Hardware architecture and routing-aware training for optimal memory usage: a case study

A work of

Theo Ballet and Jimmy Weber

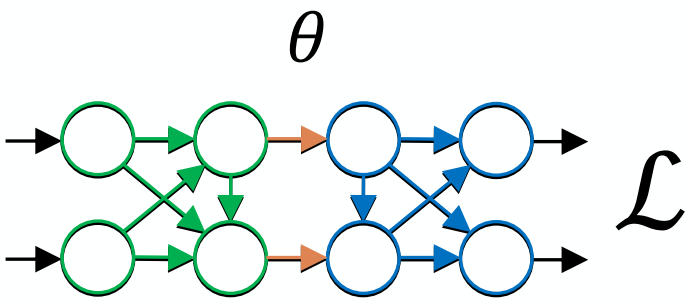
Supervised by

Melika Payvand

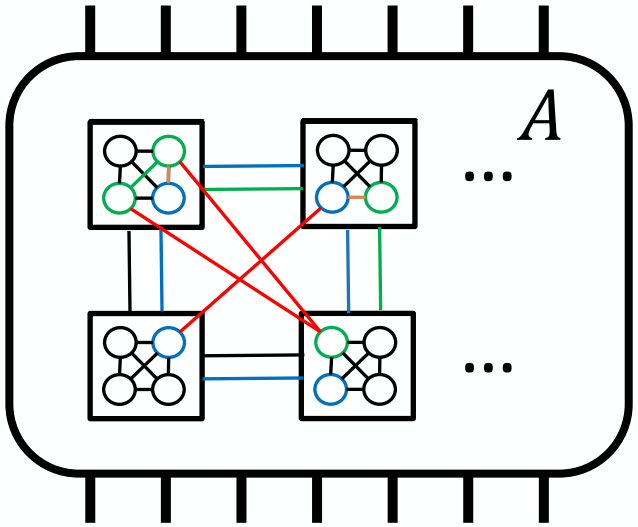
Hardware-software co-design

The software:
Spiking neural network

The hardware:
Neuromorphic chip



Mapping and routing R_A



- Enough ressources on A?
- Valid placement and routing R_A?

Hardware-software co-design

The software:
Spiking neural network

Objective:
- Minimize Loss $\mathcal{L}$

Parameters:
- Dataset
- Neuron Model
- #Neurons

Fixed

- Connectivity θ
- Weight Values θ

Degree of Freedom (DoF)

The hardware:
Neuromorphic chip

Objective:
- Power, Performance, Area: PPA

Parameters:
- Technology
- Architecture
- Placement and Routing R_A

Fixed

- Architecture configuration A (#Cores, Fan-I/O,…)

DoF

Software Objective:

$$\min_{\theta} \mathbb{E}[\mathcal{L}(\theta)]$$

Hardware Constraints:

$$C(\theta, A, R_A) \leq 0$$

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

Hardware-software co-design

Differentiable

Non-differentiable

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\ C(\theta, A, R_A) \leq 0]$$

With gradient descent

$C < 0$: Under usage of resources
$C = 0$: Complete usage of resources
$C > 0$: Over usage of resources: Not mappable!

Hardware-software co-design

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

…But first, let's get inspired from another method...

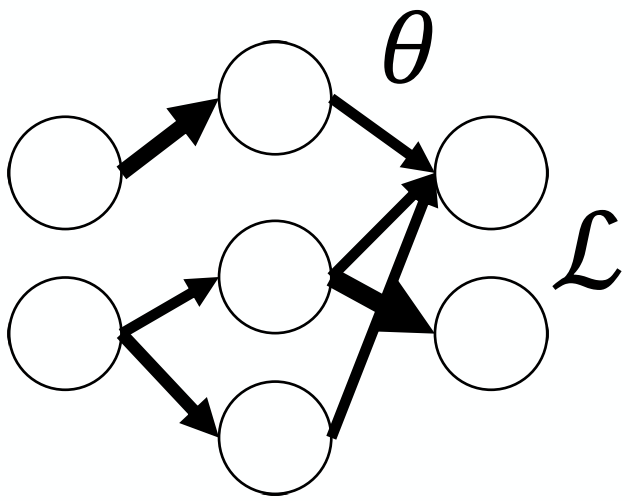
DeepR[1]

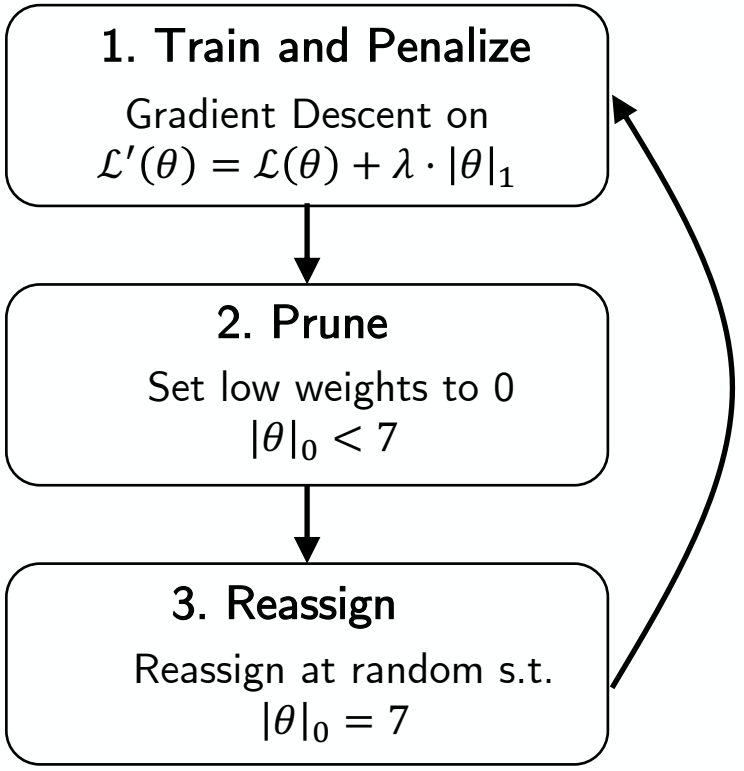
Problem:

"Train a neural network with fixed sparsity (e.g. 7 weights) and unconstrained connectivity."

Solution:

Dynamical architecture search





$|\theta|_0$: Number of non-zero elements
$|\theta|_1$: Sum of all elements

1. BELLEC, Guillaume, KAPPEL, David, MAASS, Wolfgang, et al.Deep rewiring: Training very sparse deep networks. arXiv preprint arXiv:1711.05136, 2017.

Hardware-software co-design

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

…But first, let's get inspired from another method…

Why is Jimmy telling you about this?

Hardware-software co-design

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

…But first, let's get inspired from another method...

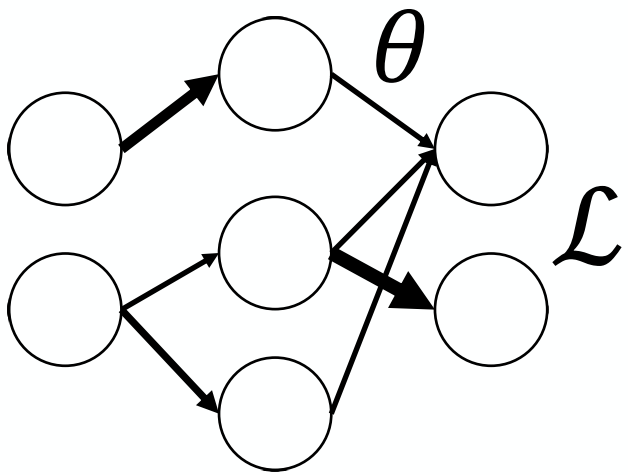
DeepR[1]

Problem:

