movies of in-vivo neuronal activity at cellular resolution. The proposed algorithm, named HNCcorr, builds on the combinatorial clustering problem Hochbaum's Normalized Cut (HNC). HNC is a model that trades off two goals: One goal is that the cluster has low similarity to the remaining objects. The second goal is that the cluster is highly similar to itself. The HNC model is closely related to the Normalized Cut problem of Shi and Malik, a well-known problem in image segmentation. However, whereas Normalized Cut is an NP-hard problem, HNC is solvable in polynomial time. The neuronal cell detection in calcium imaging movies is viewed here as a clustering problem. HNCcorr utilizes HNC to detect cells in these movies as coherent clusters of pixels that are highly distinct from the remaining pixels. HNCcorr guarantees, unlike existing methodologies for cell identification, a globally optimal solution to the underlying optimization problem. Of independent interest is a novel method, named *similarity-squared*, that is devised here for measuring similarity. In an experimental study on data from the Neurofinder cell identification benchmark, HNCcorr is a top performer. In particular, it achieves a higher average score than two frequently used matrix factorization algorithms. The Python and Matlab implementations of HNCcorr used here are publicly available. The use of HNCcorr demonstrates that combinatorial optimization is a valuable tool for neuroscience and other biomedical disciplines.

✉ Dorit S. Hochbaum
hochbaum@ieor.berkeley.edu

Roberto Asín Achá
rasin@udec.cl

Quico Spaen
qspaen@berkeley.edu

[1] Department of Computer Science, Universidad de Concepción, Edmundo Larenas 219, Concepción, Chile

[2] Department of Industrial Engineering and Operations Research, University of California, Berkeley 94720, USA

1 Introduction

This work is motivated by a critical challenge facing the neuroscience community, of extracting high quality signals of neuronal activity from calcium-imaging data. While there are excellent optical imaging tools for measuring and recording cellular resolution data, available algorithms that extract the signals from this data do not provide the quality required. State-of-the-art approaches either require tens of hours of manual labor in the processing of each experimental dataset, or else, use methods that provide results of inadequate quality due to the use of heuristic approaches rather than optimization. Due to the shortcomings of existing algorithms, the full potential of the acquired data is not realized. We introduce here a new approach for addressing this problem that utilizes a new combinatorial optimization clustering model, named Hochbaum's Normalized Cut (HNC) (Hochbaum 2010, 2013).

HNC is an intuitive clustering model that combines two desirable goals: The objects within the cluster should be *homogeneous*, and the cluster should be *distinct* from the remaining objects. Homogeneity is captured by high total similarity between the objects within the cluster, whereas distinctness is represented by small similarity between the cluster and its complement. HNC has been successfully applied to a wide array of applications. Hochbaum et al. (2013) showed that HNC outperforms spectral clustering for image segmentation. Baumann et al. (2019) provide a comparative study of binary classification algorithms including a supervised variant of HNC. They demonstrate that the supervised version of HNC is competitive with state-of-the-art machine learning techniques. Other successful applications of HNC include enhancing the detection of nuclear radiation sources (Yang et al. 2013), evaluating the effectiveness of drugs (Hochbaum et al. 2012), and video-tracking (Fishbain et al. 2013).

Calcium imaging is an imaging technique used by neuroscientists for measuring calcium concentrations at a cellular resolution in live and behaving animals (Stosiek et al. 2003). It is commonly used to simultaneously record the neuronal activity of thousands of neurons, the primary component of the central nervous system. When a neuron activates (fires), calcium ions are released into the cell. The cell then becomes fluorescent due to calcium sensitive, genetically modified proteins. After the spike, the calcium concentration rapidly decays to normal levels. These changes in fluorescence are recorded with fast-scanning laser microscopes, producing movies of the fluorescence of cells in a slice of the brain. In these movies, a pixel is characterized by a fluorescence signal that changes over time. From these movies, researchers extract the activity signals from individual neurons by estimating the time-varying fluorescence of these cells. The extracted activity signals are then used as the starting point for all subsequent analysis to test hypotheses about the function of neurons and neural circuits in relation to the behavior of the animal. We focus here on two-photon calcium-imaging, where cells have minimal overlap between them.

A crucial step in the analysis of calcium imaging datasets is to identify the locations of the individual cells in the movie in order to extract their signal. This *cell identification* problem consists of outlining the spatial footprint, represented by a set of pixels, of each cell in the imaging plane. The identification is complicated by high noise levels in the recorded movie, the presence of out-of-focus fluorescence sources above or below the imaging plane, and the fact that most cells are only visible when they activate. Our interest here is in active cells with measurable changes in fluorescence. The task of cell identification is cast here as a clustering problem of constructing a cohesive cluster of pixels that are distinct from their environment. The spatial footprint of a cell is such a cluster since the pixels in the footprint share the unique time-varying signal of the contained cell.

If done manually, the process of identifying cells requires hours of manual user input for each dataset, which, accumulated over a single project, adds up to hundreds of hours of time committed by researchers. Hence, a number of automated methods have been developed for cell identification in calcium imaging movies. These existing techniques can be classified into three classes: Semi-manual region of interest (ROI) detection (Kaifosh et al. 2014; Driscoll et al. 2017), shape-based detection algorithms (Gao 2016; Klibisz et al. 2017; Apthorpe et al. 2016; Pachitariu et al. 2013), and matrix factorization algorithms (Mukamel et al. 2009; Maruyama et al. 2014; Pnevmatikaki and Paninski 2013; Diego-Andilla and Hamprecht 2014; Pnevmatikakis et al. 2016; Pachitariu et al. 2017; Levin-Schwartz et al. 2017). Semi-manual ROI detection techniques rely on user's input for detecting and segmenting cells. This process has been reported to be highly labor-intensive (Resendez et al. 2016) and may miss cells with a low signal to noise ratio or a low activation frequency. Shape-based identification methods locate the characteristic shape(s) of cells using deep learning (Gao 2016; Klibisz et al. 2017; Apthorpe et al. 2016) or dictionary learning (Pachitariu et al. 2013). Shape-based techniques are typically applied to a summary image of the movie obtained by removing the time dimension and compressing the movie to a single average intensity image. The third class of techniques uses a matrix factorization model to decompose a movie into the spatial and temporal properties of the individual neuronal signals. The use of the matrix factorization algorithm CNMF (Pnevmatikakis et al. 2016) is currently prevalent for the task of cell identification. The name CNMF is derived from the algorithm's use of constrained non-negative matrix factorization.

We demonstrate here how HNC is adapted to an algorithm, named HNCcorr, for cell identification in two-photon calcium imaging movies. The name HNCcorr is derived from HNC and the use of a similarity measure based on correlation. A major distinction of HNCcorr as compared to most alternative algorithms is that HNC is solved efficiently and to global optimality (Hochbaum 2010, 2013). The optimality guarantee of HNC makes the output of the optimization model transparent, since the effect of the model input and parameters on the resulting optimal solution is well understood. In contrast, most other approaches for cell identification, such as matrix factorization algorithms, rely on non-convex optimization models that are intractable. This means that the algorithms cannot find a global optimal solution to their optimization model but instead find a locally optimal solution, which is dependent on the initial solution. As a result, these algorithms provide no guarantee on the quality of the delivered solutions and cells may remain undetected.

One unique feature of HNC is the use of pairwise similarities. Pairwise similarities considerably enhance the quality of prediction for data mining as has been noted in the past: by Dembczyński et al. (2009); Baumann et al. (2019) for machine learning purposes, by Ryu et al. (2004) for improved medical diagnosis, and by Zhu et al. (2003) in semi-supervised learning. For HNCcorr, we devised a novel method for computing similarities between pairs of pixels named $(\text{SIM})^2$, *similarity-squared*. The idea of $(\text{SIM})^2$ is to associate, with each pixel, a feature vector of correlations with respect to a subset of pixels. Similarities between pairs of pixels are then computed as the similarity of the respective two feature vectors. The novelty in similarity-squared is that it replaces the direct similarity between a pair of objects - pixels here - by a comparison of how each of the two objects relates to a reference set of objects. For example, one advantage of $(\text{SIM})^2$ over regular pairwise correlation is that it considers any two background pixels, pixels not belonging to a cell, as highly similar whereas correlation deems them dissimilar. This improves the clustering since it incentivizes that background pixels are grouped together.

We present here the experimental performance of the HNCcorr on the Neurofinder benchmark (CodeNeuro 2016) for cell identification in two-photon calcium imaging datasets.

This benchmark is currently the only available benchmark that objectively evaluates cell identification algorithms. The datasets in this benchmark are annotated two-photon calcium imaging movies recorded under varying experimental conditions. On this benchmark, HNCcorr achieves a higher average score than two frequently used matrix factorization algorithms CNMF (Pnevmatikakis et al. 2016) and Suite2P (Pachitariu et al. 2017).

We further provide a running time comparison between the MATLAB implementations of HNCcorr, CNMF, and Suite2P. HNCcorr has similar running time performance as Suite2P and is about 1.5 times faster than CNMF. Python and MATLAB implementations of HNCcorr are available at https://github.com/quic0/HNCcorr.

1.1 Summary of motivation and contributions

Motivation

- Calcium imaging is a crucial analysis tool in modern neuroscientific research.
- There is a lack of effective algorithms for cell identification, thus inhibiting the analysis of calcium imaging data.
- An optimization model to identify cells in calcium imaging data, that is solved efficiently, would address the lack of effective algorithms.

Contributions

- Devising the HNCcorr algorithm for cell identification in two-photon calcium imaging movies.
- Contributing to the development of automated algorithmic infrastructure that delivers an accurate extraction of neuronal activity from calcium imaging data.
- Introducing the first combinatorial optimization model of the cell identification problem that is solved in polynomial time to global optimality.
- Introducing the novel similarity measure similarity-squared, $(\text{SIM})^2$, which is expected to be applicable beyond the context of calcium imaging. The novelty in similarity-squared is that it replaces the direct similarity between a pair of objects by a comparison of how each of the two objects relates to a reference set of objects.
- Presenting an extensive experimental study demonstrating that HNCcorr delivers superior performance to that of two commonly-used matrix factorization algorithms.

1.2 Overview of this work

The remainder of this work is organized as follows: In Sect. 2, we describe the HNC problem. In Sect. 3, we describe methods for solving HNC. In Sect. 4, we discuss a solution for scaling similarity-based machine learning methods, such as HNC, to large instances. In Sect. 5, we describe the HNCcorr algorithm and how it relates to HNC. We describe the novel similarity measure $(\text{SIM})^2$ in Sect. 5 since the motivation for $(\text{SIM})^2$ is tied to the problem of cell identification. In Sect. 6, we evaluate the experimental performance of HNCcorr on the Neurofinder benchmark datasets. Finally, we present concluding remarks in Sect. 7.

2 The HNC model for similarity-based clustering

HNC (Hochbaum 2010, 2013) is a intuitive model for similarity-based data mining and clustering. It is applicable to both supervised and unsupervised learning problems. We describe in this section how HNC is defined and how it relates to Normalized Cut (Shi and Malik 2000), a well-known problem from image segmentation.

2.1 Notation

Let $G = (V, E)$ denote an undirected graph. Let n denote the number of nodes in V, and let m be the number of edges in E. Every edge $[i, j] \in E$ has an associated weight $w_{ij} \geq 0$. We denote the weighted degree of node $i \in V$ by $d_i = \sum_{[i,j]\in E} w_{ij}$. For $A, B \subseteq V$, let $C(A, B) = \sum_{[i,j]\in E,\ i\in A,\ j\in B} w_{ij}$ be the sum of weights of the edges between nodes in the set A and those in set B. Note the following relation for undirected graphs: $\sum_{i\in A} d_i = 2C(A, A) + C(A, V\backslash A)$.

