Protocol for dissecting cascade computational components in neural networks of a visual system

Jia, Shanshan; Liu, Jian K.; Yu, Zhaofei

DOI:
10.1016/j.xpro.2023.102722

Document Version
Publisher's PDF, also known as Version of record

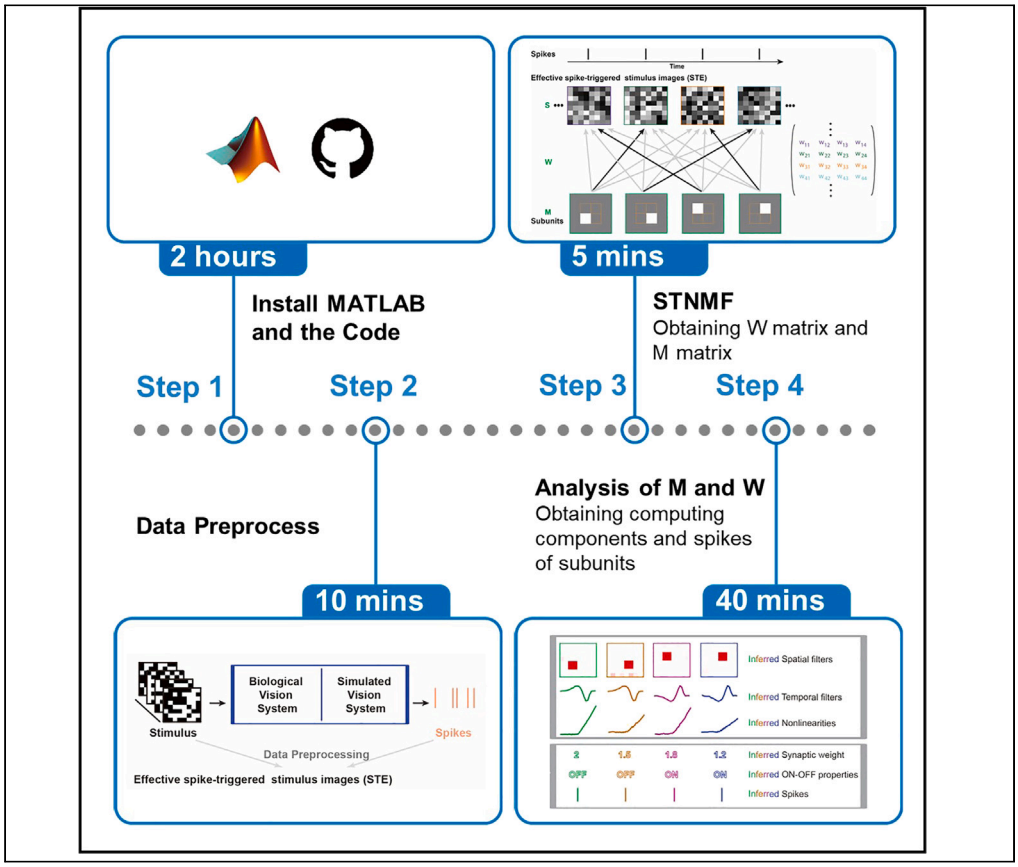
Citation for published version (Harvard):
Jia, S, Liu, JK & Yu, Z 2023, 'Protocol for dissecting cascade computational components in neural networks of a visual system', *STAR Protocols*, vol. 4, no. 4, 102722. https://doi.org/10.1016/j.xpro.2023.102722

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol for dissecting cascade computational components in neural networks of a visual system



Finding the complete functional circuits of neurons is a challenging problem in brain research. Here, we present a protocol, based on visual stimuli and spikes, for obtaining the complete circuit of recorded neurons using spike-triggered nonnegative matrix factorization. We describe steps for data preprocessing, inferring the spatial receptive field of the subunits, and analyzing the module matrix. This approach identifies computational components of the feedforward network of retinal ganglion cells and dissects the network structure based on natural image stimuli.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Shanshan Jia, Jian K. Liu, Zhaofei Yu

yuzf12@pku.edu.cn

Highlights

Dissecting cascaded computational components of synaptic circuits through STNMF

Extracting spike activity of presynaptic neurons via STNMF

Quantitative evaluation based on inferred and simulated neural activity

Three data simulation methods for visual neurons are provided

Jia et al., STAR Protocols 4, 102722
December 15, 2023
https://doi.org/10.1016/j.xpro.2023.102722

