Solving Convolutional LASSO with LCA

Neuromorphic Computing Lab | Intel Labs

Nengo Summer School 2019

Rev 0.4

Legal Information

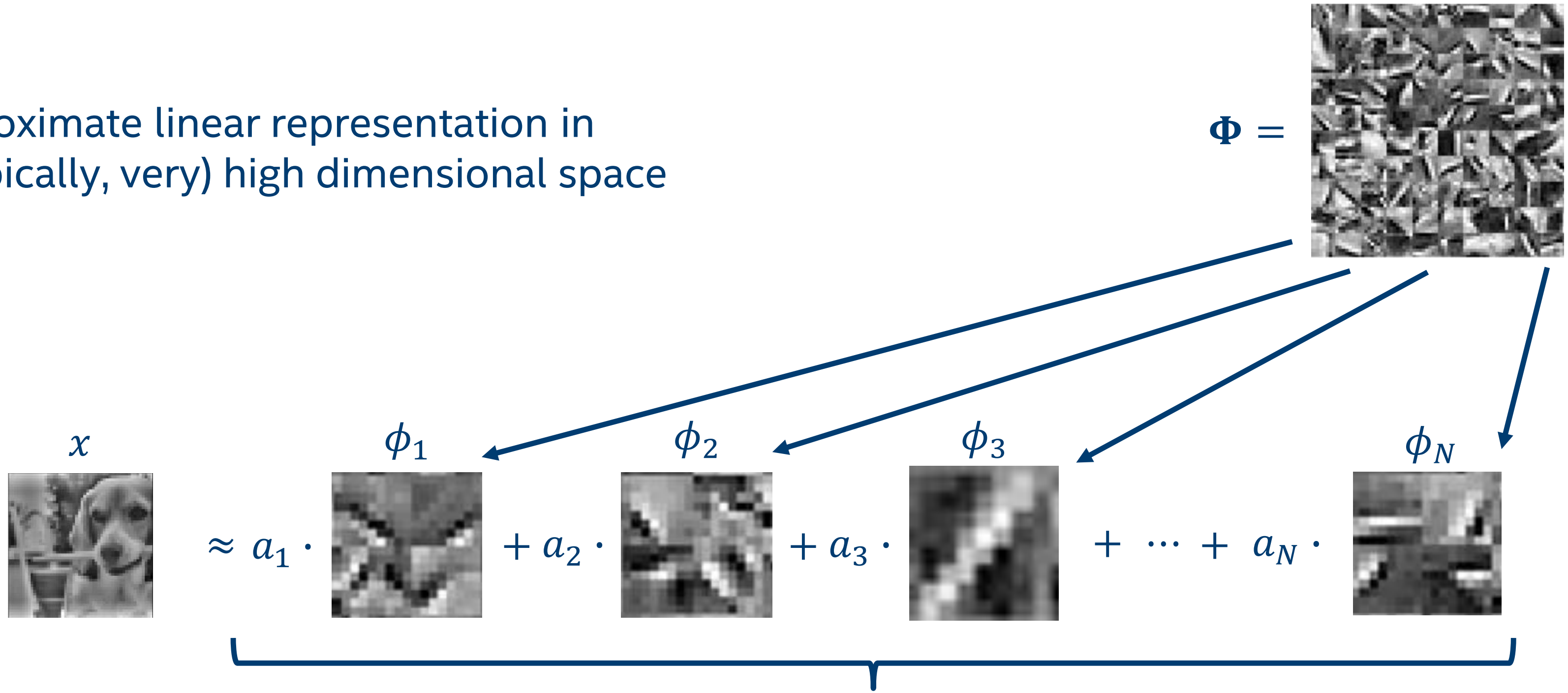
This presentation contains the general insights and opinions of Intel Corporation ("Intel"). The information in this presentation is provided for information only and is not to be relied upon for any other purpose than educational. Intel makes no representations or warranties regarding the accuracy or completeness of the information in this presentation. Intel accepts no duty to update this presentation based on more current information. Intel is not liable for any damages, direct or indirect, consequential or otherwise, that may arise, directly or indirectly, from the use or misuse of the information in this presentation.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

SPARSE CODING AS A LASSO PROBLEM

Sparse coding 2D images

Approximate linear representation in
a (typically, very) high dimensional space



Sparse → most of the a_js are 0

Conventionally expensive but solved efficiently on Loihi!