Let the set $S \subset V$ be a non-empty set of nodes. Its complement is $\bar{S} = V\backslash S$. The sets $(S, \bar{S})$ form a bi-partition referred to here as a *cut*. The *capacity* of a cut $(S, \bar{S})$ is $C(S, \bar{S})$. Given a source node $s \in V$ and sink node $t \in V$, an s, t-cut is a cut $(S, \bar{S})$ where $s \in S$ and $t \in \bar{S}$. The set S of an s, t-cut $(S, \bar{S})$ is known as the *source set* of the cut, and the set $\bar{S}$ is known as the *sink set* of the cut. The minimum s, t-cut problem on G is to find an s, t-cut that minimizes $C(S, \bar{S})$.

For a directed graph $G_D = (V_D, A)$, let $w_{ij} \geq 0$ denote the arc weight of arc $(i, j) \in A$. We use the same notation as for undirected graphs to denote a cut $(S, \bar{S})$ and its capacity $C(S, \bar{S})$. Note that the capacity $C(S, \bar{S})$ of a cut $(S, \bar{S})$ in a directed graph includes only the sum of weights on arcs from S to $\bar{S}$ and not on arcs in the reverse direction. Similar to undirected graphs, the minimum s, t-cut problem on a directed graph G_D is to find an s, t-cut that minimizes $C(S, \bar{S})$.

2.2 The HNC model

HNC (Hochbaum 2010, 2013) is a graph-based clustering model that trades-off two objectives: The homogeneity of the cluster and the distinctness of the cluster to the remaining objects. HNC is defined here on an undirected graph $G = (V, E)$, but it applies to directed graphs as well. The node set V is the set of objects, and the edges in E represent pairwise similarity relations between objects. The pairwise similarity along edge $[i, j] \in E$ is measured with a similarity weight $w_{ij} \geq 0$. The objects i and j are considered more similar as w_{ij} increases.

The cluster of interest in HNC is a non-empty set $S \subset V$. We represent the homogeneity of the cluster S with $C(S, S)$, the total sum of similarities between objects in the cluster. We measure distinctness with $C(S, \bar{S})$, which is the capacity of the cut $(S, \bar{S})$. The HNC model balances the maximization of $C(S, S)$ and the minimization of $C(S, \bar{S})$.

HNC requires that S and $\bar{S}$ contain one or more seed objects, otherwise the optimal solution is to select $S = V$ with $C(S, S) = \sum_{[i,j]\in E} w_{ij}$ and $C(S, \bar{S}) = 0$. Let $S_P \subset V$ denote a non-empty set of *positive seeds* that must be contained in the cluster S. Similarly, $S_N \subset V$ denotes the non-empty set of *negative seeds* that are in $\bar{S}$. These seed sets guarantee that both S and $\bar{S}$ are non-empty. In a supervised context, these seed sets contain the labeled objects in the training data.

The HNC model is to optimize the ratio of distinctness and homogeneity:

$$\min_{\substack{\emptyset \subset S \subset V \\ S_P \subseteq S,\, S_N \subseteq \bar{S}}} \frac{C\left(S, \bar{S}\right)}{C\left(S, S\right)}. \tag{1}$$

This objective is equivalent to minimizing the cut capacity $C(S, \bar{S})$ divided by the weighted degree of the nodes in S:

$$\min_{\substack{\emptyset \subset S \subset V \\ S_P \subseteq S,\, S_N \subseteq \bar{S}}} \frac{C\left(S, \bar{S}\right)}{\sum_{i \in S} d_i}, \tag{2}$$

as shown next:

Lemma 1 (Hochbaum 2010) *The set of optimal solutions to* (1) *and* (2) *are identical.*

Proof

$$\frac{C(S, \bar{S})}{C(S, S)} = \frac{2C(S, \bar{S})}{\sum_{i \in S} d_i - C(S, S')} = \frac{2}{\frac{\sum_{i \in S} d_i}{C(S, \bar{S})} - 1}.$$

The problems are thus equivalent, since they have equivalent objectives and the same set of feasible solutions. □

2.3 Relation to normalized cut

HNC is a close relative of the Normalized Cut (NC) problem. Its use in the context of image segmentation was popularized by Shi and Malik (2000). The NC problem is defined as:

$$\min_{\emptyset \subset S \subset V} \frac{C\left(S, \bar{S}\right)}{\sum_{i \in S} d_i} + \frac{C\left(S, \bar{S}\right)}{\sum_{i \in \bar{S}} d_i}. \tag{3}$$

Compared to (2), the NC problem has an additional term in the objective. Although this change may seem minor, it has profound impact on the complexity of the problem. In fact, Shi and Malik showed that the Normalized Cut problem is NP-hard (Shi and Malik 2000) whereas there is a practical, polynomial-time algorithm for the HNC problem (Hochbaum 2010). In previous work (Sharon et al. 2006), it was mistakenly assumed that the HNC problem is identical to Normalized Cut and therefore NP-hard.

The Normalized Cut problem is often used as the theoretical motivation for the *spectral clustering* method. Spectral clustering was popularized by Shi and Malik (2000) as a heuristic for the Normalized Cut problem, since spectral clustering solves a continuous relaxation of the Normalized Cut problem. HNC was shown to be a form of discrete relaxation of normalized cut (Hochbaum 2013). Yet, HNC is solved more efficiently than the spectral clustering method. Furthermore, an extensive empirical comparison of HNC to spectral clustering for the purpose of image segmentation was provided by Hochbaum et al. (2013). The outcomes of that comparison indicate that the HNC solutions are superior, both visually and in terms of approximating the objective of the Normalized Cut problem, to the spectral clustering algorithm.

2.4 Linearizing ratio problems

A common method to solve ratio problems such as (1) and (2) is to instead consider an optimization problem over the epigraph of the ratio function. When you minimize a ratio objective over a feasible region X, $\min_{x \in X} \frac{f(x)}{g(x)}$, this optimization problem is equivalent to:

$$\min_{x,\lambda\geq 0} \lambda \quad \text{s.t} \quad \frac{f(x)}{g(x)} \leq \lambda, \; x \in X \quad \Leftrightarrow \quad \min_{x,\lambda\geq 0} \lambda \quad \text{s.t} \quad f(x) - \lambda g(x) \leq 0, \; x \in X$$

This is equivalent to finding the smallest λ such that there is a "yes" answer to the following λ-question: "Is there a solution $x \in X$ such that $f(x) - \lambda g(x) \leq 0$?" This reduces the optimization problem to a sequence of oracle calls to the λ-question.

For the HNC formulation in (1), we can answer the associated λ-question by solving the linear optimization problem in (4) and checking whether the objective value is non-positive:

$$\min_{\substack{\emptyset\subset S\subset V \\ S_P\subseteq S,\, S_N\subseteq\bar{S}}} C\left(S,\bar{S}\right) - \lambda C(S,S). \tag{4}$$

Similarly, we answer the λ-question for the HNC formulation in (2) with the linear optimization problem in (5):

$$\min_{\substack{\emptyset\subset S\subset V \\ S_P\subseteq S,\, S_N\subseteq\bar{S}}} C\left(S,\bar{S}\right) - \lambda \sum_{i\in S} d_i. \tag{5}$$

We note that the linear formulations in (4) and (5) are alternative formulations of HNC irrespective of their associated ratio problems, since they capture the trade-off between homogeneity and distinctness.

3 Algorithms for solving HNC

In Sect. 3.1 we provide integer programming formulations for both (4) and (5). We show how these formulations imply efficient polynomial time algorithms based on minimum cut for solving the linear HNC problems with a fixed value of λ (Hochbaum 2010, 2013). In Sect. 3.2, we extend this analysis and demonstrate how (5) is solved for all λ values simultaneously in the same complexity, that of a single minimum cut, by using a parametric minimum cut procedure (Hochbaum 2010, 2013). This directly provides an algorithm to solve the ratio version of HNC.

3.1 Solving for fixed λ values

In this section, we demonstrate how to solve the linearized HNC problems in (4) and (5) for a fixed value of $\lambda \geq 0$. We first present an integer programming formulation for (4). The integer program uses the decision variable x_i to denote whether node $i \in V$ is in the cluster S:

$$x_i = \begin{cases} 1 & \text{if } i \in S, \\ 0 & \text{if } i \in \bar{S}. \end{cases}$$

Another decision variable z_{ij} denotes whether edge $[i,j] \in E$ is in the cut, and the decision variable y_{ij} denotes for edge $[i,j]$ whether i and j are both in the cluster S:

$$z_{ij} = \begin{cases} 1 & \text{if } i \in S, j \in \bar{S} \text{ or } i \in \bar{S}, j \in S, \\ 0 & \text{otherwise}, \end{cases} \qquad y_{ij} = \begin{cases} 1 & \text{if } i,j \in S, \\ 0 & \text{otherwise}. \end{cases}$$

With these decision variables, the following integer programming problem is a formulation of HNC (Hochbaum 2010):

$$\begin{aligned}
\min \quad & \sum_{[i,j]\in E]} w_{ij} z_{ij} - \lambda \sum_{[i,j]\in E]} w_{ij} y_{ij}, & (6)\\
\text{s.t.} \quad & x_i - x_j \le z_{ij} \quad \forall [i,j] \in E, \quad x_j - x_i \le z_{ji} \quad \forall [i,j] \in E,\\
& y_{ij} \le x_i \quad \forall [i,j] \in E, \quad y_{ij} \le x_j \quad \forall [i,j] \in E,\\
& x_i = 1 \quad \forall i \in S_P, \quad x_i = 0 \quad \forall i \in S_N,\\
& x_i \in \{0,1\} \quad \forall i \in V, \quad y_{ij}, z_{ij} \in \{0,1\} \quad \forall [i,j] \in E.
\end{aligned}$$

This integer program is recognized as a monotone integer programing with up to three variables per inequality (IP3) (Hochbaum 2002). Monotone IP3 are integer programs with up to three variables per inequality, where in each constraint two of the variables appear with opposite sign coefficients and a third variable, if present, appears only in one constraint. A typical monotone constraint on three variables is $a_{ij}x_i - b_{ij}x_j \le c_{ij} + z_{ij}$ where a_{ij} and b_{ij} are non-negative. Any monotone IP3 is solvable with a minimum cut on an associated directed graph (Hochbaum 2002). Monotone IP3 integer programs have multiple applications. For instance, it was used for nuclear threat detection (Hochbaum and Fishbain 2011).

The graph associated with the formulation in (6) has $m+n$ nodes and up to $4m+2n$ arcs, where n is the number of nodes and m is the number of edges in G. The complexity of solving the integer program in (6) is therefore $T(m+n+2, 4m+2n)$. The notation $T(n,m) = O(nm \log \frac{n^2}{m})$ denotes the complexity of solving the minimum s,t-cut problem on a graph with n nodes and m arcs with the HPF (Hochbaum's PseudoFlow) algorithm (Hochbaum 2008) or the Push-relabel algorithm (Goldberg and Tarjan 1988).