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol for dissecting cascade computational components in neural networks of a visual system

Shanshan Jia,[1,2,3,5] Jian K. Liu,[4] and Zhaofei Yu[1,2,3,6,*]

[1]School of Computer Science, Peking University, Beijing 100871, China

[2]Institute for Artificial Intelligence, Peking University, Beijing 100871, China

[3]Department of Computer Science and Technology, Peking University, Beijing 100871, China

[4]School of Computing, University of Leeds, Leeds, UK

[5]Technical contact: jiashsh@stu.pku.edu.cn

[6]Lead contact

*Correspondence: yuzf12@pku.edu.cn
https://doi.org/10.1016/j.xpro.2023.102722

SUMMARY

Finding the complete functional circuits of neurons is a challenging problem in brain research. Here, we present a protocol, based on visual stimuli and spikes, for obtaining the complete circuit of recorded neurons using spike-triggered nonnegative matrix factorization. We describe steps for data preprocessing, inferring the spatial receptive field of the subunits, and analyzing the module matrix. This approach identifies computational components of the feedforward network of retinal ganglion cells and dissects the network structure based on natural image stimuli.

For complete details on the use and execution of this protocol, please refer to Jia et al. (2021).[1]

BEFORE YOU BEGIN

The following protocol describes a computing method (spike-triggered non-negative matrix factorization, STNMF) for dissecting computing components in a visual system. STNMF typically infers the computational components of a neuron's presynaptic network based on spike activity and visual stimulus frames. In the general case, we use white noise black-and-white checkerboard stimuli. If it is difficult to induce a neuronal response in the target brain region with white noise stimulation, natural image stimulation is also applicable. Before starting, one needs to make the following preparations.

1. MATLAB software with a version higher than 2016. If one does not have MATLAB installed, one can test it using the MATLAB runtime version. Please refer to "Optional" below each code block for details.
2. Collecting Data. Experimenters can use physiological data or generate simulated data. For the collection of physiological data, white noise was used to stimulate and electrodes were used to record the spike response of neurons in the visual system.[2–4] In terms of generating simulation data, this protocol provides three simulation methods which are based on the linear nonlinear cascade model (LNLN model) to simulate retinal ganglion cells, and based on the multi-layer spike neural network model to simulate simple cells and complex cells in the primary visual cortex.

Collecting data

Physiological data

Timing: 1 min

To effectively utilize this protocol, users need to organize their data the appropriate format acceptable for STNMF.

3. Here is a step-by-step guide on how to prepare the physiological data:
   a. Reshape each stimulus image into a 1×P sequence, which is then normalized to a range of −0.5 to 0.5.
   b. Assemble all stimulus images into a P×N stimulus sequence matrix.
   c. Set the spike sampling rate to be the same as the stimulus sampling rate and generate an N×1 spike trains.
   d. Save CB, spike, nx, ny as Data.mat.

Note: The appropriate format acceptable for STNMF should include at least the stimulus image sequence matrix CB, the spike sequence of recorded neurons, and the stimulus image pixel resolution [nx, ny]. Here, nx denotes the number of rows of stimulating pixels in the image while 'ny' represents the number of columns. Additionally, N signifies the total number of stimulus images, and P corresponds to the number of pixels in each stimulus frame (P = nx × ny).

Simulation data 1: Simulation of ganglion cell response based on the LNLN model

Timing: 1 s

The simulated data of retinal ganglion cells based on the LNLN model can be obtained here.

4. Generate 8 × 8 white noise black-white checkerboard stimulation.
   a. Generate a 0–1 sequence based on the Gaussian distribution.
   b. Reshape every 64 digits to a stimulating image of 8 × 8 pixels.
5. Build a two-layer network model.
   a. Each subunit in the first layer uses spatial and temporal filters to convolution the stimulus image.
   b. Use a threshold linear nonlinear function[5] to rectify the results of step a.
   c. The neurons in the second layer receive the output of each subunit through weighted connections.
   d. Apply another threshold linear nonlinearity with a higher positive threshold to rectify the results of step c.
   e. Finally, generate the spike train by a Poisson process.[4]

```
>Weight = [1 1 1 1];

>ModelSubRF_2layers_LNLN_GC(Weight);
```

Note: The setup of this model is equivalent to a neural network with two layers (synaptic transmission membrane potential).[5] Figure 1A, right, showcases some examples of stimulus images, which are black and white checkerboard stimulus with a resolution of 8 × 8. The input layer consists of four subunits, while the output layer comprises a single neuron. The first layer's subunits are characterized by OFF polarity and each possesses a 2 × 2 pixel static spatial filter as well as a temporal filter. The weight denotes the connection strength between neurons in the first and second layers. It is worth noting that experimenters have the flexibility to modify this parameter.