Sparse coding: LASSO formulation

Reconstruction error

ℓ-1 regularization term

LASSO Cost function: $E = \frac{1}{2}\|\mathbf{x} - \boldsymbol{\Phi} \cdot \boldsymbol{a}\|_2^2 + \lambda \cdot \|\boldsymbol{a}\|_1$

sparse code: $\boldsymbol{a}^* = \underset{\boldsymbol{a}}{\operatorname{argmin}} E(\boldsymbol{a})$

$\|\boldsymbol{a}\|_1 = \sum_j |a_j|$

$\boldsymbol{\Phi} \cdot \boldsymbol{a} = \sum_k \Phi_{jk} a_k$

approx. reconstruction

LASSO to LCA

- Want to solve LASSO…
- Using conventional gradient descent leads to ISTA/FISTA solvers.
- From ISTA derive LCA (locally competitive algorithm) dynamics.

LCA* solves LASSO by gradient descent

(*) C. J. Rozell, D. H. Johnson, and R. G. Baranuik *Neural Computation*, 20:2526 – 2563, 2008.

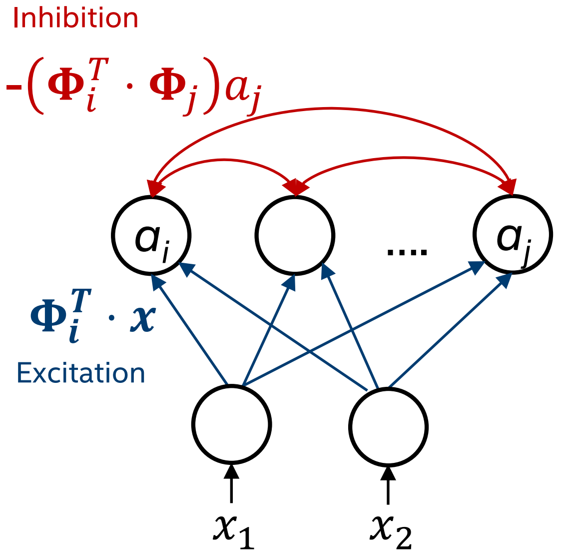
LCA-derived network structure

LCA dynamics:

$$\dot{\boldsymbol{u}} = \frac{1}{\tau}(\boldsymbol{\Phi}^T\boldsymbol{x} - \boldsymbol{u} - (\boldsymbol{\Phi}^T\boldsymbol{\Phi} - \boldsymbol{I}) \cdot \boldsymbol{a})$$

$$\boldsymbol{a} = \mathcal{T}_\lambda(\boldsymbol{u})$$

Network structure:



Feature neurons compete to represent to reconstruct inputs

Analog to Spiking LCA: Rate Coding

- Equivalence between analog LCA dynamics and spike based models*
  - Sparse code a is represented by neuron spiking rates

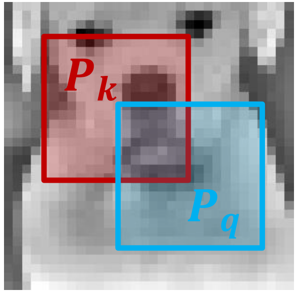
Analog-LCA ⟺ Spiking-LCA

* P. Tang, T.-H. Lin, and M. Davies, arXiv:1705.05475v1

Resource efficiency through Convolutional LCA

Image: $\boldsymbol{x} \in \mathbb{Z}^{n \times n}$

Image patches: $\boldsymbol{P}_k, \boldsymbol{P}_q \in \{0,1\}^{n \times n}, |P_k| = r$