The linearized version of (2) as given in (5) translates to another monotone IP3 for which the corresponding graph is smaller (Hochbaum 2013):

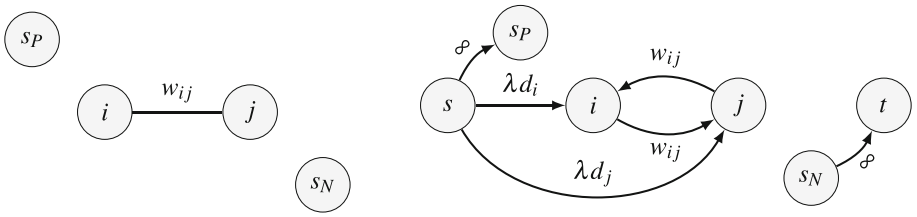
$$\begin{aligned}
\min \quad & \sum_{[i,j]\in E]} w_{ij} z_{ij} - \lambda \sum_{i\in V} d_i x_i, & (7)\\
\text{s.t.} \quad & x_i - x_j \le z_{ij} \quad \forall [i,j] \in E, \quad x_j - x_i \le z_{ji} \quad \forall [i,j] \in E,\\
& x_i = 1 \quad \forall i \in S_P, \quad x_i = 0 \quad \forall i \in S_N,\\
& x_i \in \{0,1\} \quad \forall i \in V, \quad z_{ij} \in \{0,1\} \quad \forall [i,j] \in E.
\end{aligned}$$

The graph associated with the formulation in (7) and contains only $O(n)$ nodes. We now show the construction of the graph for this formulation, referred to as the *auxiliary* graph (Hochbaum 2013).

Let the auxiliary graph be a directed graph $G_A = (V \cup \{s,t\}, A)$. The arc set A consists of the following sets of arcs: For every edge $[i,j] \in E$, add the two arcs (i,j) and (j,i) to A. Both arcs have a capacity of $w_{ij} \ge 0$. We also add arcs (s,i) with capacity λd_i for every node $i \in V$. Finally, for every positive seed $i \in S_P$ we add an infinite capacity arc (s,i), and for every negative seed $j \in S_N$ we add an infinite capacity arc (j,t). A schematic of the auxiliary graph is provided in Fig. 1.

The source set of a minimum s,t-cut in the auxiliary graph G_A is an optimal solution to the linearized HNC problem in (5):

Theorem 1 (Hochbaum 2013) *The source set S^* of a minimum s,t-cut $(S^* \cup \{s\}, \bar{S}^* \cup \{t\})$ in G_A is an optimal solution to the linearized HNC problem in (5) for a fixed value of λ.*



(a) Similarity graph G with one edge. **(b)** Auxiliary graph G_A.

Fig. 1 Schematics of a similarity graph G and the corresponding auxiliary graph G_A for the linearized HNC in (5). The node $s_P \in S_P$ is a positive seed, whereas the node $s_N \in S_N$ is a negative seed

Proof Let $(S \cup \{s\}, \bar{S} \cup \{t\})$ be any finite s,t-cut in G_A. The infinite capacity arcs (s,i) for $i \in S_P$ guarantee that $S_P \subseteq S$. Similarly, $S_N \subset \bar{S}$. Thus, the set S is a feasible solution for 5. G_A always contains a finite cut when the seed sets are disjoint, $S_P \cap S_N = \emptyset$.

The capacity of cut $(S \cup \{s\}, \bar{S} \cup \{t\})$ is equal to:

$$C(S \cup \{s\}, \bar{S} \cup \{t\}) = C(S, \bar{S}) + \lambda \sum_{i \in \bar{S}} d_i = C(S, \bar{S}) + \lambda \left(D - \sum_{i \in S} d_i \right)$$
$$= \lambda D + C(S, \bar{S}) - \lambda \sum_{i \in S} d_i,$$

where $D = \sum_{i \in V} d_i$ denotes the sum of weighted degrees. Thus, minimizing the capacity of the cut is equivalent to minimizing $C(S, \bar{S}) - \lambda \sum_{i \in S} d_i$. □

The auxiliary graph has $n + 2$ nodes and up to $2m + 2n$, since every edge $[i, j] \in E$ has two associated arcs and every node has up to two associated arcs. One arc connecting it to the source node s with capacity λd_i, and potentially another arc that connects a seed node to either the source node s or to the sink node t.

Corollary 1 *The linearized HNC problem in* (5) *for a fixed value of* λ *is solvable in* $O(T(n + 2, 2m + 2n))$.

Since the linearized HNC problem in (5) provides a more efficient algorithm than the problem in (4), we henceforth consider only formulation (5) and its associated ratio problem (2).

3.2 Solving simultaneously for all λ values

Whereas the previous section describes an algorithm to solve the problem in (5) for a fixed value of λ, we now show that we can solve (5) simultaneously for all values of λ in the complexity of a single minimum cut with a parametric minimum cut problem (Gallo et al. 1989; Hochbaum 2008). This directly implies that the ratio version of HNC in (2) is solved in the same complexity, since these problems are equivalent to finding the smallest λ such that the objective of the linear problem in (5) is non-positive.

Parametric minimum cut solves the minimum cut problem as a function of λ on a special type of graph, the *parametric flow graph*. A directed s,t-graph $G(\lambda)$ is a parametric flow graph if all arcs capacities adjacent to the source node s are monotone non-decreasing and all arc capacities adjacent to the sink node t are monotone non-increasing as a function of λ.

The only arcs in G_A whose capacity depends on λ are the arcs adjacent to the source node s. The capacity of arc $(s, i) \in A$ is λd_i and is monotone non-decreasing in λ since d_i, the weighted degree of node i, is non-negative for all nodes $i \in V$. The graph $G_A(\lambda)$ is therefore a parametric flow graph.

It was shown that the source set of a minimum cut in a parametric flow graph as a function of λ are nested:

Lemma 2 (Gallo et al. 1989; Hochbaum 2008) *Let $S^*(\lambda)$ denote the source set of a minimum s, t-cut in the parametric flow graph $G(\lambda)$. Then, for $\lambda_1 < \lambda_2$:*

$$S^*(\lambda_1) \subseteq S^*(\lambda_2).$$

Although the capacity of the cut increases with increased λ, the source set of the minimum cut is strictly incremented at most n times. This implies that there are at most n values of $0 \leq \lambda_1 < \lambda_2 < \cdots < \lambda_\ell$ for which the source set is augmented by at least one node. We refer to such λ-values as *breakpoints*. Let the sets $S_1^* \subset S_2^* \subset \cdots \subset S_\ell^*$ denote the associated minimal source sets of the minimum cut, where S_1^* is the source set of the minimum cut for $\lambda \in [0, \lambda_1]$.

Gallo et al. (1989) showed that the breakpoints and their associated source sets are found in the complexity of a single minimum cut with the push-relabel algorithm (Goldberg and Tarjan 1988). Hochbaum (2008) showed that the HPF algorithm also identifies all breakpoints in the complexity of a single minimum cut. These are the only two known algorithms that solve the parametric minimum cut in the complexity of a single minimum cut. Since the auxiliary graph is a parametric flow graph with $n + 2$ nodes and $2m + 2n$ arcs, it implies that solutions to the linearized HNC in (5) for all values of λ are found simultaneously in $T(n + 2, 2n + 2m)$.

With the breakpoints λ_i and associated source sets S_i^* for $i = 1, \ldots, \ell$ we are able to solve the ratio version of HNC as well. Recall that these problems are equivalent to finding the smallest λ such that the objective value of (5) is non-positive. Since the source sets only change at the breakpoints, it is sufficient to find the smallest λ-breakpoint where the objective value of (5) is non-positive. The source set for this breakpoint is the optimal cluster for the ratio version of HNC.

Theorem 2 (Hochbaum 2013) *The HNC ratio problem in (2) is solved in $T(n + 2, 2n + 2m)$ with a parametric minimum cut procedure.*

4 Sparse computation: addressing the challenge of scaling in the presence of pairwise similarities

Computing pairwise similarities poses a challenge at scale as the number of pairwise similarities grows quadratically in the number of objects (Hochbaum and Baumann 2016; Baumann et al. 2016, 2017). As a result, it is prohibitive to compute or store all pairwise similarities for large datasets. Hochbaum and Baumann (2016) proposed the method of *sparse computation* to overcome this challenge.

Sparse computation identifies a set of relevant pairwise similarities without computing the pairwise similarities first. For binary classification, it was shown, for a large number of datasets, that the accuracy obtained with sparse computation is almost identical to the accuracy obtained with the complete set of all pairwise similarities (Hochbaum and Baumann 2016). See Fig. 2 for an example. Sparse computation maintains high accuracy levels by discarding

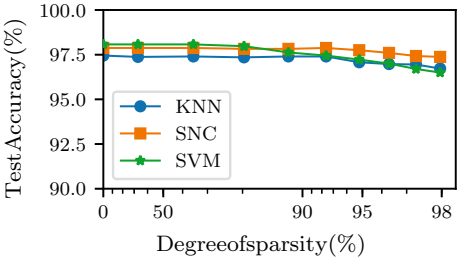


Fig. 2 Accuracy at selected sparsity levels for the letter recognition dataset (Frey and Slate 1991) obtained with three similarity-based classifiers [k-nearest neighbors, a supervised variant of HNC (Yang et al. 2013), and SVM (Cortes and Vapnik 1995)] on a sparse set of pairwise similarity generated with sparse computation. Sparsity, reported on a log scale, is measured as the percentage of all possible pairwise similarities that are not computed. The runtime of the algorithms is inversely proportional to the sparsity level

negligibly small similarities between far-away objects without computing their pairwise similarity first. The edge is discarded for each pairwise similarity that is deemed negligible, thus rendering the graph sparse.

While sparse computation enables the scaling of similarity-based machine learning methods to larger datasets, it also has benefits for small datasets. Foremost, it substantially reduces the running time of similarity-based clustering or classification methods such as HNC or k-nearest neighbors at almost no loss in accuracy (Hochbaum and Baumann 2016; Baumann et al. 2016). The computational bottleneck for these methods is typically the computation of the pairwise similarities. In addition, sparse computation helps to discern structure in the datasets by discarding negligible pairwise similarities.

Sparse computation takes as input a dataset with n objects represented by their feature vectors $R_1, \ldots, R_n \in \mathbb{R}^d$. The method consists of two steps: dimension reduction, and grid construction and pair selection. In the dimension reduction step, the data is projected onto a low-dimensional space. Sparse computation relies on closeness in the low-dimensional space as a proxy for similarity in the original space. In the grid construction and pair selection step, pairs of close objects in the low-dimensional space are identified. The output of sparse computation consists of the set of such pairs. These pairs form the edge set E for the similarity graph used by similarity-based machine learning algorithms. Next, we explain each step in more detail.

In the dimension reduction step, the input data is projected from a d-dimensional space onto a p-dimensional space, where $p \ll d$. The projection is done with a probabilistic variant of principle component analysis (PCA) called approximate PCA (Hochbaum and Baumann 2016). Approximate PCA is based on the work by Drineas et al. (2006). Let the data in the p-dimensional space be normalized, i.e., the values of each dimension are scaled to the range $[0, 1]$. The complexity of this step is $O(d^2n + d^3)$, where n is the number of objects.

In the grid construction and pair selection step, the goal is to select objects that are close in the low-dimensional space. In particular, we select all pairs of objects that have an L_∞ distance smaller or equal to ω in the p-dimensional space. This is achieved as follows. First, the range of values along each dimension is subdivided into $\kappa = \frac{1}{\omega}$ equally long intervals. This partitions the p-dimensional space into κ^p grid blocks. Parameter κ denotes the grid resolution. Each object is thus assigned to a single grid block based on its p coordinates.