Optional: MATLAB runtime version, GC_2layers_LNLN.exe.

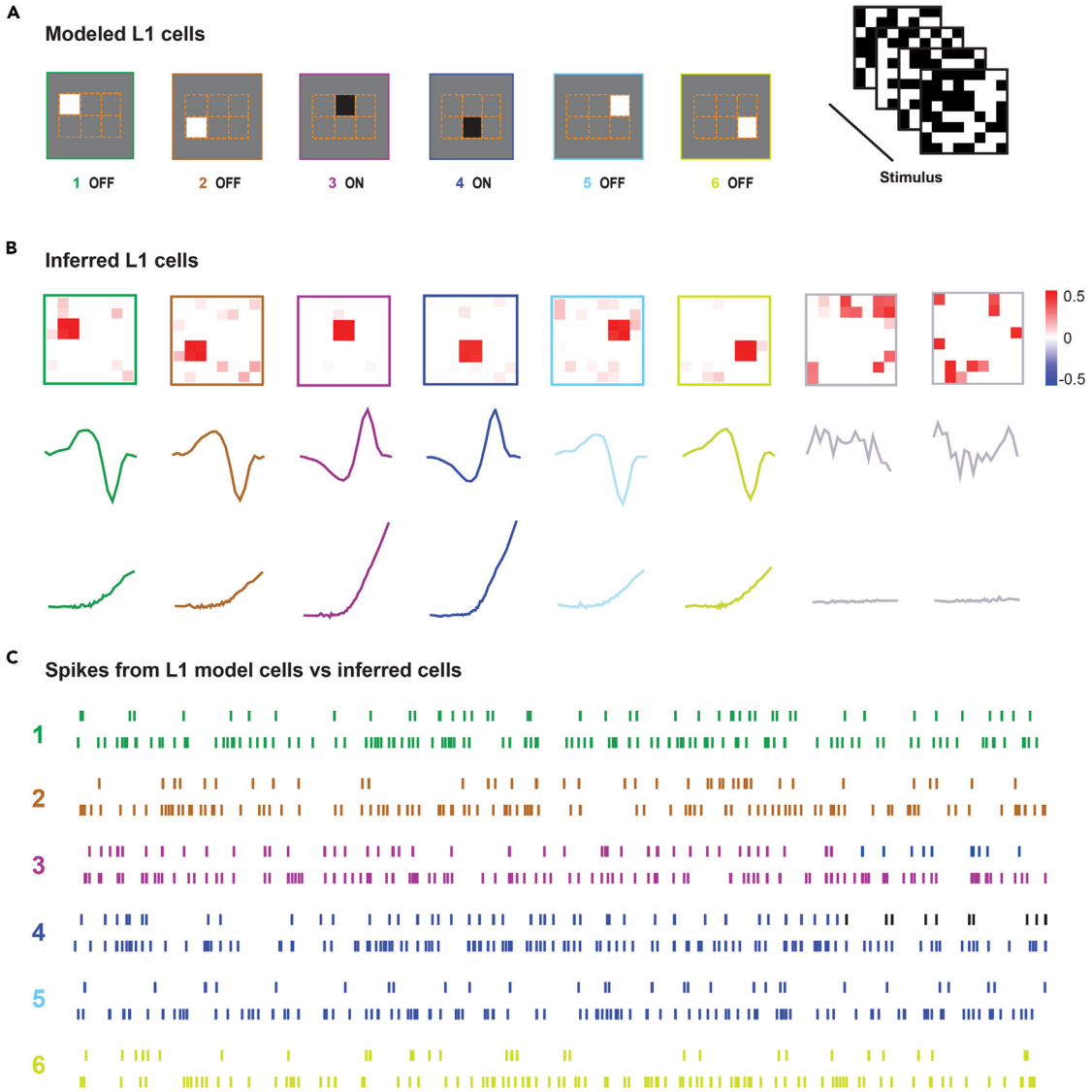


Figure 1. STNMF analysis of a simulated V1 simple cell

(A) Left, modeled L1 cells. Right, stimulus sequence.