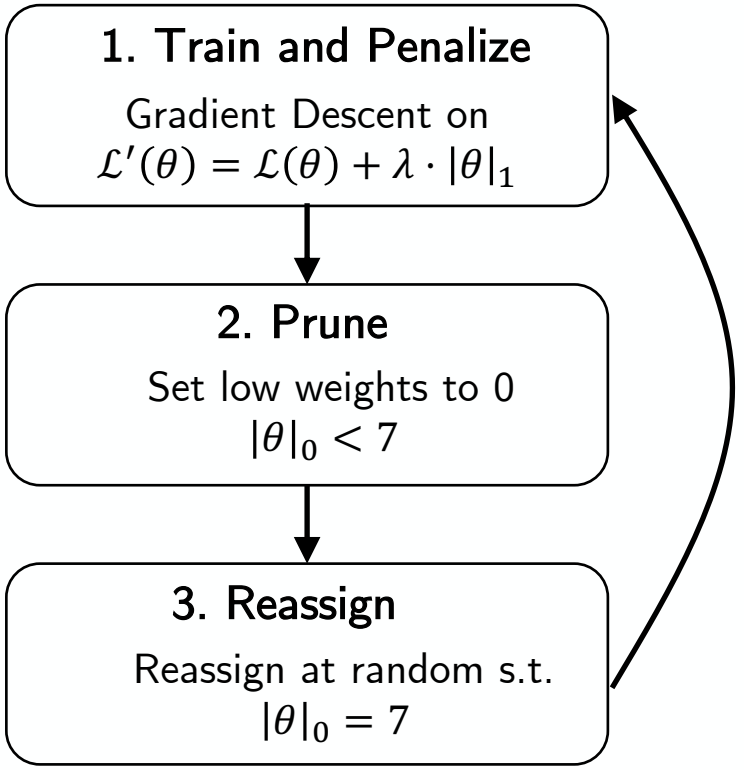
"Train a neural network with fixed sparsity (e.g. 7 weights) and unconstrained connectivity."

Solution:

Dynamical architecture search



$$C = |\theta|_0 - 7$$



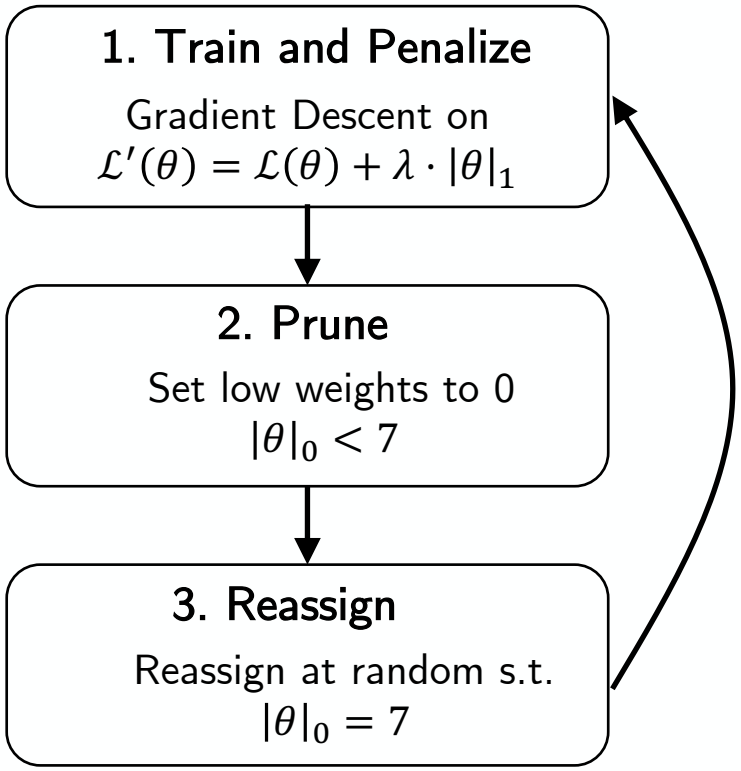
$|\theta|_0$: Number of non-zero elements
$|\theta|_1$: Sum of all elements

1. BELLEC,G, et al. Deep rewiring: Training very sparse deep networks. 2017.

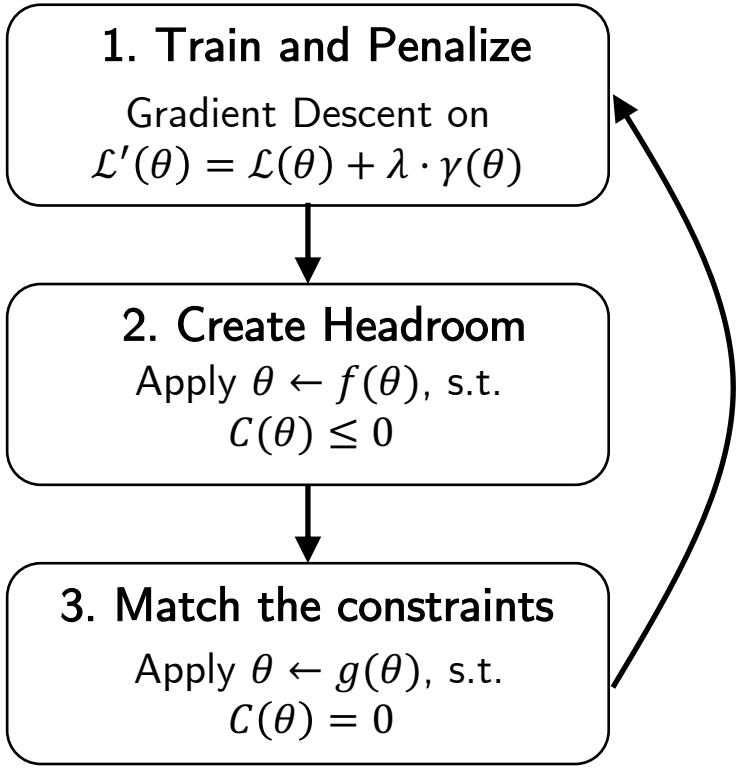
Hardware-software co-design

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

DeepR[1] ($C = |\theta|_0 - 7$)



Extension of DeepR



Considerations:

- Computing the constraints load a lot.
  - ➢ **Evaluating $C(\theta)$ should be efficient.**
- $\gamma(\theta)$ is a regularizer: a proxy of $C(\theta)$.
  It is a soft constraints, differentiable w
  θ.
  - ➢ **Get $\gamma(\theta)$, a proxy of $C(\theta)$.**

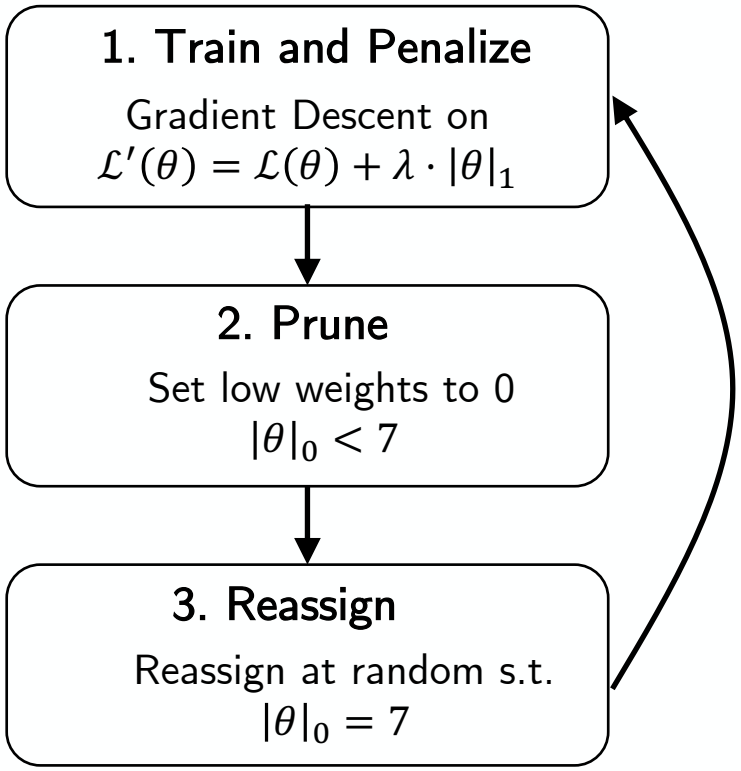
$|\theta|_0$: Number of non-zero elements
$|\theta|_1$: Sum of all elements