Since the largest L_∞ distance within a block is equal to $\omega = \frac{1}{\kappa}$, all pairs of objects that belong to the same block are selected and deemed relevant. In addition, some pairs of objects are within a distance of ω but fall in different yet adjacent blocks. To select those pairs as

well, adjacent blocks need to be considered. Each block has up to $3^p - 1$ neighbors in a p-dimensional space. By selecting all pairs of objects that are assigned to adjacent blocks, it can be guaranteed that all pairs of objects whose distance is less than or equal to ω are selected. It is possible that pairs of objects whose L_∞ distance is more than ω but less than 2ω are selected as well, whereas pairs of objects whose distance is more than 2ω will not be selected. The pairwise similarity is computed for the selected pairs of objects. The complexity of computing the pairs of objects is $O(3^d n)$.

The set of pairs forms the edge set E. The number of pairs that are selected is controllable with the grid resolution κ and the dimension p of the low-dimensional space.

5 A description of the HNCcorr algorithm

HNCcorr is an adaptation of HNC for cell identification in calcium imaging. It takes as input a motion-corrected calcium imaging movie and outputs the spatial footprints of the cells that were identified in the movie. The resulting spatial footprints can then be used to identify the activity signal of each cell with specialized algorithms (cf. Vogelstein et al. 2010; Grewe et al. 2010; Theis et al. 2016; Pnevmatikakis et al. 2013; Jewell and Witten 2018).

We now give a brief description of HNCcorr. Additional details on each aspect of the algorithm is provided in the appropriate subsections. The HNCcorr algorithm initializes with a nearly-exhaustive ordered list of candidate locations. A candidate location is a single pixel in the imaging plane that is indicative of a potential cell. Section 5.2 describes how this list is generated. HNCcorr then evaluates the ordered list of candidate locations one at a time. Let us refer to the selected pixel as pixel i. First, HNCcorr checks if pixel i was part of a spatial footprint of a previously identified cell. If it is, then pixel i is discarded. This prevents repeated identification of the same cell. If it was not, it constructs a graph on a subset of the pixels and computes the pairwise similarities with $(\text{SIM})^2$ as explained in Sects. 5.3 and 5.4. For this graph, we solve the HNC problem in (5) simultaneously for all λ with the seeds selected as in Sect. 5.5. We refer to the clusters for each λ-breakpoint in the solution as *candidate clusters*. Based on these candidate clusters, a simple post-processor decided whether a cell was detected and returns its spatial footprint if any. The post-processor is discussed in Sect. 5.6. Potential extensions for HNCcorr are discussed in Sect. 5.7, and additional notation is introduced in Sect. 5.1

The pseudocode of the HNCcorr algorithm is provided in Algorithm 1. The core elements of the HNCcorr algorithm are the construction of the graph, the computation of similarity weights with $(\text{SIM})^2$, and the associated HNC instance. The other components of the algorithm, e.g. the procedure for selecting candidate locations and the post-processing method for candidate clusters, can be replaced with alternative methods. Python and MATLAB implementations of HNCcorr are available on GitHub at https://github.com/quic0/HNCcorr.

5.1 Notation

A calcium imaging movie consists of T frames each with $N = n_1 \times n_2$ pixels . We assume that the pixels are numbered in row-major order. Each pixel i has an associated fluorescence vector $\mathbf{x}_i \in \mathbb{R}_+^T$. Furthermore, the pairwise correlation $\text{corr}(u, v)$ between pixel i and j is:

$$\text{corr}(u, v) = \frac{1}{T-1} \frac{(\mathbf{x}_u - \hat{\mu}(\mathbf{x}_u))^\mathsf{T}(\mathbf{x}_v - \hat{\mu}(\mathbf{x}_v))}{\hat{\sigma}(\mathbf{x}_u)\,\hat{\sigma}(\mathbf{x}_v)},$$

Algorithm 1 Overview of the HNCcorr algorithm.

Input: M: Movie with fluorescence vector $\mathbf{x}_i$ for pixels $i = 1, \ldots, n$
Output: Set of spatial footprints of cells
Parameters:
- $n_{\text{patch}} = 31$: Patch size in pixels
- $\gamma = 32\%$: Percentage of pixels selected from the patch for the reference set
- $k = 3$: Size of super pixel for positive seeds
- $n_{\text{neg}} = 10$: Number of negative seeds
- $\rho = 13$ - Radius of the negative seeds circle (depends on m),
- $\alpha = 1$: Scaling factor
- $p = 3$: Dimension of the low-dimensional space for sparse computation
- $\kappa = 35$: Grid resolution for sparse computation

function HNCcorr(M)
  $L \leftarrow$ CandidateLocations(M)
  $S \leftarrow \emptyset$ `// Set of spatial footprints`
  $SP \leftarrow \emptyset$ `// Set of pixels in spatial footprints`
  for $cl \in L$
    if $cl \in SP$ `// Skip to prevent duplication`
      continue
    Construct $n_{\text{patch}} \times n_{\text{patch}}$ patch of pixels centered at cl. V is the set of pixels in the patch. $n = |V|$.
    Let S_P be the $k \times k$ super pixel centered at cl.
    Select S_N, set of n_{neg} negative seeds, uniformly from a circle of radius ρ centered at lc.

    Sample without replacement pixels $r_1, \ldots, r_{n_R}$ from V for $n_R = \gamma \times n$. `// Sample reference set for (SIM)` [2]
    for $i \in V$
      $(R_i)_h = corr(x_i, x_{r_h})$ for $h = 1, \ldots, n_R$.

    Use sparse computation with feature vectors R_i, dimension p, and resolution κ to determine edge set E.
    for $[i, j] \in E$
      $w_{ij} = \exp\left(-\alpha \|R_i - R_j\|_2^2\right)$.

    Solve linearized HNC in (5) on graph $G = (V, E)$ with weights w_{ij} and seed sets S_P and S_N.
    Let $S_1, \ldots, S_\ell$ denote the candidate clusters.

    Clean each candidate cluster $S_1, \ldots, S_\ell$ to obtain the cleaned clusters $S_1^C, \ldots, S_\ell^C$.
    $C \leftarrow$ PostProcessor($S_1^C, \ldots, S_\ell^C$) `// Select best cluster (empty set if not a cell)`
    if $C \neq \emptyset$
      $S \leftarrow S \cup \{C\}$
      $SP \leftarrow SP \cup C$
  return S

where $\hat{\mu}(\mathbf{x})$ is the sample mean of the vector $\mathbf{x}$ and $\hat{\sigma}(\mathbf{x})$ is the sample standard deviation.

5.2 Generating candidate locations for cells

The candidate locations are stored in an ordered list of pixels that indicate a potential location of a cell in the dataset. With each candidate location, the goal is to identify a new cell. The candidate locations are generated at the start of the algorithm and processed one at a time. As mentioned, candidate locations are discarded if they are in previously identified cell's footprint. This prevents duplicate segmentations of the same cell.

The HNCcorr algorithm is indifferent to the method used to generate the list of candidate locations. This feature allows for any, possibly parallelized, procedure. Our implementation of HNCcorr uses a nearly-exhaustive enumeration method, so it is unlikely to miss any prospective cell.

The algorithm first computes, for each pixel, the average pairwise correlation to its 8 neighboring pixels. We refer to this value as the *local correlation* of pixel i:

$$lc_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} \text{corr}(x_i, x_j),$$

where $N(i)$ denotes the 8-neighborhood of pixel i and $|N(i)| = 8$. Subsequently, the pixels are partitioned into $n_{grid} \times n_{grid}$ grid blocks starting from the north-west corner. By default, $n_{grid} = 5$. These grid blocks are used for spatial aggregation. For each grid block, it selects the pixel with the highest local correlation. All other pixels are discarded. We sort the remaining pixels from high to low in terms of local correlation. The top p_{seed} percent of these pixels are selected as candidate locations. A summary of this procedure is provided in Algorithm 2. The complexity of this step is $O(N \log N)$, where N is the number of pixels in the dataset.

Algorithm 2 Method for generating candidate locations for cells.

Input: M: Movie with fluorescence vector $\mathbf{x}_i$ for pixels $i = 1, \ldots, n$
Output: List of candidate locations
Parameters:
- $n_{\text{grid}} = 5$: Size of each square grid block
- $p_{\text{seed}} = 40\%$: Percentage of candidate locations kept for processing

function GENERATELOCATIONS(M)
`// Compute local correlation`
for $i = 1, \ldots, n$
$lc_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} \text{corr}(x_i, x_j)$

`// Select pixel from each grid block with highest local correlation`
Initialize L as empty list
Split the pixels into a blocks $B_1, \ldots, B_k$ of $n_{\text{grid}} \times n_{\text{grid}}$ pixels starting at north-west pixel
for $b = 1, \ldots, k$
$i^* \leftarrow \arg\max_{i \in GB_b} lc_i$
Append i^* to L

Sort L in decreasing order of local correlation: $lc_1 \geq lc_2 \geq lc_3 \ldots$
return first $\lceil p_{\text{seed}} \times |L| \rceil$ pixels in L.

Typically, p_{seed} is set to 40 percent. This value was decided based on an empirical study. This study indicates that the benefit of increasing it above 40 percent is small for most datasets, whereas it increases running time.

This procedure for selecting candidate location is an enumeration of all possible pixels with two types of speedups: Spatial aggregation and discarding of pixels with low local correlation. These speedups can be deactivate by constructing grid blocks of 1×1 pixel ($n_{\text{grid}} = 1$) and selecting all remaining pixels ($p_{\text{seed}} = 100\%$).

5.3 Construction of the similarity graph for HNC

Given a pixel as a candidate location, the algorithm constructs a graph on an $n_{\text{patch}} \times n_{\text{patch}}$ square subset of pixels, which we call a *patch*. The purpose of the patch is to limit the number of cells and the size of the similarity graph. The patch is centered on the input pixel, the candidate location. The patch size n_{patch} should be chosen such that it contains a typical

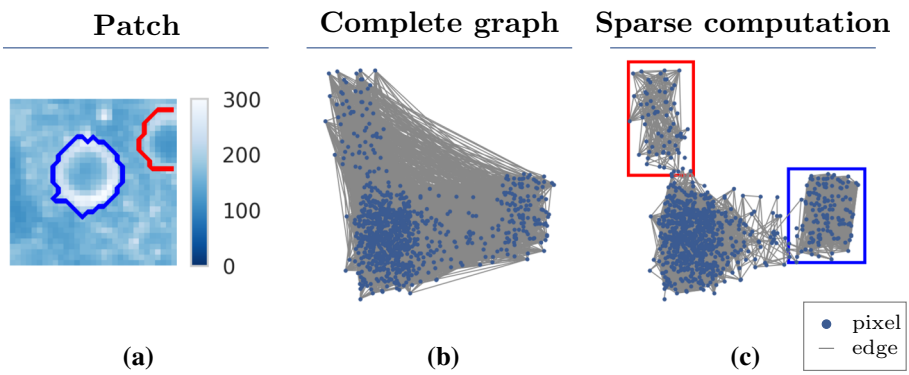


Fig. 3 Sparse computation constructs a sparse similarity graph. Comparison of a complete similarity graph and the similarity graph constructed by sparse computation for an example patch. For the purpose of illustration, the nodes are positioned based on the 2-dimensional PCA projection of the pixels' feature vectors offset by a small uniformly sampled perturbation. **a** Mean intensity image of the patch with the outline of two cells marked in red and blue. **b** Complete similarity graph with an edge between every pair of pixels. For the purpose of illustration, only 10,000 randomly sampled edges are shown. **c** Sparse similarity graph constructed by sparse computation with a three-dimensional PCA projection and a grid resolution of $\kappa = 25$. Two clusters of pixels (marked with red and blue rectangles) are identified by Sparse Computation. These two clusters match the spatial footprints of the two cells shown in **a**. (Color figure online)

cell. By default, $n_{\text{patch}} = 31$, which is significantly less than the size of a movie, typically $n_1 \times n_2 = 512 \times 512$ pixels.