(B) Inferred spatial receptive fields, temporal filters, and nonlinearities of L1 cells.

(C) Spikes from L1 model cells vs. inferred cells.

Revised according to Jia et al. (2021).[1]

Simulation data 2: Simple cells simulated by spiking neural networks

Timing: 2 min

The simulated data of V1 simple cells based on the multi-layer spike neural network model can be obtained here.

6. Generate 8 × 8 white noise black-white checkerboard stimuli.
   a. Generate a 0–1 sequence based on the Gaussian distribution.
   b. Reshape every 64 digits to a stimulating image of 8 × 8 pixels.
7. Build a three-layer network model.
   a. Apply the linear nonlinear spiking (LNP)[6] model to simulate neurons in the first layer.

b. By employing the Leaky Integrate-and-Fire (LIF) model,[7] we simulated neurons in the second layer.
c. The third layer comprises a neuron that effectively assimilates spikes originating from both neurons in the second layer, employing the LIF model for integration.

```
>Weight = [1 1 1 1];

>Weight2=[1 1];

>V_reset = -0.075;

>V_e = -0.07;

>V_th = -0.04;

>ModelSubRF_3layers_LIF_SNN_V1simple(Weight,Weight2, V_reset,V_e,V_th);
```

Note: The configuration of this model corresponds to a three-layer neural network (synaptic transmission of spikes).[7] The input layer consists of six neurons, the hidden layer contains two neurons, and the output layer comprises one neuron. The LNP neuron is characterized by a spatial extent of 2 × 2 pixels, an OFF-type temporal filter, and nonlinearity. Additionally, spike generation follows a Poisson process. In the second layer, there are two neurons: L2-1 receives inputs from neurons 1–4 in the first layer, while L2-2 receives inputs from neurons 3–6. Weight and Weight2 represent connection weights between the first and second layers as well as between the second and third layers, respectively. For simplicity purposes, synapse weight remains fixed at 1. V_e denotes resting potential while V_th represents threshold potential and V_reset signifies reset potential within the context of the LIF model. The generated data can be seen in Table 1.

Optional: MATLAB runtime version, V1simple_3layers_LIF_SNN.exe.

Simulated data 3: Complex cells simulated by spiking neural networks

Timing: 10 min

The simulated data of V1 complex cells based on the multi-layer spike neural network model can be obtained here.

8. Generate 8 × 8 white noise black-white checkerboard stimuli.
   a. Generate a 0–1 sequence based on the Gaussian distribution.
   b. Reshape every 64 digits to a stimulating image of 8 × 8 pixels.
9. Build a three-layer network model.
   a. Apply the LNP model to simulate neurons in the first layer.
   b. By employing the LIF model, we simulated neurons in the second layer.
   c. The third layer is a neuron that receives spikes from both neurons in the second layer and integrates the spikes based on the LIF model.

```
>Weight = [1 1 1 1];

>Weight2=[1 1];

>V_reset = -0.075;

>V_e = -0.07;

>V_th = -0.04;

>ModelSubRF_3layers_LIF_SNN_V1complex(Weight,Weight2, V_reset,V_e,V_th);
```

Note: The setting of this model is equivalent to a neural network with three layers (synaptic transmission of spikes). There are eight neurons in the first layer, two neurons in the hidden layer, and one neuron in the output layer. The first layer consists of eight LNP neurons with four OFF-type and four ON-type spatiotemporal filters. The second layer contains two neurons: L2-1 receives inputs from neurons 1–4 of the first layer, and L2-2 receives inputs from neurons 5–8. The neurons in the second layer are simulated using the LIF model.

Optional: MATLAB runtime version, V1complex_3layers_LIF_SNN.exe.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
MATLAB	MathWorks, Inc. 2018b	N/A
Custom code		https://doi.org/10.5281/zenodo.10049958 or https://github.com/jiankliu/STNMF-SNN

STEP-BY-STEP METHOD DETAILS

Install STNMF

Timing: 10 min

Download the code package.

1. Download the code package from https://github.com/jiankliu/STNMF-SNN (GitHub: https://doi.org/10.5281/zenodo.10049958) or directly clone the GitHub repository, and then add the extracted folder to the MATLAB search path.

Data preprocess

Timing: 1 min

After obtaining the stimulus array and spike trains of the neurons, the data need to be processed into a form suitable for STNMF. The following steps use simulated V1 simple cell data as input.