8

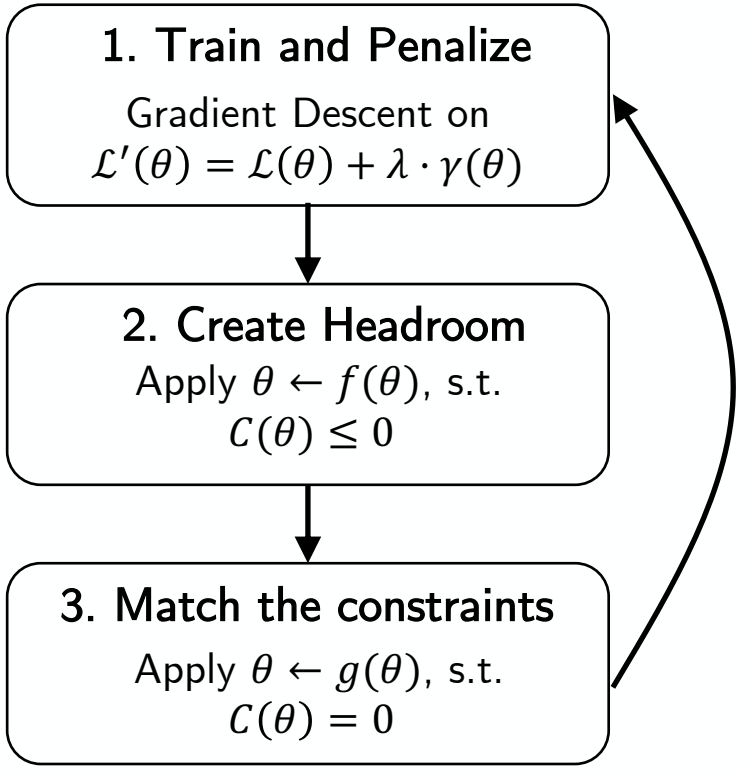
Hardware-software co-design

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

DeepR[1] ($C = |\theta|_0 - 7$)



Extension of DeepR



Considerations:

- Computing the constraints load a lot.
  - ➢ **Evaluating $C(\theta)$ should be efficient.**
- $\gamma(\theta)$ is a regularizer: a proxy of $C(\theta)$.
  It is a soft constraints, differentiable w
  θ.
  - ➢ **Get $\gamma(\theta)$, a proxy of $C(\theta)$.**

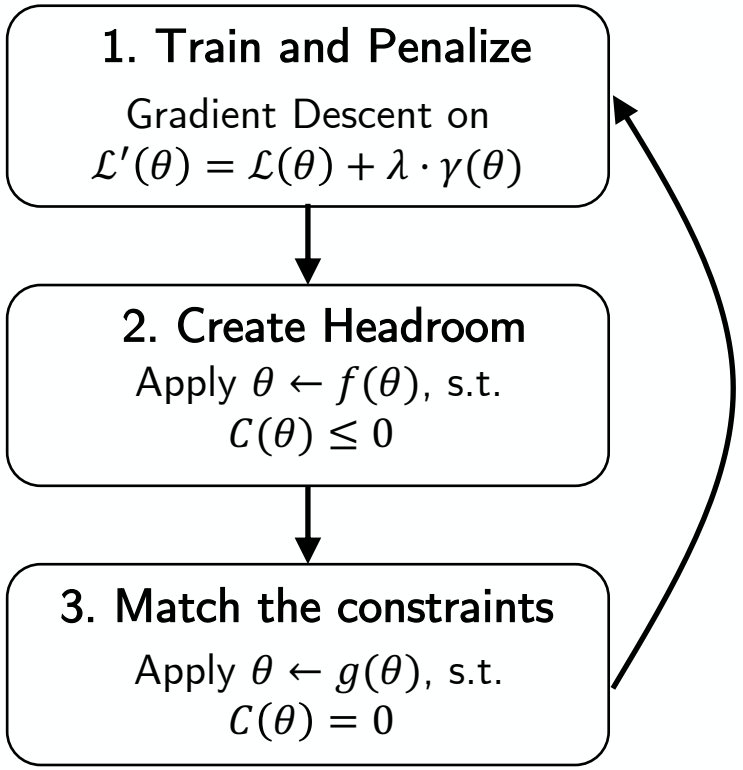
$|\theta|_0$: Number of non-zero elements
$|\theta|_1$: Sum of all elements

9

Hardware-software co-design

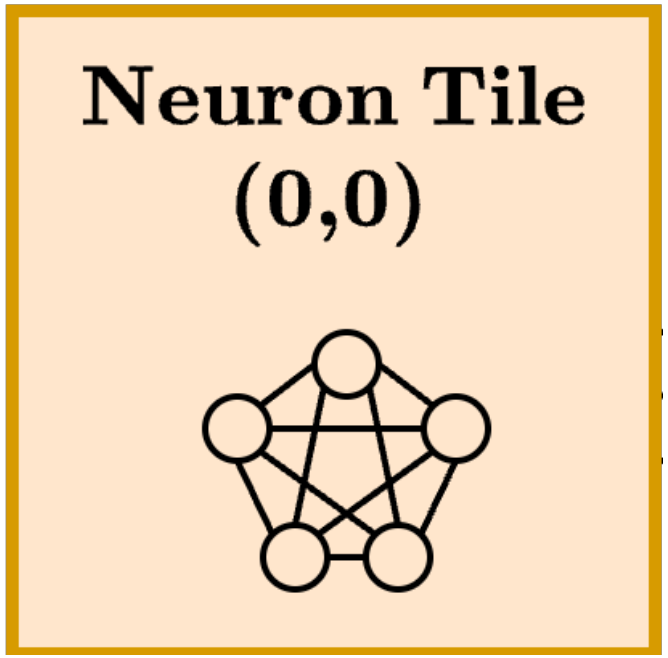
$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

Extension of DeepR



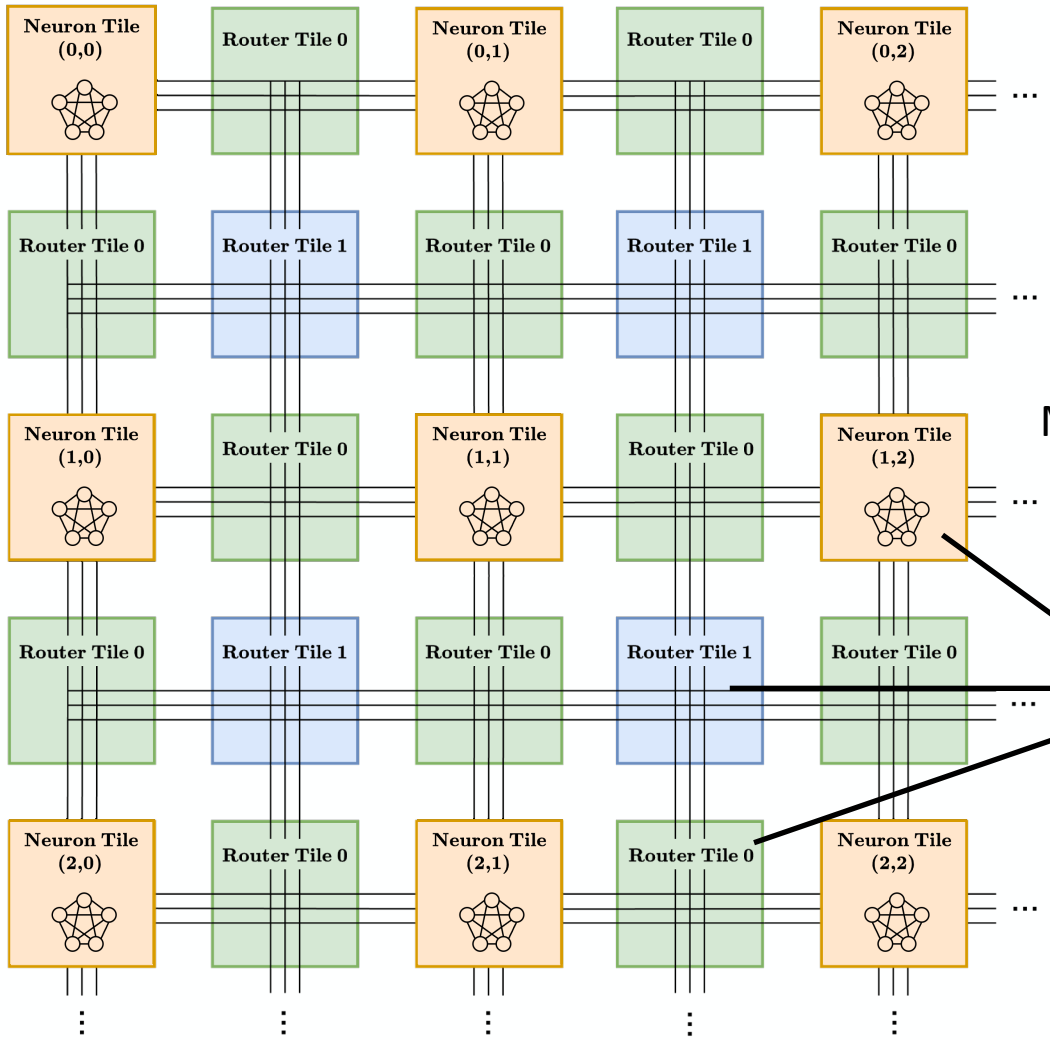
- **Evaluating $C(\theta)$ should be efficient.**
- **Get $\gamma(\theta)$, a proxy of $C(\theta)$.**