The algorithm then constructs the graph $G = (V, E)$ consisting of the set of nodes, denoted by V, and the set of edges that represent the pairwise similarity relations. Each edge has an associated similarity weight $w_{ij} \geq 0$. The similarity weights are described in Sect. 5.4. The nodes represent the pixels in the patch. The edge set is determined with sparse computation (Hochbaum and Baumann 2016), see Sect. 4. We use sparse computation to reduce the number of edges. Sparse computation also helps to identify structure in the data as shown in Fig. 3.

5.4 Similarity-squared: a novel similarity measure

In HNCcorr, the similarity weight w_{ij} associated with edge $[i, j] \in E$ measures the similarity between pixels i and j. We describe here the novel similarity measure $(\text{SIM})^2$. We devised $(\text{SIM})^2$ to measure similarity between pixels in calcium imaging movies, but the concept itself is applicable in other contexts as well. $(\text{SIM})^2$ measures similarities between the similarity patterns. The name $(\text{SIM})^2$ explains that it measures second-order similarity.

5.4.1 Pairwise correlation and background pixels

A common approach to assess the similarity between two pixels i and j is to measure the correlation $\text{corr}(x_i, x_j)$ between their fluorescence signals across all frames of the movie. Correlation is effective for measuring the similarity between two pixels from the same cell, since the pixels have a correlated fluorescence pattern due to the shared signal of the cell. Another important group of pixels is the background pixels. These pixels do not belong to a cell, and they do not typically share a common pattern. Background pixels are only weakly

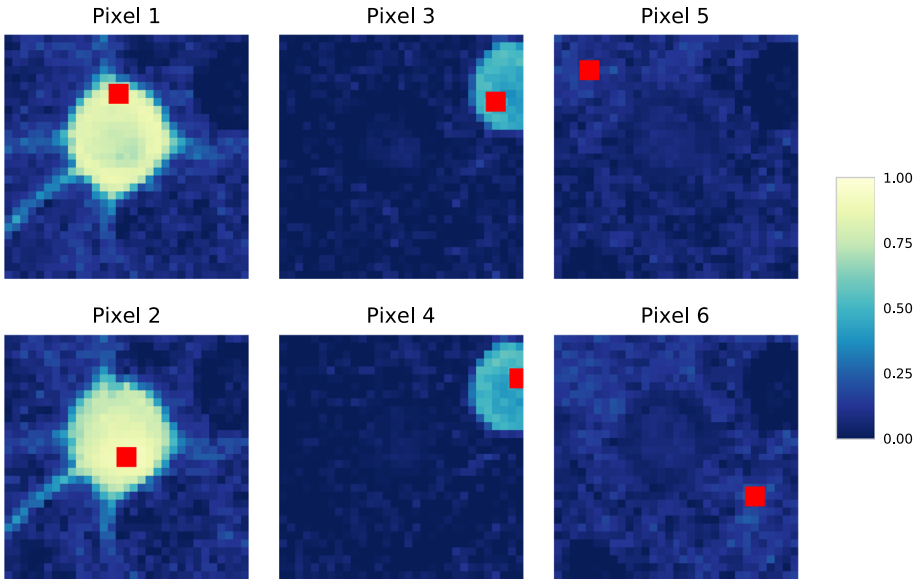


Fig. 4 Visualization of the correlation images of six pixels. The color (value) of each pixel shown in each correlation image is the pixel-to-pixel correlation between that pixel and the pixel marked in red. Lighter colors represent higher correlations, with the correlation scale truncated at zero. The correlation image is a visualization of the feature vector of a pixel. The patch is taken from the Neurofinder 02.00 training dataset and contains two cells. Pixels 1 and 2 belong to the same cell, pixels 3 and 4 belong to the same cell, and pixels 5 and 6 are background pixels. Even though the pairs of pixels 1 & 2, pixels 3 & 4, and pixels 5 & 6 are not always highly correlated, their correlation images are nearly identical. (Color figure online)

correlated to each other. As such, the similarity between these background pixels is low if the similarity is measured by correlation.

However, the lack of strong correlation with other pixels is effectively what characterizes background pixels. We use this observation to strengthen the similarity measure and aid the clustering. In other words, the background pixels are similar to each other in the pattern of being weakly correlated to every other pixel, whereas pixels that belong to a cell are highly correlated to other pixels in the same cell.

Instead of computing the similarity between pixels directly based on their correlation, we should evaluate the similarity between any two pixels based on how these two pixels correlate with all pixels. Figure 4 illustrates this. For a small patch, six pixels are marked in red and are shown with their pairwise correlation to all pixels in the patch. We refer to these images as correlation images. Two pixels in each of the two visible cells are marked as well as two background pixels. The correlation image for each pixel, e.g. pixel 1, shows the pairwise correlation between pixel 1 and every pixel in the patch, represented by the color of the pixel, with a lighter color corresponding to higher correlation. The following observations are drawn from the figure: Pixels belonging to the same cell have nearly identical correlation images (see pixels 1 & 2 and pixels 3 & 4). Furthermore, background pixels also have nearly identical correlation images even though the pixels themselves are not correlated (see pixels 5 & 6).

Similarity between background pixels can thus be captured by comparing their correlation images, whereas correlation or other standard similarity measures would not recognize this. Comparing correlation images instead of computing signal correlation also boosts the

similarity between pixels in cells with a low signal-to-noise ratio, such as the cell containing pixel 3 & 4. Whereas pixels 3 and 4 are only weakly correlated, their correlation images are nearly identical. This approach is an application of a novel technique for computing pairwise similarities that we call $(\text{SIM})^2$.

5.4.2 Similarity-squared

The idea of $(\text{SIM})^2$ is to determine the similarity between a pair of objects (here pixels), not by directly comparing their features, but rather by comparing the objects' similarities to a set of reference objects, called the *reference set*. The resulting pairwise similarities between objects can be interpreted as a similarity derived from other similarities, hence the name $(\text{SIM})^2$.

Our use of $(\text{SIM})^2$ in HNCcorr consists of a three-step procedure. In the first step, HNCcorr samples a random subset of pixels. This subset of pixels forms the reference set. Let γ denote the percentage of pixels that are sampled. If n denotes the number of pixels in the patch, then the reference set RS consists of pixels $r_1, \ldots, r_{n_R}$ for $n_R = \gamma n$. We use sampling to reduce the running time of the feature vector computation. It has a negligible effect on the cell detection performance of HNCcorr as long as γ is at least 25 percent.

In the second step, we compute the feature vector R_i of pixel i. R_i is a vector of dimension n_R. Mathematically, the kth element of the feature vector R_i of pixel $i \in V$ is defined as $R_i[k] = \text{corr}(x_i, x_{r_k})$. Assuming that $\gamma = 100\%$, the feature vector can be interpreted as a vector representation of the correlation image of pixel i or as the row of pixel i in the pixel-to-pixel correlation matrix defined over the pixels in the patch.

In the third step, the similarity weights w_{ij} are computed for every edge $[i, j] \in E$. For general $(\text{SIM})^2$, we can use any function of the feature vectors to compute the similarity. We choose here the Gaussian similarity:

$$w_{ij} = \exp\left(-\alpha \| R_i - R_j \|_2^2\right), \tag{8}$$

where α is a scaling parameter, which is typically set to 1. Note that $w_{ij} \neq \text{corr}(x_i, x_j)$. The complexity of computing all similarity weights is $O\left(\gamma n_{\text{patch}}^2 \left(m + n_{\text{patch}}^2 T\right)\right)$, where m is the number of edges in the similarity graph and T is the number of frames in the movie.

The default value of $\gamma = 32\%$ for sampling was determined based on an experimental evaluation of the cell detection performance and the running time performance. The value of 32% provides a substantial improvement in running time compared to $\gamma = 100\%$ and almost no loss in cell detection accuracy.

5.5 The selection of seeds for HNC

Recall that HNC requires a set of positive seeds S_P for the cell S and a set of negative seeds S_N for $\bar{S}$. For a given candidate pixel and the associated patch, the algorithm selects a super pixel, a square of $k \times k$ pixels, as positive seed set S_P. The super pixel is centered on the input pixel and is thus located in the center of the patch. k is typically set to 3 or selected by validation on a dataset annotated with cell locations. The super pixel forms the basis of the cell cluster S. It also ensures that HNC attempts to delineate the footprint of the current cell of interest.

In addition to the positive seed, HNC requires also a set of negative seeds S_N. These negative seeds are pixels that cannot be selected as part of the cluster in HNC. The negative

seeds ensure that not all pixels are selected. A total of n_{neg} negative seeds are selected uniformly from a circle of sufficiently large radius centered at the positive seed. The radius ρ of the circle is chosen such that any cell that contains the positive seed is inside the circle. It is typically chosen to be slightly less than $\frac{m}{2}$.

The seeds are selected in $O(k^2 + n_{neg})$ per patch.

5.6 Post-processing

As output of the HNC problem, we obtain all nested, optimal clusters $S_1^* \subset S_2^* \subset \ldots S_\ell^*$. A *cleaning* process is applied to each of the clusters: Pixels that are not contiguous to the positive seed are removed, and pixels that are fully enclosed by pixels in the candidate cluster are added to the cluster. The resulting cleaned clusters are $S_1^{*C} \subset S_2^{*C} \subset \ldots S_\ell^{*C}$.

The post-processing algorithm selects one of the cleaned candidate clusters as the cell footprint or decides that no cell was detected. We retain only the cleaned clusters whose size $\left|S_i^{*C}\right|$ is in a given, user-provided range, typically 40 to 200 pixels. In case all clusters are discarded, then the algorithm returns the statement that no cell was detected. Otherwise, each remaining cluster's size is compared to a preferred cell size that is set by the user. The candidate cluster for which the square root of the size, $\sqrt{\left|S_i^{*C}\right|}$, is closest to the square root of the expected cell size is selected as the spatial footprint of the cell. We use the square root of the cluster size since this best reflects the scaling of the area of a circle. This algorithm is summarized in Algorithm 3. More complex post-processing techniques based on convolutional neural networks have been explored as well. However, preliminary experiments showed no substantial improvements.

Algorithm 3 Size-based post-processor of candidate clusters.

Input: $S^C = \{S_1^{*C} \subset \ldots S_\ell^{*C}\}$: Set of cleaned candidate clusters, where each cluster is a set of pixels
Output: Spatial footprint of the cell (empty set if no cell detected)
Parameters:
- $n_{\min}$: Minimum number of pixels in a spatial footprint (dataset dependent)
- n_{avg}: Expected number of pixels in a spatial footprint (dataset dependent)
- $n_{\max}$: Maximum number of pixels in a spatial footprint (dataset dependent)

function POSTPROCESSOR(C)
  for $s \in S^C$
    if $|s| < n_{\min}$ or $|s| > n_{\max}$ `// Discard small/large clusters`
      $S^C \leftarrow S^C \backslash \{s\}$.
  if $S^C = \emptyset$
    return $\emptyset$
  else
    return $\arg\min_{s \in S^C} \left(\sqrt{|s|} - \sqrt{n_{\text{avg}}}\right)^2$

The complexity of this post-processing procedure is $O(n_{\text{patch}}^2 \ell)$.