2. According to the CB and the spike sequence to generate the effective spike-triggered stimulus images (STE).
   a. Based on the spikes of the recorded neuron to find the response time (tsp) of each spike in turn.
   b. Find the stimulus sequence within each time period [tsp-nt+1, tsp] in turn. For the i-th spike, the corresponding 20-frame stimulus sequence is $[s_{i-19}, \ldots s_{i-1}, s_i]$.
   c. Apply Singular Value Decomposition on STA of neurons to obtain spatial receptive fields (spSVD) and temporal filters (tmSVD).
   d. Based on tmSVD, average the stimulus sequence $[s_{i-19}, \ldots s_{i-1}, s_i]$ corresponding to the i-th spike, and reshape it into a row to obtain $s(\tau)^i$. Form the $s(\tau)^i$ of all spikes into an N×P matrix STE.

```
>load('Data.mat');

>pretreatment_snn(CB,spike,nt,nx,ny);

>load('Datapre.mat');

>STE=getSTE_snn(SS,spklist,tmSVD,nt,nx,ny);
```

Note: The nt is generally set to 20, representing 20 stimulus frames. STA is the spike-triggered average method,[6] which calculates the receptive field of neurons based on stimuli and spikes. The generated simulation data or user data is stored in Data.mat. The available data format for STNMF is obtained by pretreatment_snn. The effective spike-triggered stimulus images are then generated using get_STE_snn.

Optional: MATLAB runtime version, Data_Preprocessing.exe.

Decompose STE based on STNMF to obtain W and M

Timing: 5 min

In this step, the STNMF is used to infer the spatial receptive field of the subunits.

3. Perform the STNMF on the STE to obtain the weight matrix W and the module matrix M.
   a. Preset the number of subunits k.
   b. Run the algorithm to get a N×K weight matrix W and a K×P module matrix M.
4. Reshape each row in the module matrix M to the size of the stimulus frame.

```
> load('STE.mat');

> k=8;

> STNMFanalysis_snn('STE.mat','subunit.mat',k,20,1);
```

Note: Remove the noise-like frames in the background, and all preceding frames have clear receptive fields, thus obtaining the spatial receptive field of the subunit. Run STNMFanalysis_SNN obtains spatial receptive fields for inferred subunits.

Optional: MATLAB runtime version, STNMF_WM.exe.

Analyses of the M and W for more computational components

Timing: 15 min (for steps 5 and 6)

Timing: 25 s (for steps 7–9)

In this step, the temporal filter, nonlinearity, and synaptic connection weights of the subunits are inferred by analyzing the module matrix.

5. Obtain the temporal filter for the subunit.
6. Calculate the nonlinearity based on the obtained spatial filter and temporal filter for each subunit.
   a. Firstly, convolve the spatial and temporal filters of the stimulus with the subunits to obtain the generator signal.
   b. Then divide it into 40 bins of equal size, ensuring that each bin contains an equal number of data points.
   c. Finally, visualize the nonlinearity by plotting the mean generator signal against the mean spike rate in each bin as a histogram mean.

```
> calculatTF_subSP_snn('Datapre.mat', 'subunit.mat', 'subTemporalKernel.mat', k,k ,nt,
nx,ny);
```

```
>CalculateOutputGainandDrawNL_snn('Data.mat', 'Datapre.mat', 'subunit.mat', 'subTemporalKernel.mat', 'gainNL.mat');
```

Note: By utilizing the spatial receptive field of this subunit to filter all stimulus sequences and subsequently averaging them, we can obtain the temporal filter for that specific subunit. Run calculatTF_subSP_snn to obtain the time filter for the inferred subunits. Run CalculateOutputGainandDrawNL_snn to obtain gain and nonlinearity.

Optional: MATLAB runtime version, temporalfilter.exe, GainandNL.exe.

In this part, by analyzing the weight matrix to infer the synaptic connection weights and spike trains of the subunits.