$\boldsymbol{x} =$ 

Linearized image: $\boldsymbol{x}' \in \mathbb{Z}^{n^2} = \mathbb{Z}^m$

Elementary dictionary: $\boldsymbol{\Psi} \in \mathbb{Z}^{r \times p}$

$\boldsymbol{\Psi} =$ 

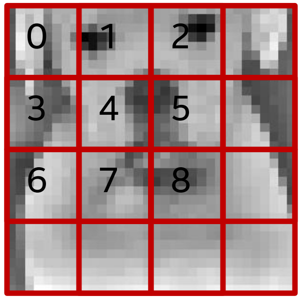
Patch dictionary:
$\boldsymbol{\Phi}_k \in \mathbb{Z}^{m \times p}, \boldsymbol{\Phi}_\mathrm{k}[\boldsymbol{P}_k, :] = \boldsymbol{\Psi}$

- Extend to convolutional LCA
  - Image patches
  - Generalized (inhibitory) interaction matrix:
    - $\mathbf{W}_{\mathrm{k,q}} = \Phi_\mathrm{k}^T \cdot \Phi_\mathrm{q} - I$
  - Neurons per patch integrate input from overlapping patches
  - Significant increase in number of connections

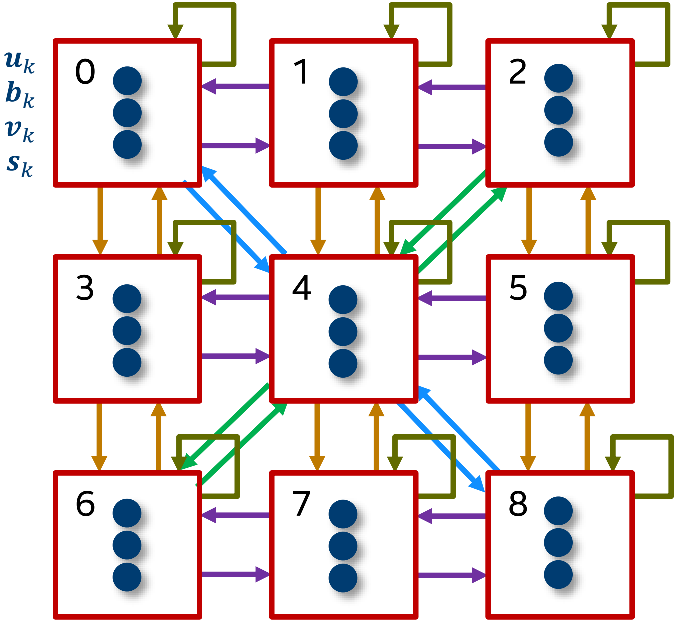
Loihi takes advantage of convolutional topology for efficient network compression