The Mosaic[2] as a case study



2: Dalgaty, T. et al. Mosaic: in-memory computing and routing for small-world spike-based neuromorphic systems. 2024

The Mosaic[2] as a case study



The hardware:
Neuromorphic chip

Objective: ➢ Power, Performance, Area

Parameters:
- ➢ Technology
- ➢ Architecture
- ➢ Placement and Routing R_A

Fixed

- Memristor Crossbars
- Systolic array
- Manhattan

→ Size & Fan-I/O of the different tiles

- ➢ Architecture configuration A (#Cores, Fan-I/O,…)

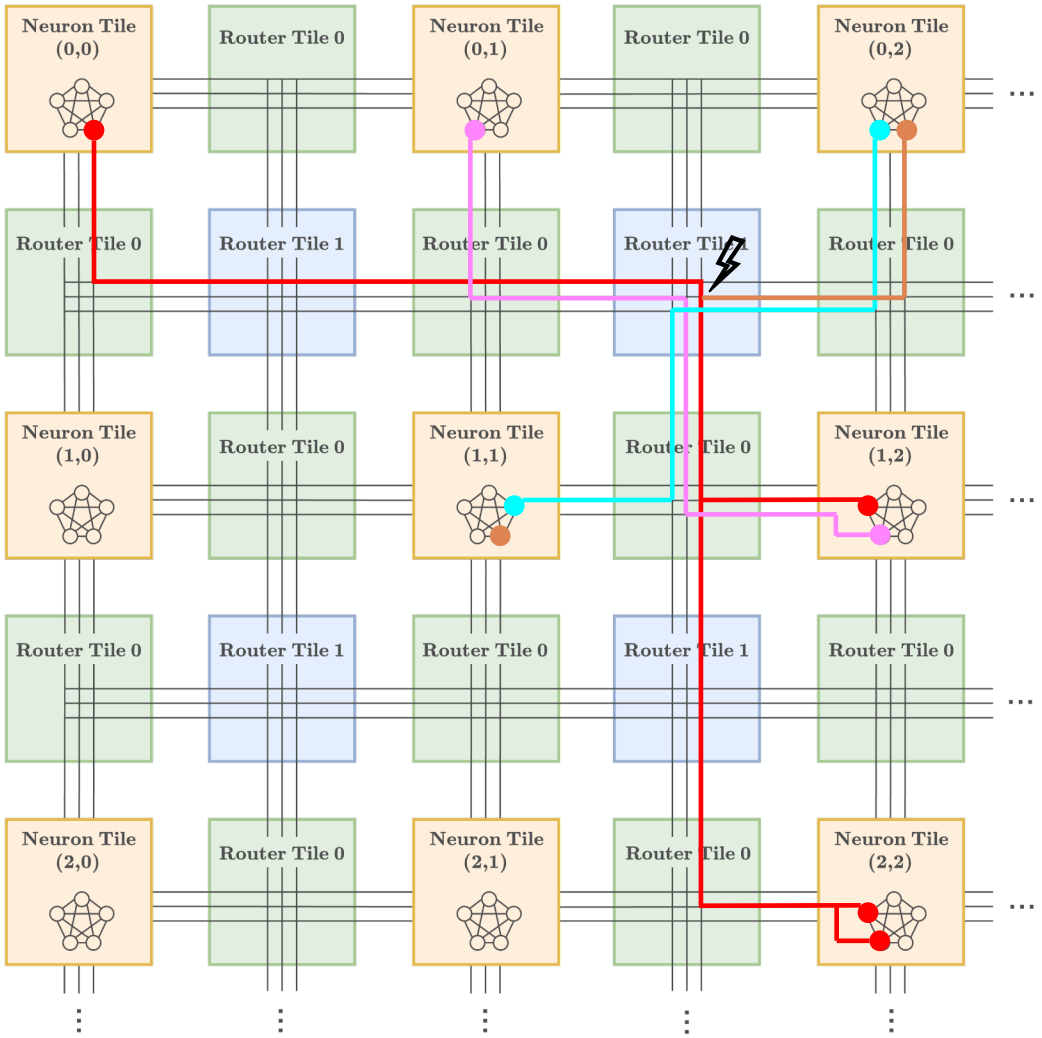
DoF

Hardware Constraints:

$$C(\theta, A, R_A) \leq 0$$

2: Dalgaty, T. et al. Mosaic: in-memory computing and routing for small-world spike-based neuromorphic systems. 2024

The Mosaic[2] as a case study



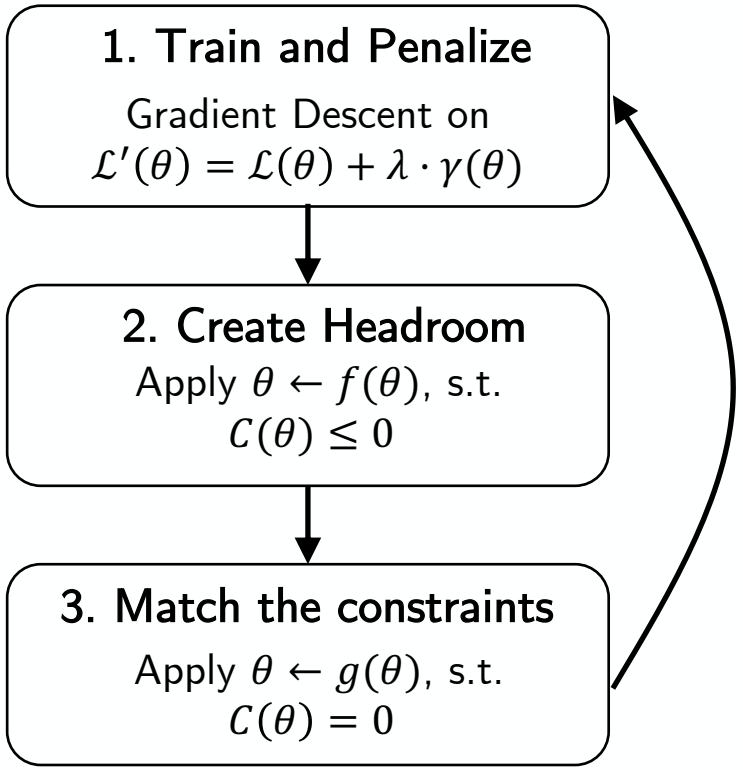
Constraints of MOSAIC mappability = routing resources