7. Average each column of the weight matrix to get the connection weight W_j for each subunit.
8. Sum each column of the weight matrix W to obtain the ON-OFF attribute of the.
9. If the subunit is of type OFF, take the minimum of each row in the weight matrix W. Then classify the spike into the module corresponding to the minimum.

```
>get_module_weight_snn('STE.mat','subunit.mat','module_weight.mat');

>load('module_weight.mat','W_matrix');

>load('subunit.mat','unit','Ngood');

>spklist_sub=get_subSTA_min_snn(Ngood,spklist,W_matrix,STE,nx,ny);

>subunit_sp=get_subunit_spike_train_snn(Ngood,spike,spklist_sub);

>save('subunit_sp.mat','subunit_sp');
```

Note: If the sum of the weights of the subunits is greater than 0, it is an ON property, and if the sum is negative, it is an OFF property. Run get_module_weight_snn, get_subunit_spike_train_snn to obtain the synaptic weights and spike subsets.

Optional: MATLAB runtime version, module_weight.exe, subunit_spike_train.exe.

EXPECTED OUTCOMES

STNMF is a flexible tool widely used for visual system identification, requiring only fine-structure spike trains of neurons under visual stimuli, and is thus suitable for simultaneous recording of a large number of neurons. This approach is suitable for circuit inference of various types of neurons from the retina, lateral geniculate, or visual cortex. After validation, STNMF can analyze the computational components of presynaptic subunits of neurons based on visual stimuli and recorded spikes, including spatiotemporal filters, nonlinearities, prominent connection weights, ON-OFF properties, and spike subsets. We use simulated V1 simple cells as an example to examine the expected outcomes of STNMF inference, as shown in Figure 1. In the user's own experiment, it is possible to consider enlarging the size of the stimulus image. Furthermore, it is worth noting that stimulating images should not be limited solely to white noise; rather, they can encompass complex color noise or even natural images. Figure 1B shows the spatial receptive field, temporal filter, and nonlinearities of the inferred subunits. The polarity of the temporal filter indicates the ON/OFF property of the subunits. Additionally, when the number of subunits k is set larger than the actual number of subunits, the additional modules lack a distinct receptive field shape and resemble noise. Figure 1C exhibits the molded spike trains (bottom) and the inferred spike trains (top).

QUANTIFICATION AND STATISTICAL ANALYSIS

Timing: 3 s

To determine the accuracy of the spatial filter, temporal filter, nonlinearity, and connection weights of the subunits inferred by STNMF, visual methods can be used to compare them with the ground truth. In addition, spatial filters, ON-OFF attributes, connection weights, and spike subsets can also be quantized using the following code.

1. Measure the accuracy of inference results by calculating the dot product between the real spatiotemporal filter and the inferred spatiotemporal filter.
2. The polarity of the time filter reflects the on off polarity of the subunit.
3. Compare the ratio of the inferred weights of each subunit with the nonlinear gain (calculate the gain value of each nonlinearity, i.e., the difference between the maximum and minimum values) ratio. The closer the ratio between the two groups, the more accurate the inference.
4. By calculating the Pearson correlation coefficient between the real spike train and the inferred spike train. The larger the correlation coefficient, the more accurate it is.

```
>load('Data.mat');
>load('subunit.mat', 'unit');
>load('subTemporalKernel.mat','subTemporal');
>nsub=size(spM,1);
>dotspM=zeros(k,nsub);
>on_off=[];
>CC=[];
>for i=1:k
> [~,ind]=max(abs(subTemporal(i,:)));
> on_off(i)=sign(subTemporal(i, ind));
> for j=1:nsub
> temp=reshape(unit1/norm(unit1),1,[]);
> dotspM(i,j)=dot(spM(j,:),temp);
> R=corrcoef(subunit_sp(i,:), sub_SP(j,:));
> CC(i,j)=R(1,2);
> end
>end
```

Note: The unit, subTemporal, on_off, and subunit_sp store the spatial receptive field, temporal filters, ON/OFF properties and spike subsets of the inferred subunits. At the same time, spM, tmM, and sub_SP separately store subsets of the true spatial receptive field, temporal filters, and spikes of the subunits.

Optional: MATLAB runtime version, Quantification_statistic.exe.

LIMITATIONS

Currently, this algorithm only supports system identification of visual system based on spike data. The signal-to-noise ratio, stimulus image resolution, and experimental recording duration depend on the recorded spike response. First, the higher the Signal to Noise Ratio of the spike, the less data is needed to obtain accurate inference results. Second, the higher the resolution of the stimulus image, the more refined the receptive field of the inferred subunits. Finally, if the experimental recording time is longer, enough spikes are collected to ensure the accuracy of the inference.

TROUBLESHOOTING

When running code following this protocol in MATLAB to run, carefully read the step-by-step method details.