Neural network structure

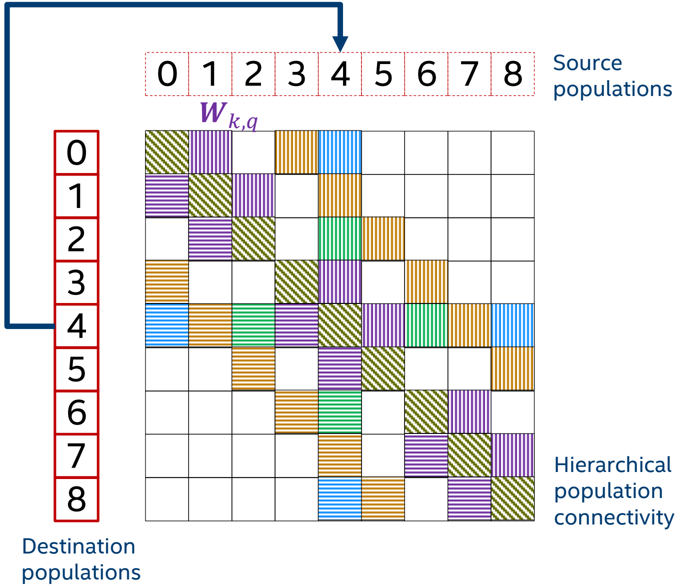
Input image with patches



Patches -> Populations with population connectivity



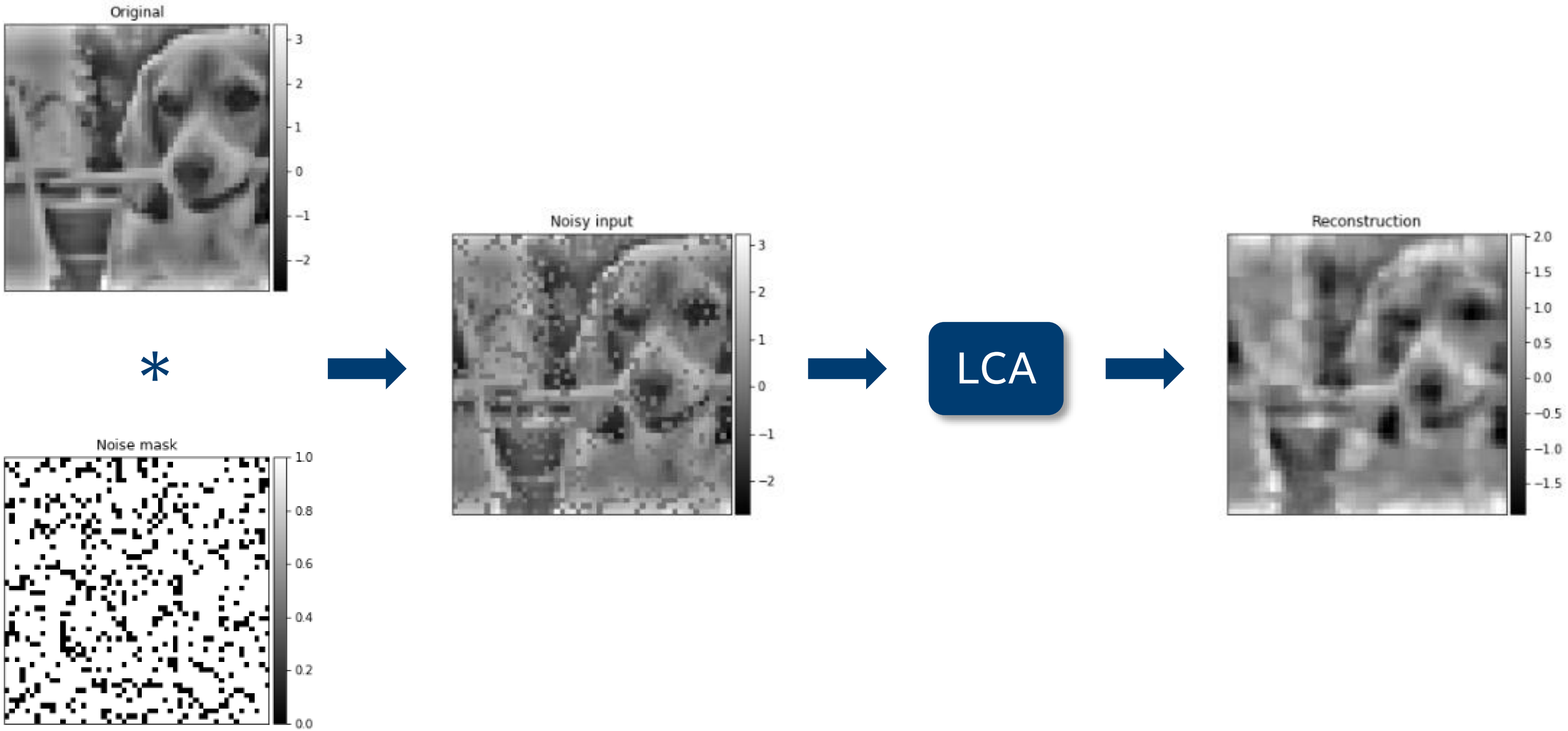
Connectivity matrix



Sharing of same sub-matrices reduces resource requirements!

TUTORIAL: IMAGE DE-NOISING WITH LCA ON LOIHI

De-noising workload



Tutorial: LCA Module for Image De-Noising

*

* Other names and brands may be claimed as the property of others

Questions?