2: Dalgaty, T. et al. Mosaic: in-memory computing and routing for small-world spike-based neuromorphic systems. 2024

Hardware-software co-design

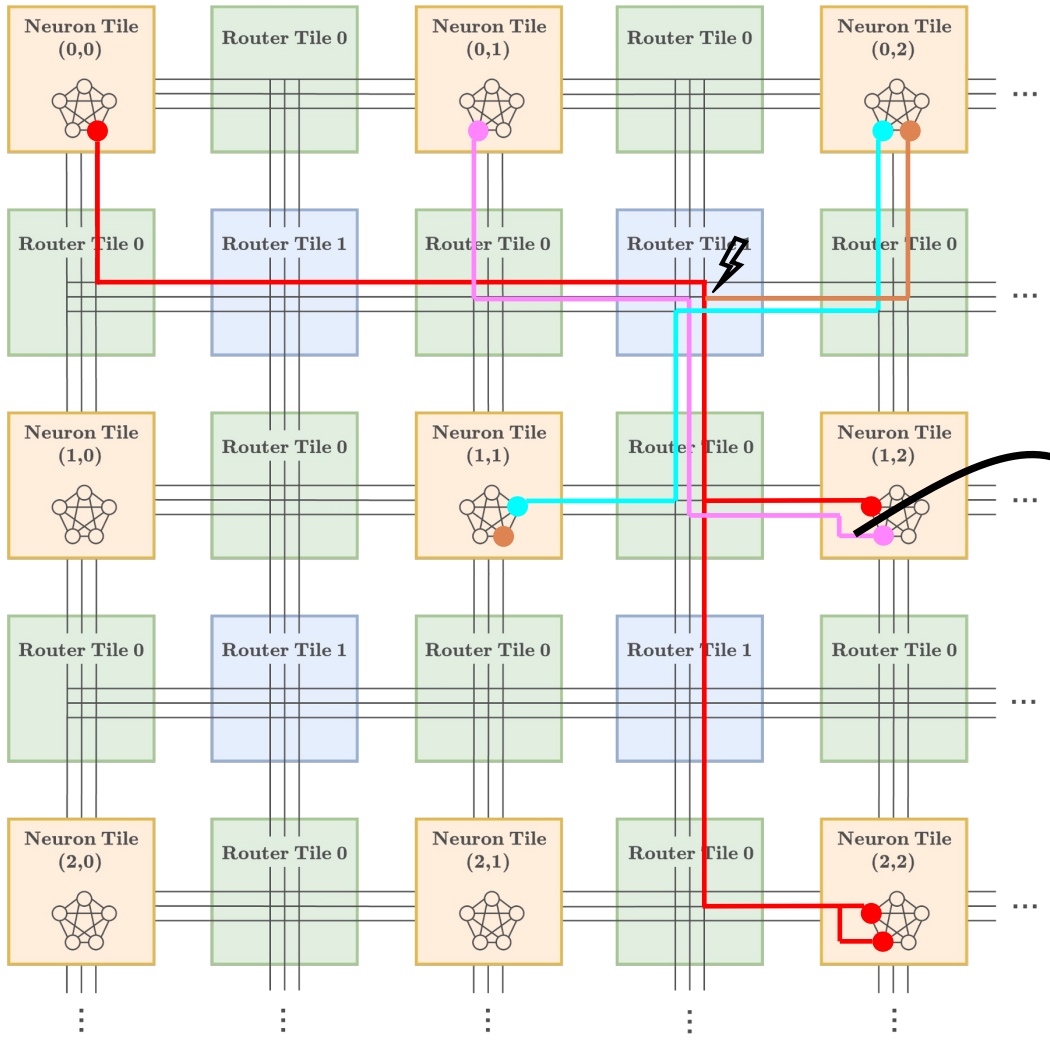
$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

Extension of DeepR



- **Evaluating $C(\theta)$ should be efficient.**
- **Get $\gamma(\theta)$, a proxy of $C(\theta)$.**

The Mosaic as a case study



➢ **Evaluating $C(\theta)$ should be efficient.**

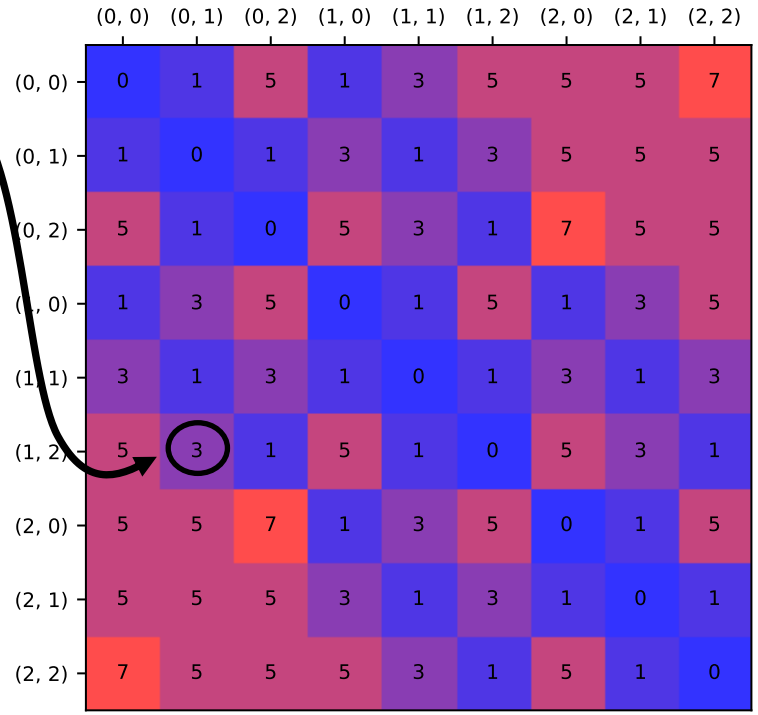
Constraints of MOSAIC mappability = routing resources

☹ Not efficient

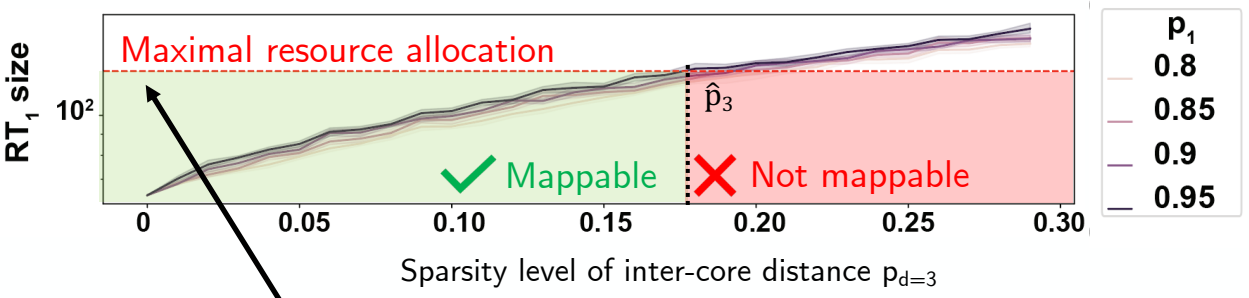
Proportional to the connection distance

Fast approximation: Penalizing according to the distance

distances $H(\mathrm{NT}_i, \mathrm{NT}_j)$



The Mosaic as a case study



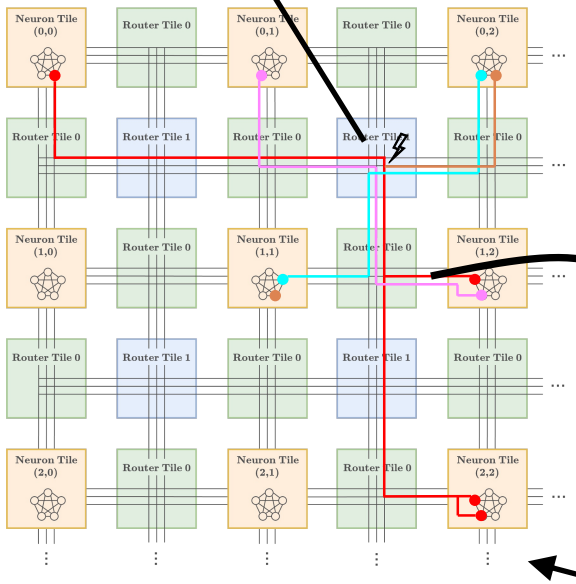
➢ **Evaluating $C(\theta)$ should be efficient.**