Problem 1

Errors occur when running this protocol using MATLAB, or when the output does not match the desired result.

Potential solution

- Please check the code syntax first. For example, misspelled functions or file names.
- In case of memory error, first, consider improving the virtual memory of MATLAB, or use a computer with higher memory.
- Then, check if the input data is too large. On the one hand, it is possible to consider reducing the size of the stimulus image. On the other hand, consider shortening the duration of the input data. Related to: step-by-step method details section - data preprocess. Or related to: collecting data section - Simulated data 1, 2, 3.

Problem 2

ran1 not found in the current folder or MATLAB path.

Potential solution

- We have placed the compiled files for Mac, Windows and Linux in the code folder. However, if the above issues arise, please compile ran1.cpp file in the MATLAB command line window. Related to: collecting data section - Simulated data 1, 2, 3.

```
> mex ran1.cpp;
```

Problem 3

The function or variable spM or sub_SP is not recognized.

Potential solution

- When a user runs the code in the quantification and statistical analysis section using their own experimental data, this issue occurs if the user does not have a ground truth about the subunit calculation component. The spM, tmM, and sub_SP separately store subsets of the true spatial receptive field, temporal filters, and spikes of the subunits. Related to: step-by-step method details section - data preprocess

Table 1. Example of raw data measured on a plate reader

Raw data		
Name	Size	Description
CB	64 × 900000	White noise stimulus matrix, 900000 is the total number of frames stimulated.
Spike	900000 × 1	Spike train of recorded neurons
Nt	20	Number of temporal filters spanning stimulus images
Nx	8	Pixels per row of stimulus frames
Ny	8	Pixels per column of stimulus frames
tmM	6 × 20	Ground truth of temporal filters
spM	6 × 64	Ground truth of spatial filters
sub_SP	6 × 900000	Ground truth of spike trains in the first layer

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Zhaofei Yu (yuzf12@pku.edu.cn).

Materials availability

This study did not generate new unique reagents.

Data and code availability

The code as well as simulated data generated during this study are available at the GitHub repository and can be accessed using the following link: https://github.com/jiankliu/STNMF-SNN (GitHub: https://doi.org/10.5281/zenodo.10049958).

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China grants 62176003 and 62088102 (Z.Y.) and Royal Society Newton Advanced Fellowship of UK grant NAF-R1-191082 (J.K.L.).

AUTHOR CONTRIBUTIONS

J.K.L. and Z.Y. conceived the idea. S.J. and J.K.L. developed the software and generated the analysis protocol described here. S.J., J.K.L., and Z.Y. wrote, reviewed, and edited the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Jia, S., Xing, D., Yu, Z., and Liu, J.K. (2021). Dissecting cascade computational components in spiking neural networks. PLoS Comput. Biol. *17*, e1009640. https://doi.org/10.1371/journal.pcbi.1009640.

2. Liu, J.K., Schreyer, H.M., Onken, A., Rozenblit, F., Khani, M.H., Krishnamoorthy, V., Panzeri, S., and Gollisch, T. (2017). Inference of neuronal functional circuitry with spike-triggered non-negative matrix factorization. Nat. Commun. *8*, 149. https://doi.org/10.1038/s41467-017-00156-9.

3. Liu, J.K., and Gollisch, T. (2015). Spike-triggered covariance analysis reveals phenomenological diversity of contrast adaptation in the retina. PLoS Comput. Biol. *11*, e1004425. https://doi.org/10.1371/journal.pcbi.1004425.

4. Liu, J.K., Karamanlis, D., and Gollisch, T. (2022). Simple model for encoding natural images by retinal ganglion cells with nonlinear spatial integration. PLoS Comput. Biol. *18*, e1009925. https://doi.org/10.1371/journal.pcbi.1009925.

5. Jia, S., Yu, Z., Onken, A., Tian, Y., Huang, T., and Liu, J.K. (2022). Neural system identification with spike-triggered non-negative matrix factorization. IEEE Trans. Cybern. *52*, 4772–4783. https://doi.org/10.1109/TCYB.2020.3042513.

6. Chichilnisky, E.J. (2001). A simple white noise analysis of neuronal light responses. Network *12*, 199–213. https://doi.org/10.1088/0954-898X/12/2/306.

7. Liu, Y.-H., and Wang, X.J. (2001). Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. J. Comput. Neurosci. *10*, 25–45. https://doi.org/10.1023/A:1008916026143.