5.7 Potential extensions for HNCcorr

We sketch here how HNCcorr could be extended for real-time data collection and for countering signal contamination due to overlapping cells.

5.7.1 Online, real-time data collection

Most cell identification algorithms are designed for fully recorded movies, allowing the algorithms to identify cells based on all available data. There is also an interest in online algorithms that detect cells in real-time. The first such algorithm is OnAcid, which is closely related to CNMF (Giovannucci et al. 2017). Online algorithms enable closed-loop experiments, where the environment is adapted according to previously recorded activity (Giovannucci et al. 2017). Real-time detection requires that cell identification algorithms works with streaming data, observing only a single frame at a time.

We sketch here how HNCcorr could be adapted for online, real-time cell detection. Similar to the offline implementation, we would use local correlation to identify candidate locations for cells. The main suggestion for achieving real-time processing is to implement HNCcorr in parallel, where one process is responsible for maintaining the local correlation estimates, and one or more other processes independently select and evaluate candidate locations to identify cells based on the recorded data so far. This parallel setup requires that only the process for updating the local correlation estimates runs in real-time, whereas the processes for evaluating candidate locations can be run asynchronously. The extraction of the temporal activities of identified cells as well as de-duplication of identified cells could be handled asynchronously as well.

To see how local correlation estimates could be updated in real-time, we note that a straightforward arithmetic manipulation shows that it is possible to update a pairwise correlation estimate in constant time when a new frame arrives. This makes it possible to update the local correlation estimate for a pixel in constant time. The work per new frame is thus $O(N)$, where N is the number of pixels in the movie. Based on a calculation of the required number of floating point operations per frame, we estimate that this approach should be capable of processing movies with 512×512 pixels at regular frame rates on a modern laptop or desktop computer.

5.7.2 Overlapping cells

As it stands, HNCcorr does not have the capability to demix cell signals of overlapping. However, it tends to identify the part of the spatial footprint that does not overlap with another cell, unless the positive seed happens to be located on the overlap. This makes HNCcorr robust to contamination. In order to further counter signal contamination for overlapping cells, a matrix factorization method, initialized with the footprints identified by HNCcorr, can be applied locally as post-processing.

6 Experiments

To evaluate the experimental performance of HNCcorr we compare against other leading algorithms on the Neurofinder benchmark. Specifically, we compare HNCcorr with the matrix factorization algorithms CNMF (Pnevmatikakis et al. 2016) and Suite2P (Pachitariu et al. 2017) as well as 3dCNN, a 3-dimensional convolutional neuronal network.

Table 1 Characteristics of the test datasets of the Neurofinder benchmark and their corresponding training datasets. Data reproduced from Neurofinder (CodeNeuro 2016)

Name	Source	Resolution	Length (s)	Frequency (Hz)	Brain region	Annotation method
00.00	Svoboda Lab	512 × 512	438	7.00	vS1	Anatomical markers
00.01	Svoboda Lab	512 × 512	458	7.00	vS1	Anatomical markers
01.00	Hausser Lab	512 × 512	300	7.50	v1	Human labeling
01.01	Hausser Lab	512 × 512	667	7.50	v1	Human labeling
02.00	Svoboda Lab	512 × 512	1000	8.00	vS1	Human labeling
02.01	Svoboda Lab	512 × 512	1000	8.00	vS1	Human labeling
03.00	Losonczy Lab	498 × 467	300	7.50	dHPC CA1	Human labeling
04.00	Harvey Lab	512 × 512	444	6.75	PPC	Human labeling
04.01	Harvey Lab	512 × 512	1000	3.00	PPC	Human labeling

3dCNN is an algorithm that only recently appeared on the Neurofinder benchmark. As of the submission of this paper, there is no corresponding publication nor is there publicly available code for the algorithm. We requested the code for 3dCNN but have not received a response so far. We therefore excluded the 3dCNN algorithm from the runtime analysis.

6.1 Experimental setup

6.1.1 Datasets

The Neurofinder community benchmark (CodeNeuro 2016), http://neurofinder.codeneuro.org, is an initiative of the CodeNeuro collective of neuroscientists that encourages software tool development for neuroscience research. The collective also hosts the Spikefinder benchmark, which has led to improved algorithms for spike-inference in calcium imaging data (Berens et al. 2018). The Neurofinder benchmark aims to provide a collection of datasets with ground truth labels for benchmarking the performance of cell detection algorithms.

The benchmark consists of 28 motion-corrected calcium imaging movies provided by four different labs. Datasets are annotated manually or based on anatomical markers. They differ in recording frequency, length of the movie, magnification, signal-to-noise ratio, and in the brain region that was recorded. The datasets are split into two groups, *training* datasets and *test* datasets. The eighteen training datasets are provided together with reference annotations for the cell locations, whereas the reference annotations for the nine test datasets are not disclosed.The test datasets and their undisclosed annotations are used by the Neurofinder benchmark to provide an unbiased evaluation of the performance of the algorithms. The characteristics of the test datasets are listed in Table 1. We note that some cells are not marked in the reference annotation. This does not invalidate the experimental analysis since the task of annotating cells remains equally difficult for each algorithm.

All datasets can be downloaded directly from the Neurofinder benchmark for cell identification (CodeNeuro 2016), https://neurofinder.codeneuro.org.

6.1.2 Active versus inactive cells

We group the datasets into two sub-groups: Datasets in which most annotated cells are active, i.e. have a detectable fluorescence signal other than noise, and datasets in which

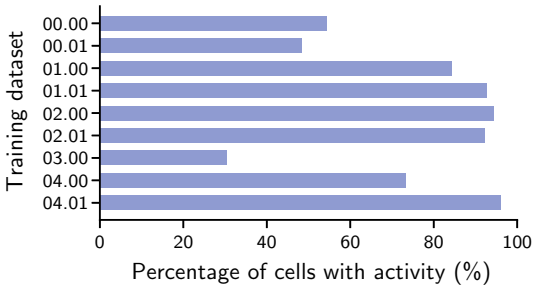


Fig. 5 Approximate percentage of active cells among annotated cells in the training datasets. To determine the activity of the cells in a movie we used the following approximate analysis. First, we downsample the movie by averaging 10 frames. For every annotated cell, we compute the average intensity over time of the pixels in the spatial footprint. This timeseries is used an estimate of the cell's signal. A cell is then considered active if the $\Delta f/f$ (Jia et al. 2011) of this timeseries is at least 3.5 standard deviations above the median of $\Delta f/f$ for a total of at least 3 time steps in the movie. Due to the approximate nature of this analysis, its interpretation should be limited to understanding the general ratio between active and inactive cells in the datasets

most annotated cells are inactive. We make this distinction because inactive cells cannot be found by CNMF, Suite2P, and HNCcorr due to the lack of measurable changes in fluorescence other than noise. These cells are detectable with shape-based detection algorithms—e.g. (Gao 2016), when the cell's fluorescence differs from the background.

The datasets with inactive cells are: 00.00, 00.01, and 03.00. The presence of inactive cells in these datasets has been noted in previous work (Pachitariu et al. 2017) as well as in a discussion of the benchmark's reference annotation on GitHub.[1] Experimentally, we also observed that the percentage of annotated cells that are active is substantially lower for training datasets 00.00, 00.01, and 03.00 as shown in Fig. 5. Results for the test datasets could not be evaluated due to the lack of a reference annotation but similar results are expected.

6.1.3 Evaluation metrics

The list of cells identified by each of the algorithms is compared with the reference annotation. For this comparison, we use the submissions on the Neurofinder website. Each algorithm's submission was submitted by the algorithm's authors. This ensures that the results are representative of the algorithm's performance. Furthermore, the evaluation is fair since none of the authors had access to the reference annotation.

The algorithms are scored based on their ability to reproduce the reference annotation using standard metrics from machine learning. Each cell in the reference annotation is counted as a true positive (TP) if it also appears in the algorithm's annotation. The cells in the reference annotation that are not identified by the algorithm are counted as false negatives (FN), and the cells identified by the algorithm but that do not appear in the reference annotation are counted as false positives (FP). Each algorithm's performance is scored on a dataset based on $recall = \frac{TP}{TP+FN}$ and $precision = \frac{TP}{TP+FP}$. The overall performance of the algorithm on a dataset is summarized by the *F1-score*, which is the harmonic mean of recall and precision. For all metrics, higher scores are better.

[1] See e.g. issues 16 and 24 on https://github.com/codeneuro/neurofinder.

Table 2 Dataset dependent parameter values used for the HNCcorr implementation

Dataset	Patch size (m)	Radius circle negative seeds (ρ)	Size superpixel (k)	Post-processor lower bound ($n_{\min}$)	Post-processor upper bound ($n_{\max}$)	Post-processor expected size (n_{avg})
00.00	31×31	10 pixels	5×5	40 pixels	150 pixels	60 pixels
00.01	31×31	10 pixels	5×5	40 pixels	150 pixels	65 pixels
01.00	41×41	14 pixels	5×5	40 pixels	380 pixels	170 pixels
01.01	41×41	14 pixels	5×5	40 pixels	380 pixels	170 pixels
02.00	31×31	10 pixels	1×1	40 pixels	200 pixels	80 pixels
02.01	31×31	10 pixels	1×1	40 pixels	200 pixels	80 pixels
03.00	41×41	14 pixels	5×5	40 pixels	300 pixels	120 pixels
04.00	31×31	10 pixels	3×3	50 pixels	190 pixels	90 pixels
04.01	41×41	14 pixels	3×3	50 pixels	370 pixels	140 pixels

A cell in the ground truth annotation is identified if the center of mass of the closest cell in the algorithm's annotation is within 5 pixels. These two cells are then matched and can not be matched with another cell. Each cell in either annotation is thus matched at most once.

6.1.4 Code accessibility

The software used to generate the results in this work is available for non-commercial use at https://github.com/quic0/HNCcorr. The code is also available as supplementary material.

6.1.5 Computational hardware

The experiments were run with MATLAB 2016a on a single core of a Linux machine running Linux Mint 18.1. The machine is equipped with an Intel i7-6700HQ CPU running at 2.60 GHz and 16GB RAM.

6.1.6 Algorithm implementations

The results in Figs. 6 and 8 are taken directly from Neurofinder (CodeNeuro 2016) and were produced by the authors of the respective algorithms. The results in Fig. 9 were generated by us. In the remainder, we describe the HNCcorr implementation used for all experiments including those reported in Figs. 6 and 8. We also describe the CNMF and Suite2P implementation used for the experiment reported in Fig. 9.

HNCcorr All datasets for HNCcorr were preprocessed by averaging every ten frames into a single frame to reduce the noise. All parameters were kept at their default settings with the exception of dataset specific parameters listed in Table 2. These parameters are dataset dependent due to varying cell sizes across datasets. For the experiments reported in Figs. 6 and 8, we used non-default values for the reference set sampling rate ($\gamma = 100\%$) and the grid resolution used for sparse computation ($\kappa = 25$). These changes have a marginal effect on the cell detection quality of the algorithm, but they result in increased running times.