Constraints of MOSAIC mappability = routing resources

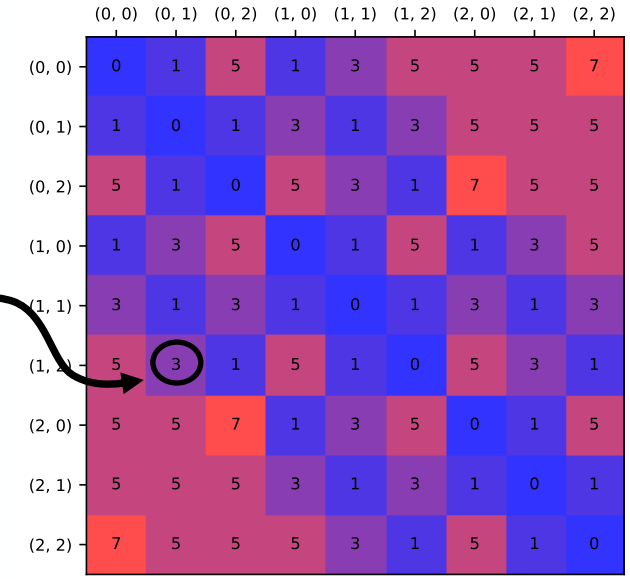
☹ Not efficient

Proportional to the connection distance

Fast approximation: Penalizing according to the distance

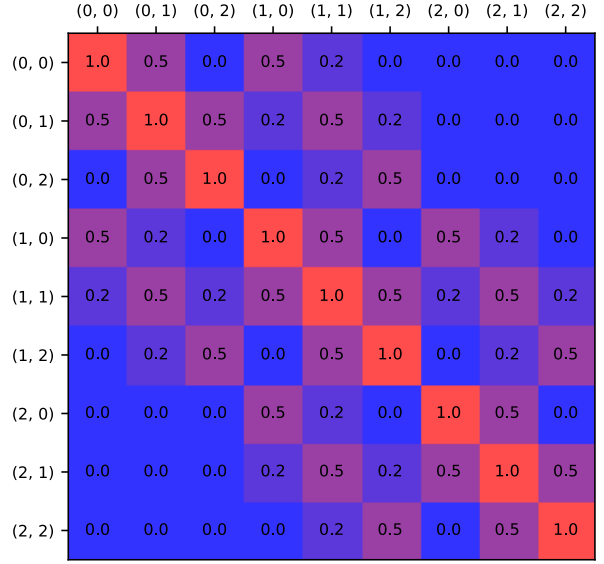


distances $H(\mathrm{NT}_i, \mathrm{NT}_j)$

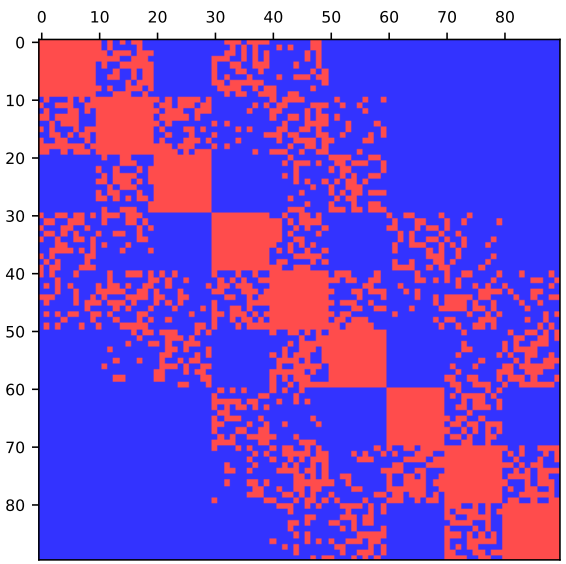


Sparsity wrt. distances

$p_1 = 50\%$, $p_3 = 20\%$, $p_{d>3}=0\%$

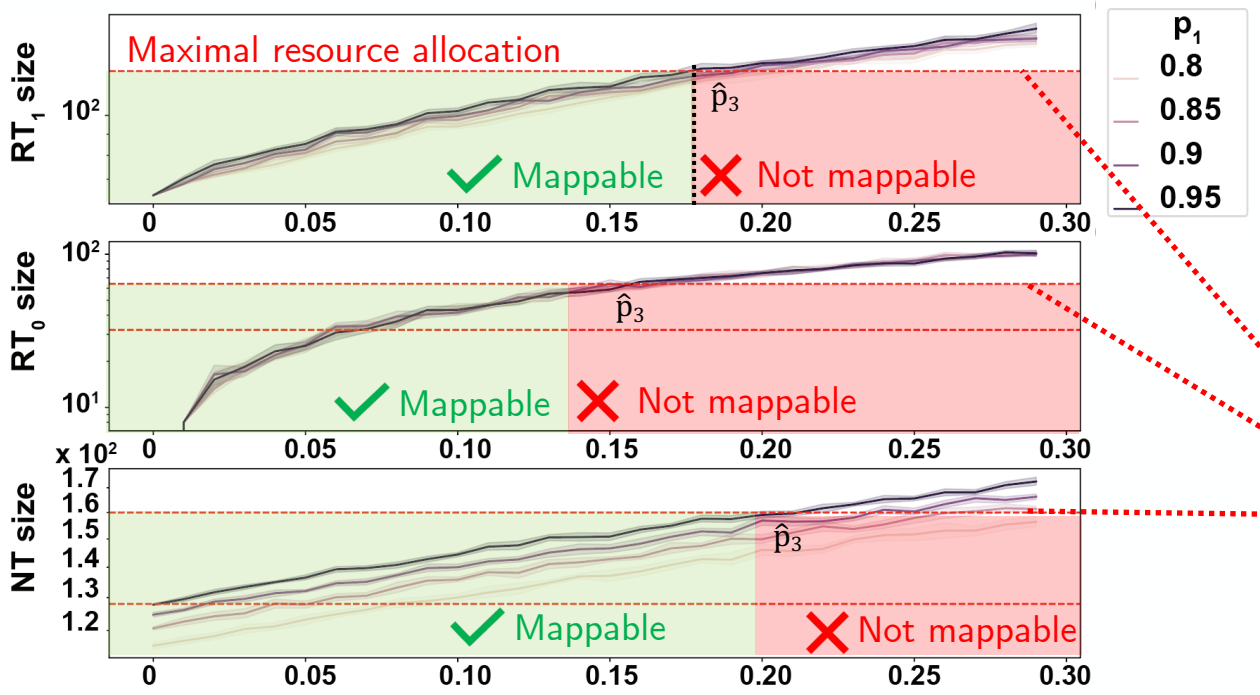


Example of network



Mappable?

The Mosaic as a case study



Sparsity level of inter-core distance $p_{d=3}$

➢ **Evaluating $C(\theta)$ should be efficient.**

Constraints of MOSAIC mappability = routing resources

☹ Not efficient

Proportional to the connection distance

Fast approximation: Penalizing according to the distance

$$C(\theta, A, R_A) = P(\theta) - \hat{P} \leq 0$$

→ Good approximation: Low σ^2

→ Still non differentiable

→ A function of the precise configuration A: #Cores, Fan-I/O,...

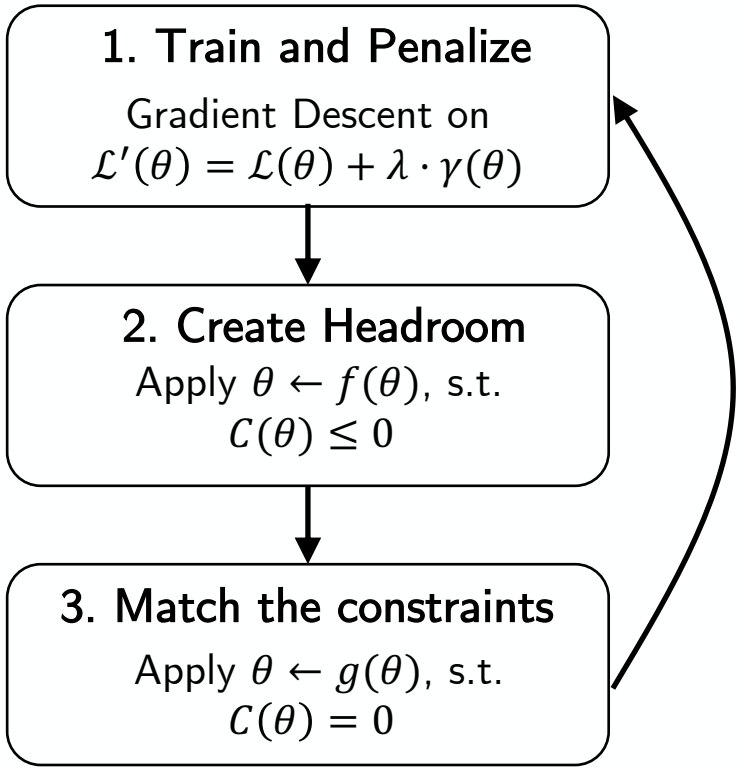
$$\hat{P} = \{\widehat{p_1}, \widehat{p_3}, \widehat{p_5}, \dots\}$$

Inter-core sparsity level, for core at distance d.

Hardware-software co-design

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

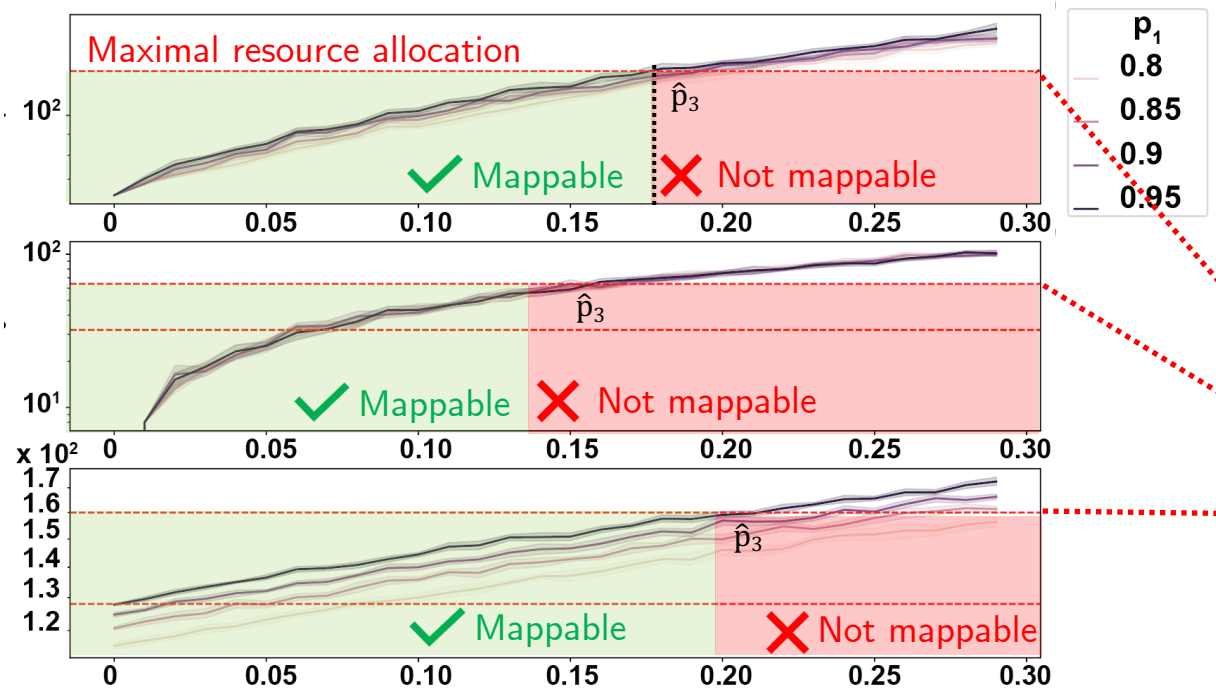
Extension of DeepR



- **Evaluating $C(\theta)$ should be efficient.**

- **Get $\gamma(\theta)$, a proxy of $C(\theta)$.**

The Mosaic as a case study



Sparsity level of inter-core distance $p_{d=3}$

$$\hat{P} = \{\widehat{p_1}, \widehat{p_3}, \widehat{p_5}, \dots\}$$

Inter-core sparsity level, for core at distance d.

➢ **Evaluating $C(\theta)$ should be efficient.**

Constraints of MOSAIC mappability = routing resources

☹ Not efficient

Proportional to the connection distance

Fast approximation: Penalizing according to the distance

$$C(\theta, A, R_A) = P(\theta) - \hat{P} \leq 0$$

→ Good approximation: Low σ^2

→ Still non differentiable

→ A function of the precise configuration A: #Cores, Fan-I/O,...

➢ **Get $\gamma(\theta)$, a proxy of $C(\theta)$.**

Sparsity: $p_d = \frac{|\theta_d|_0}{N_d^2}$

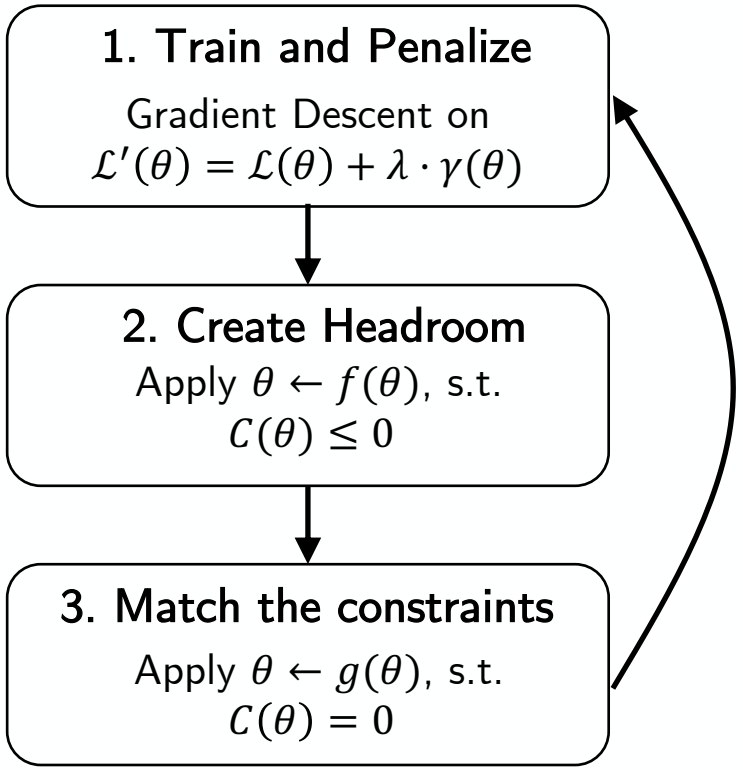
With: N^2 : Number of potential connections
$|\theta|_0$: Number of non-zero elements
$|\theta|_1$: Sum of all elements