CNMF The CNMF implementation was obtained from https://github.com/epnev/ca_source_extraction. The base configuration was taken from the file `run_pipeline.m` in the CNMF

repository. We turned off the patch-based processing, and set the number of cells, as denoted by K, equal to 600. We used the same values for the maximum and minimum cell size, `max_size_thr` and `min_size_thr`, as in HNCcorr. We also set the temporal down-sampling `tsub` to 10 to match the down-sampling used with HNCcorr.

Suite2P The Suite2P implementation was obtained from https://github.com/cortex-lab/Suite2P. The base configuration was taken from the file `master_file_example.m` in the Suite2P repository. The `diameter` parameter was tweaked per dataset to maximize the F1-score on the Neurofinder training datasets. The selected values per (dataset) are: 8 (00.00), 10 (00.01), 13 (01.00), 13 (01.01), 11 (02.00), 11 (02.01), 12 (03.00), 11 (04.00), and 12 (04.01).

6.2 Cell detection performance

The experimental performance of the algorithms HNCcorr, 3dCNN, CNMF, and Suite2P on the six test datasets containing active cells is shown in Fig. 6. The cells identified by

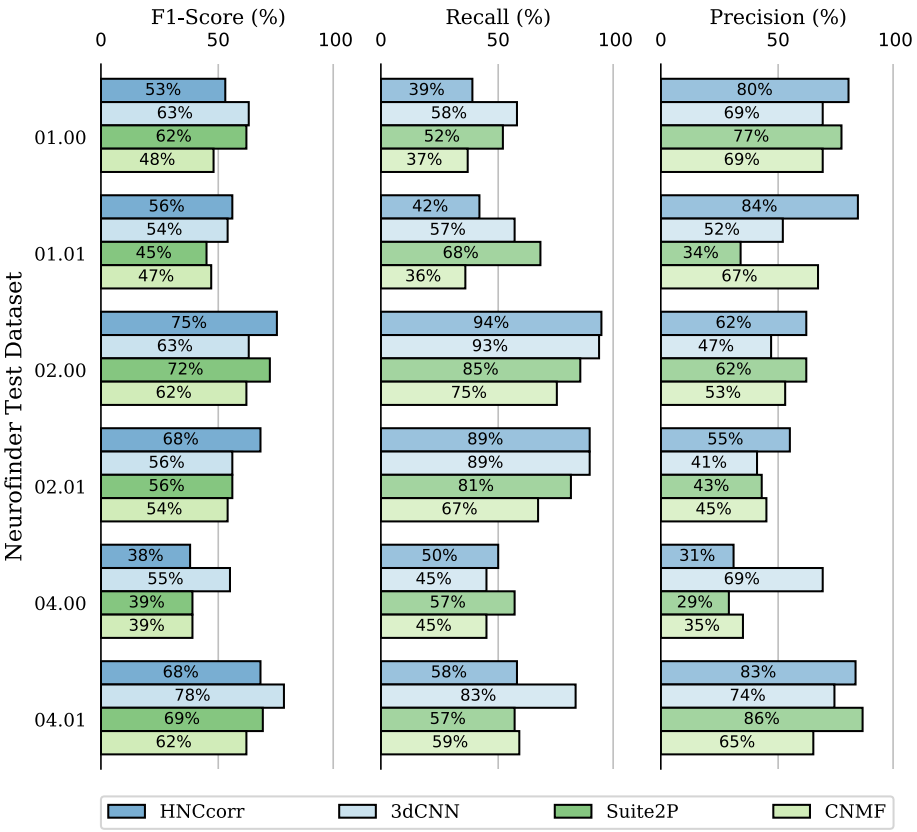
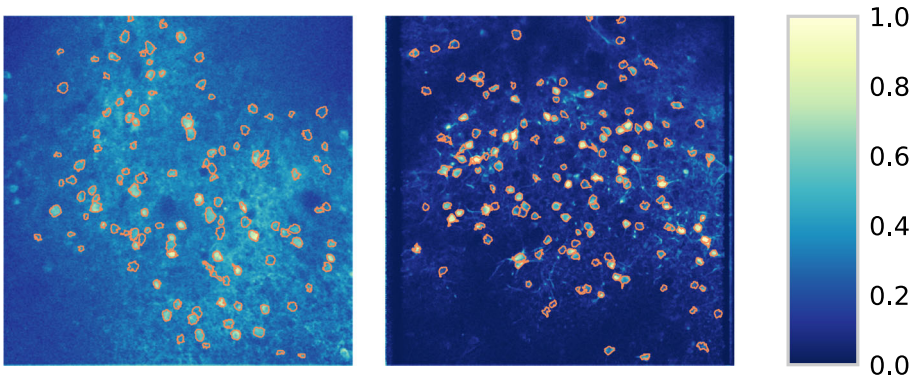


Fig. 6 Cell identification scores for the HNCcorr, CNMF, and Suite2P algorithms on the Neurofinder test datasets with active cells. For each of the listed metrics, higher scores are better. The data is taken from Neurofinder submissions `Sourcery` by `Suite2P`, `CNMF_PYTHON` by `CNMF`, `3dCNN` by `ssz`, and submission `HNCcorr` by `HNCcorr`



(a) Neurofinder 01.01 test dataset. **(b)** Neurofinder 02.00 test dataset.

Fig. 7 Contours of the cells identified by HNCcorr overlaid on the respective local correlation image

HNCcorr are shown for two datasets in Fig. 7. Overall, the HNCcorr algorithm has superior performance across datasets compared to the matrix factorization algorithms. The HNCcorr algorithm achieves a 15 percent relative improvement compared to CNMF in terms of average F1-score across datasets. It also attains a minor improvement of 4 percent compared to Suite2P. However, it performs about 3 percent worse than 3dCNN, which detects both active and inactive cells unlike HNCcorr, Suite2P, and CMNF.

HNCcorr performs particularly well on datasets 02.00 and 02.01 where it attains substantially higher F1-scores than the other algorithms, due to higher detection capability as measured by recall. Although 3dCNN is able to match the near-perfect recall of HNCcorr, it attains lower precision for datasets 02.00, 02.01. This indicates that 3dCNN detects either cells that are not in the reference annotation or incorrectly marks non-cell regions as cells.

6.3 Leading neurofinder submissions

HNCcorr and matrix factorization algorithms identify cells based on their unique fluorescence signal. As such, they are able to detect cells that activate, i.e. have one or more spikes in calcium concentration, but they cannot detect cells without any signal. As discussed, Neurofinder datasets 00.00, 00.01, and 03.00 have a large number of inactive cells. Therefore, matrix factorization algorithms and HNCcorr only detect a small percentage of cells in these datasets.

The leading Neurofinder submission for both Suite2P and HNCcorr therefore rely on a shape-based detection algorithm for these datasets. The HNCcorr + Conv2d submission uses the Conv2d (Gao 2016) neural network for datasets 00.00, 00.01, and 03.00 and HNCcorr for the remaining datasets. Similarly, Suite2P + Donuts submission uses Donuts (Pachitariu et al. 2013) for the datasets 00.00, 00.01, and 03.00 and Suite2P for the remaining datasets. Together with the 3dCNN, these submissions are the top three submissions for the Neurofinder benchmark. Figure 8 shows how the three submissions compare. In particular, the 3dCNN algorithm outperforms the other shape-based detection algorithms on datasets 00.00, 00.01, and 03.00. As discussed before, HNCcorr attains higher F1-scores for the 02.00 and 02.01 datasets.

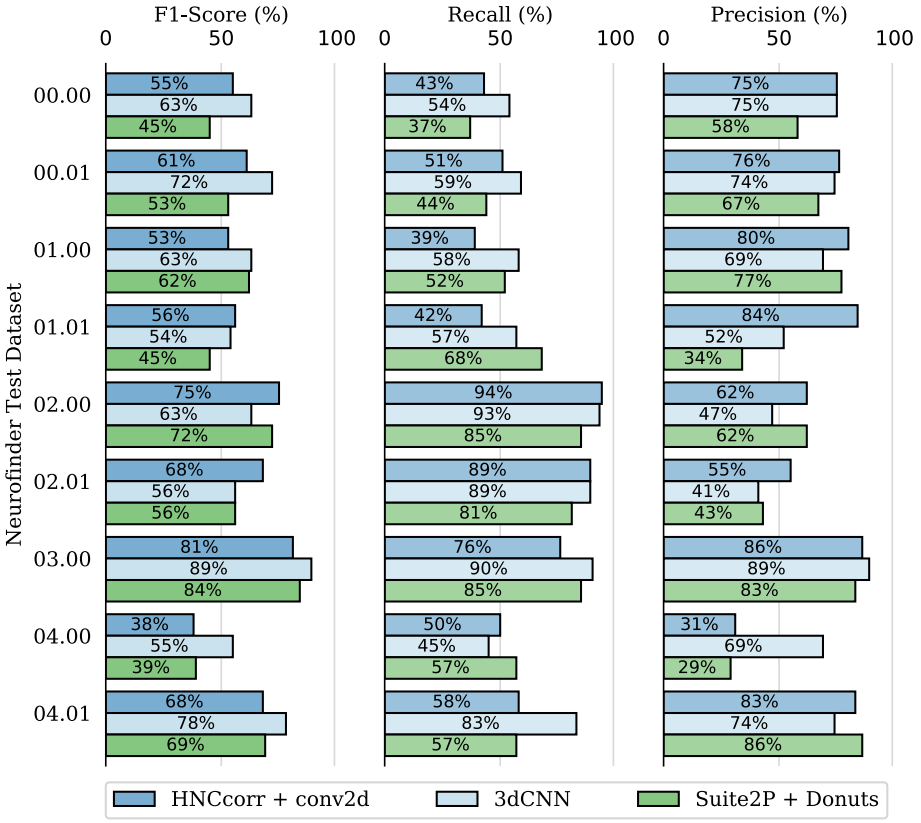


Fig. 8 Cell identification scores on all test datasets for the three leading submissions of the Neurofinder benchmark in July 2018. For each of the listed metrics, higher scores are better. The 3dCNN entry is based on the Neurofinder submission `3dCNN` by `ssz`. The Suite2P + Donuts (Pachitariu et al. 2013) entry is taken from the submission `Sourcery` by `Suite2P`. It uses the Donuts algorithm for datasets 00.00, 00.01, and 03.00 and the Suite2P algorithm for the remaining datasets. The HNCcorr + Conv2d entry is taken from the submission `HNCcorr + conv2d` by `HNCcorr`. It uses the Conv2d algorithm (Gao 2016) for datasets 00.00, 00.01, and 03.00 and the HNCcorr algorithm for the remaining datasets. The results obtained with the Conv2d algorithm reported here differ slightly from the `Conv2d` submission on the Neurofinder benchmark since the Conv2d model was retrained by the authors of this paper

6.4 Runtime analysis

We compared the running time performance of HNCcorr, CNMF, and Suite2P on nine training datasets of the Neurofinder benchmark. 3dCNN was excluded since the underlying code is not available. Running time results are given in Fig. 9. The measured time for CNMF and Suite2P also includes the time to provide the associated cell signals, since they are determined simultaneously, whereas for HNCcorr it does not. On average, HNCcorr is 1.5 times faster than CNMF. Compared to Suite2P, HNCcorr performs equally well on average. We observed similar performance for a large experimental dataset consisting of 50,000 frames not reported here.