$$\gamma(\theta) = |\theta_d|_1$$

Hardware-software co-design

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

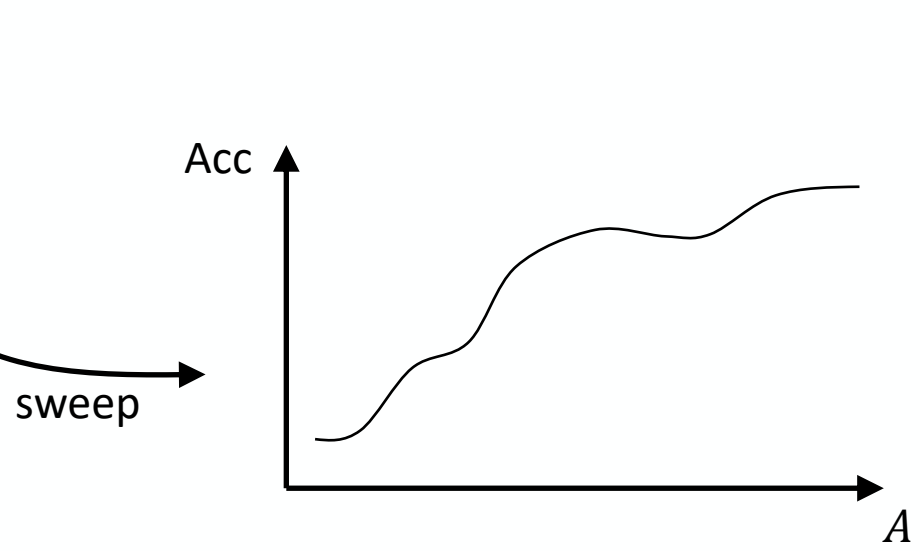
Extension of DeepR



- ✓ ➢ **Evaluating $C(\theta)$ should be efficient.**
- ✓ ➢ **Get $\gamma(\theta)$, a proxy of $C(\theta)$.**

Hardware-software co-design

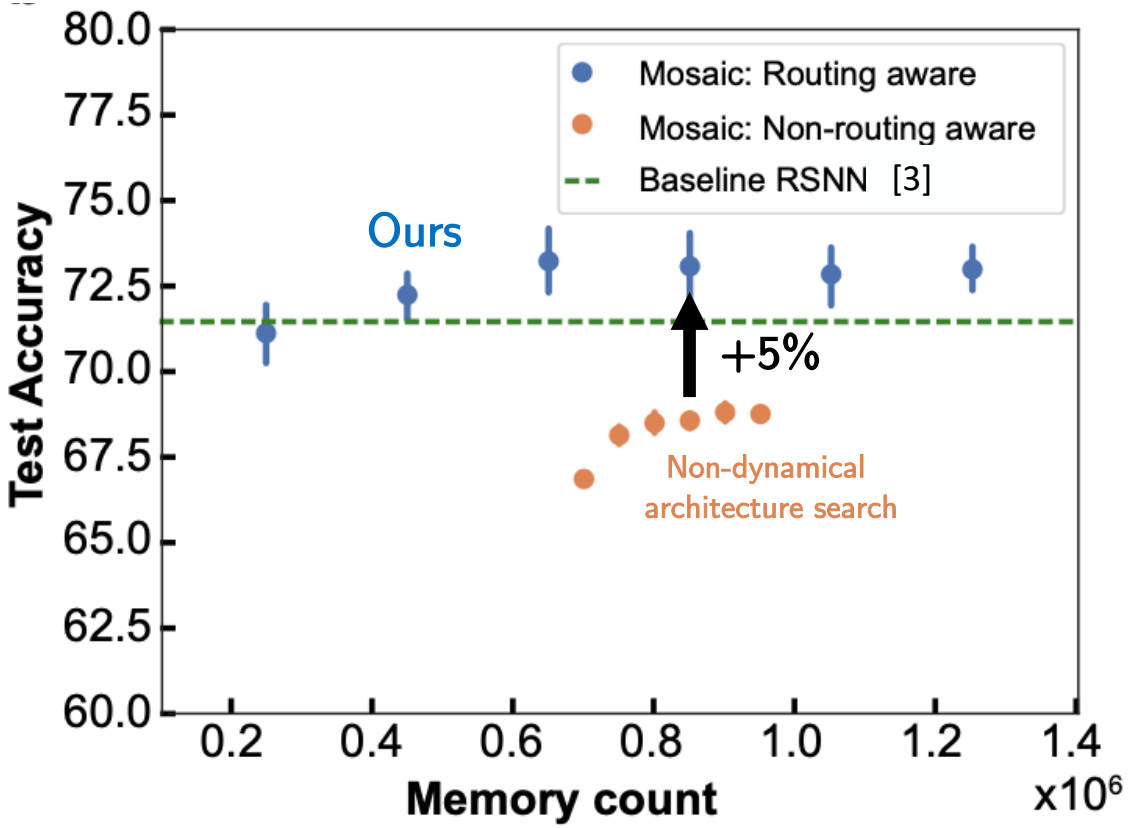
$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$



Hardware-software co-design

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

Results on SHD

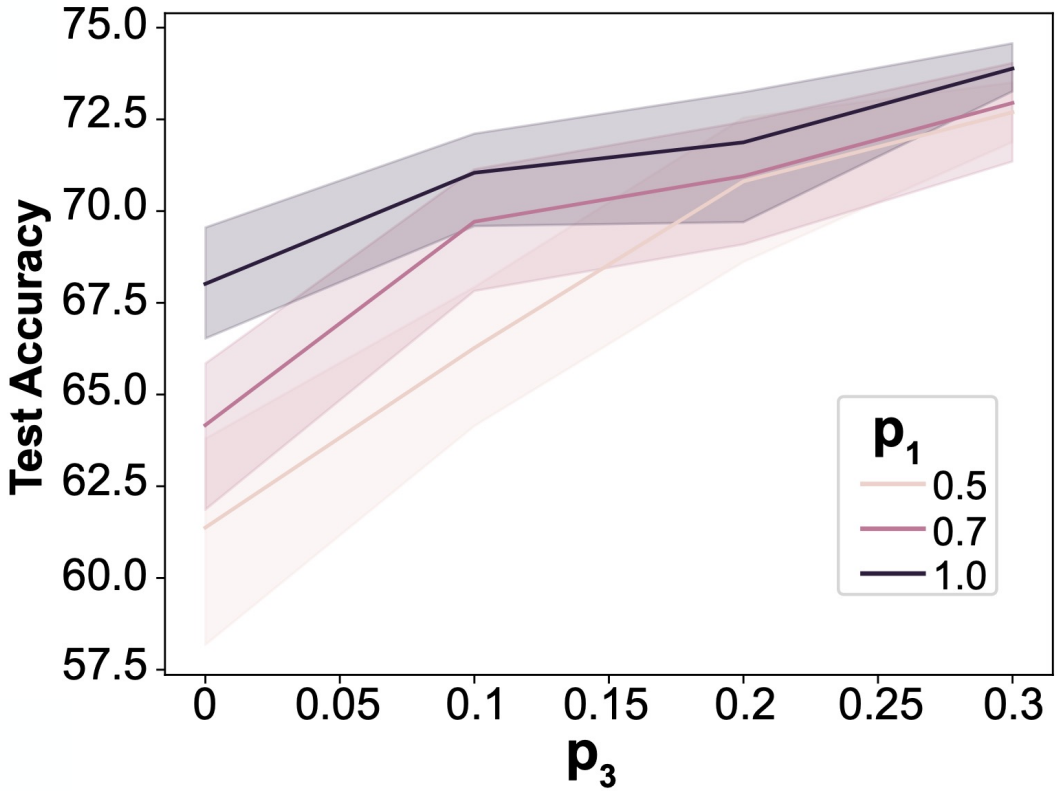


3. Cramer, B. et al. The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks. 2020.

Hardware-software co-design

$$\min_{\theta,\, A} \mathbb{E}[\mathcal{L}(\theta) |\, C(\theta, A, R_A) \leq 0]$$

Further insights and outlook



→ A family of HW have the same accuracy

→ Accuracy is a function of sparsity profile (inter-population sparsity based on their distance)

Take home message

Mathematical formulation of architectural and routing constrains: $C(\theta, A, R_A)$

Developing a method inspired by DEEPR $\min_{\theta} \mathbb{E}[\mathcal{L}(\theta) | \, C(\theta, A, R_A) \leq 0]$

Hardware architecture and routing-aware training for optimal memory usage: a case study

Such that $C = 0$

The Mosaic Architecture

⇒ Get accuracy/memory improvement

Acknowledgment

The EIS-lab, retreat 2024

Théo Ballet,
Paris.

Melika Payvand,
ETH/UZH