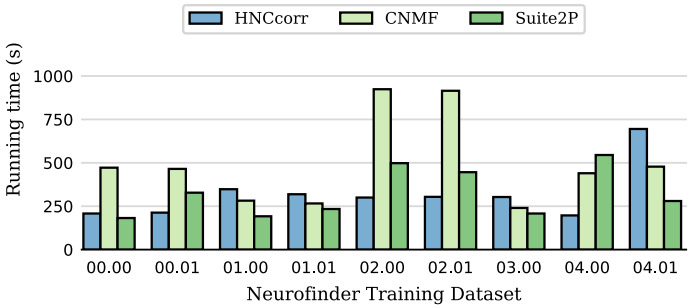


Fig. 9 Running time results for nine training datasets of the Neurofinder benchmark. The measured time for CNMF and Suite2P also includes the time to provide the associated cell signals, since they are determined simultaneously, whereas for HNCcorr it does not. Running times are based on a single evaluation

7 Conclusions

We present here the HNCcorr algorithm for cell detection in two-photon calcium imaging datasets. The HNCcorr algorithm builds upon the combinatorial clustering problem HNC and makes use of the newly devised similarity measure $(\text{SIM})^2$. Experimentally, we demonstrate that HNCcorr is a top performer for the Neurofinder benchmark both in terms of cell detection quality and running time. HNCcorr achieves a higher average score than the matrix factorization algorithms CNMF and Suite2P with similar or faster running time. This work demonstrates that combinatorial optimization is a valuable tool for analysis in neuroscience or other disciplines.

In future work, HNCcorr could be extended to support real-time data, which enables direct feedback experimentation. Another extension of interest is adapting HNCcorr for one-photon and light-field calcium imaging, where the main challenge is dealing with overlapping cells.

Acknowledgements The fund was supported by Division of Civil, Mechanical and Manufacturing Innovation (Grant No. 1760102).

References

Apthorpe, N., Riordan, A., Aguilar, R., Homann, J., Gu, Y., Tank, D., & Seung, H.S. (2016). Automatic neuron detection in calcium imaging data using convolutional networks. In *Advances in neural information processing systems* (pp. 3270–3278)

Baumann, P., Hochbaum, D.S., & Spaen, Q. (2016). Sparse-reduced computation: Enabling mining of massively-large data sets. In *Proceedings of the 5th international conference on pattern recognition applications and methods, SCITEPRESS*, Rome, Italy (pp. 224–231)

Baumann, P., Hochbaum, D.S., & Spaen, Q. (2017). High-performance geometric algorithms for sparse computation in big data analytics. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 546–555)

Baumann, P., Hochbaum, D., & Yang, Y. (2019). A comparative study of the leading machine learning techniques and two new optimization algorithms. *European Journal of Operational Research*, *272*(3), 1041–1057.

Berens, P., Freeman, J., Deneux, T., Chenkov, N., McColgan, T., Speiser, A., et al. (2018). Community-based benchmarking improves spike rate inference from two-photon calcium imaging data. *PLoS Computational Biology*, *14*(5), e1006157.

CodeNeuro (2016). The neurofinder challenge. http://neurofinder.codeneuro.org/. Accessed June 01, 2018

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Dembczyński, K., Kotłowski, W., & Słowiński, R. (2009). Learning rule ensembles for ordinal classification with monotonicity constraints. *Fundamenta Informaticae*, *94*(2), 163–178.
Diego-Andilla, F., & Hamprecht, F.A. (2014). Sparse space-time deconvolution for calcium image analysis. In *Advances in neural information processing systems* (pp. 64–72)
Drineas, P., Kannan, R., & Mahoney, M. W. (2006). Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, *36*(1), 158–183.
Driscoll, L. N., Pettit, N. L., Minderer, M., Chettih, S. N., & Harvey, C. D. (2017). Dynamic reorganization of neuronal activity patterns in parietal cortex. *Cell*, *170*(5), 986–999.
Fishbain, B., Hochbaum, D.S., & Yang, Y.T. (2013). Real-time robust target tracking in videos via graph-cuts. In *Real-time image and video processing 2013, international society for optics and photonics*, (Vol. 8656, p. 865602)
Frey, P. W., & Slate, D. J. (1991). Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, *6*(2), 161–182.
Gallo, G., Grigoriadis, M. D., & Tarjan, R. E. (1989). A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, *18*(1), 30–55.
Gao, S. (2016). Conv2d: Convolutional neural network. https://github.com/iamshang1/Projects/tree/master/Advanced_ML/Neuron_Detection. Accessed June 01, 2018
Giovannucci, A., Friedrich, J., Kaufman, M., Churchland, A., Chklovskii, D., Paninski, L., & Pnevmatikakis, E.A. (2017) Onacid: Online analysis of calcium imaging data in real time. In *Advances in neural information processing systems* (pp. 2381–2391)
Goldberg, A. V., & Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *J ACM*, *35*(4), 921–940.
Grewe, B. F., Langer, D., Kasper, H., Kampa, B. M., & Helmchen, F. (2010). High-speed in vivo calcium imaging reveals neuronal network activity with near-millisecond precision. *Nature Methods*, *7*(5), 399–405.
Hochbaum, D. S. (2002). Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations. *European Journal of Operational Research*, *140*(2), 291–321.
Hochbaum, D. S. (2008). The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations Research*, *56*(4), 992–1009.
Hochbaum, D. S. (2010). Polynomial Time Algorithms for Ratio Regions and a Variant of Normalized Cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(5), 889–898.
Hochbaum, D. S. (2013). A polynomial time algorithm for rayleigh ratio on discrete variables: Replacing spectral techniques for expander ratio, normalized cut, and cheeger constant. *Operations Research*, *61*(1), 184–198.
Hochbaum, D. S., & Baumann, P. (2016). Sparse computation for large-scale data mining. *IEEE Transactions on Big Data*, *2*(2), 151–174.
Hochbaum, D. S., & Fishbain, B. (2011). Nuclear threat detection with mobile distributed sensor networks. *Annals of Operations Research*, *187*(1), 45–63.
Hochbaum, D. S., Hsu, C. N., & Yang, Y. T. (2012). Ranking of multidimensional drug profiling data by fractional-adjusted bi-partitional scores. *Bioinformatics*, *28*(12), i106–i114.
Hochbaum, D. S., Lyu, C., & Bertelli, E. (2013). Evaluating performance of image segmentation criteria and techniques. *EURO Journal on Computational Optimization*, *1*(1), 155–180.
Jewell, S., & Witten, D. (2018). Exact spike train inference via ℓ_0 optimization. *The Annals of Applied Statistics*, *12*(4), 2457–2482.
Jia, H., Rochefort, N. L., Chen, X., & Konnerth, A. (2011). In vivo two-photon imaging of sensory-evoked dendritic calcium signals in cortical neurons. *Nature Protocols*, *6*(1), 28.
Kaifosh, P., Zaremba, J. D., Danielson, N. B., & Losonczy, A. (2014). SIMA: Python software for analysis of dynamic fluorescence imaging data. *Frontiers in Neuroinformatics*, *8*, 40.
Klibisz, A., Rose, D., Eicholtz, M., Blundon, J., & Zakharenko, S. (2017). Fast, simple calcium imaging segmentation with fully convolutional networks. In M. J. Cardoso, T. Arbel, G. Carneiro, T. Syeda-Mahmood, J. M. R. Tavares, M. Moradi, A. Bradley, H. Greenspan, J. P. Papa, A. Madabhushi, J. C. Nascimento, J. S. Cardoso, V. Belagiannis, & Z. Lu (Eds.), *Deep learning in medical image analysis and multimodal learning for clinical decision support. lecture notes in computer science* (pp. 285–293). Berlin: Springer.
Levin-Schwartz, Y., Sparta, D. R., Cheer, J. F., & Adalı, T. (2017). Parameter-free automated extraction of neuronal signals from calcium imaging data. *IEEE international conference on acoustics. Speech and signal processing* (pp. 1033–1037). IEEE
Maruyama, R., Maeda, K., Moroda, H., Kato, I., Inoue, M., Miyakawa, H., et al. (2014). Detecting cells using non-negative matrix factorization on calcium imaging data. *Neural Networks*, *55*, 11–19.

Mukamel, E. A., Nimmerjahn, A., & Schnitzer, M. J. (2009). Automated analysis of cellular signals from large-scale calcium imaging data. *Neuron*, *63*(6), 747–760.

Pachitariu, M., Packer, A.M., Pettit, N., Dalgleish, H., Hausser, M., & Sahani, M. (2013). Extracting regions of interest from biological images with convolutional sparse block coding. In *Advances in neural information processing systems* (pp 1745–1753)

Pachitariu, M., Stringer, C., Dipoppa, M., Schröder, S., Rossi, L.F., Dalgleish, H., Carandini, M., & Harris, K.D. (2017). Suite2p: beyond 10,000 neurons with standard two-photon microscopy. bioRxiv p 061507

Pnevmatikaki, E.A., & Paninski, L. (2013). Sparse nonnegative deconvolution for compressive calcium imaging: Algorithms and phase transitions. In *Advances in neural information processing systems* (pp. 1250–1258)

Pnevmatikakis, E.A., Merel, J., Pakman, A., & Paninski, L. (2013). Bayesian spike inference from calcium imaging data. In *Asilomar conference on signals, systems and computers* (pp. 349–353)

Pnevmatikakis, E. A., Soudry, D., Gao, Y., Machado, T. A., Merel, J., Pfau, D., et al. (2016). Simultaneous denoising, deconvolution, and demixing of calcium imaging data. *Neuron*, *89*(2), 285–299.

Resendez, S. L., Jennings, J. H., Ung, R. L., Namboodiri, V. M. K., Zhou, Z. C., Otis, J. M., et al. (2016). Visualization of cortical, subcortical, and deep brain neural circuit dynamics during naturalistic mammalian behavior with head-mounted microscopes and chronically implanted lenses. *Nature Protocols*, *11*(3), 566.

Ryu, Y. U., Chandrasekaran, R., & Jacob, V. (2004). Prognosis using an isotonic prediction technique. *Management Science*, *50*(6), 777–785.

Sharon, E., Galun, M., Sharon, D., Basri, R., & Brandt, A. (2006). Hierarchy and adaptivity in segmenting visual scenes. *Nature*, *442*(7104), 810–813.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905.

Stosiek, C., Garaschuk, O., Holthoff, K., & Konnerth, A. (2003). In vivo two-photon calcium imaging of neuronal networks. *Proceedings of the National Academy of Sciences*, *100*(12), 7319–7324.

Theis, L., Berens, P., Froudarakis, E., Reimer, J., Rosón, M. R., Baden, T., et al. (2016). Benchmarking spike rate inference in population calcium imaging. *Neuron*, *90*(3), 471–482.

Vogelstein, J. T., Packer, A. M., Machado, T. A., Sippy, T., Babadi, B., Yuste, R., et al. (2010). Fast nonnegative deconvolution for spike train inference from population calcium imaging. *Journal of Neurophysiology*, *104*(6), 3691–3704.

Yang, Y. T., Fishbain, B., Hochbaum, D. S., Norman, E. B., & Swanberg, E. (2013). The supervised normalized cut method for detecting, classifying, and identifying special nuclear materials. *INFORMS Journal on Computing*, *26*(1), 45–58.

Zhu, X. R., Yoo, S., Jursinic, P. A., Grimm, D. F., Lopez, F., Rownd, J. J., et al. (2003). Characteristics of sensitometric curves of radiographic films. *Medical Physics*, *30*(5), 912–919.