# Re-basin via implicit Sinkhorn differentiation

Paper Tag: THU-AM-357

Fidel Guerrero Pena, Heitor Rapela Medeiros, Thomas Dubail, Masih Aminbeidokhti, Eric Granger, Marco Pedersoli

LIVIA, Department of Systems Engineering
École de Technologie Supérieure

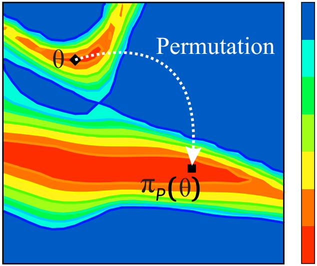
ÉTS Engineering for Industry
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
Université du Québec

LIVIA
LABORATOIRE D'IMAGERIE, DE VISION ET D'INTELLIGENCE ARTIFICIELLE

JUNE 18-22, 2023
CVPR
VANCOUVER, CANADA

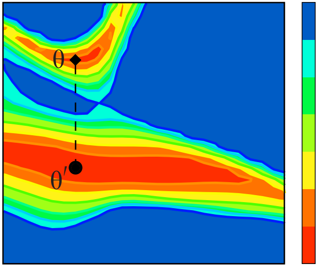
# Introduction

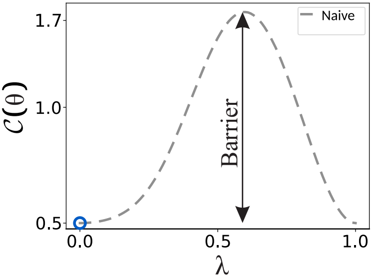
**Re-basin:** Find a suitable permutation of the parameters such that minimizes a given objective



# Introduction

**Example:** Re-basin for linear mode connectivity



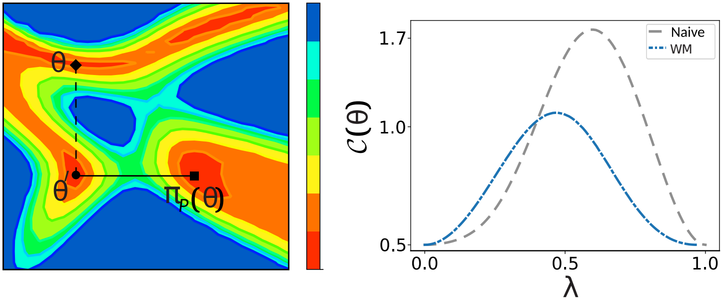




**Objective:** Find the permutation of $\theta$ that minimizes Barrier$(\theta, \theta')$

$$\text{Barrier}(\theta,\theta') = \sup_{\lambda} \left[ \left[ \mathcal{C}\left( (1-\lambda)\theta + \lambda\theta' \right) \right] - \left[ (1-\lambda)\mathcal{C}(\theta) + \lambda\mathcal{C}(\theta') \right] \right]$$

# Introduction

**Example:** Re-basin for linear mode connectivity




WM - Weight Matching re-basin [Ainsworth et al., 2022]

# Re-basin via implicit Sinkhorn differentiation

Learning the permutations using Sinkhorn operator

**Model:** $\ell_i'(z) = \sigma(S_\tau(P_i)W_i S_\tau(P_{i-1}^T)z + S_\tau(P_i)b_i)$

**Cost functions:**

L2 distance loss: $\mathcal{C}_{L2}(\mathcal{P};\theta_A,\theta_B) = ||\theta_A - \pi_{\mathcal{P}}(\theta_B)||^2$

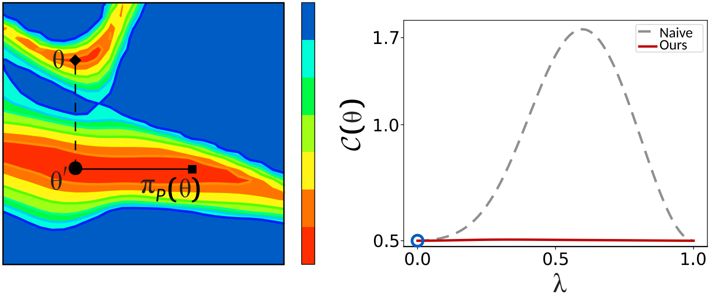
Middle point loss: $\mathcal{C}_{Mid}(\mathcal{P};\theta_A,\theta_B) = \mathcal{C}\left(\frac{\theta_A + \pi_{\mathcal{P}}(\theta_B)}{2}\right)$

Random point loss: $\mathcal{C}_{Rnd}(\mathcal{P};\theta_A,\theta_B) = \mathcal{C}\left((1-\lambda)\theta_A + \lambda\pi_{\mathcal{P}}(\theta_B)\right)$, $\lambda \sim U(0,1)$

**Optimization problem:** $\mathcal{P}^* = \arg\min_{\mathcal{P}} \mathcal{C}(\mathcal{P};\theta_A,\theta_B)$

# Introduction

**Example:** Re-basin for linear mode connectivity

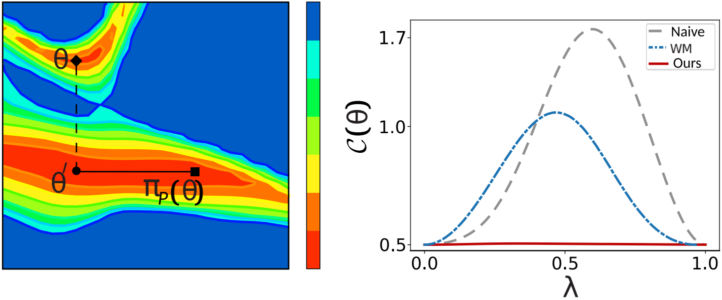



**Objective:** Find the permutation of $\theta$ that minimizes $\mathcal{C}_{Rnd}(\mathcal{P};\theta,\theta')$

$$\mathcal{C}_{Rnd}(\mathcal{P};\theta,\theta')=\mathcal{C}\left((1-\lambda)\theta'+\lambda\pi_{\mathcal{P}}(\theta)\right),\lambda\sim \mathrm{U}(0,1)$$

# Introduction

**Example:** Re-basin for linear mode connectivity




WM - Weight Matching re-basin [Ainsworth et al., 2022]

# Proposed method

# Permutation transformation

Let a feed forward neural network be defined as:

$f_\theta(x) = (\ell_h \circ \ldots \circ \ell_1)(x),$

$\ell_i(z) = \sigma(W_i z + b_i)$

a permuted model is defined as:

$f'_\theta(x) = (\ell'_h \circ \ldots \circ \ell'_1)(x),$

$\ell'_i(z) = \sigma(P_i W_i P_{i-1}^T z + P_i b_i),$

with $P_h = P_0^T = \boldsymbol{I}$

such that the functionally equivalence holds, $f_\theta(x) = f'_\theta(x), \forall x$

# Permutation transformation

$$f'_\theta(x) = (\ell'_h \circ \ldots \circ \ell'_1)(x),$$
$$\ell'_i(z) = \sigma(P_i W_i P_{i-1}^T z + P_i b_i),$$

where $P_i$ is a permutation matrix, i.e., a binary matrix such that there is only a single 1 per row and column.

$$P_i^T P_i = I$$

Example of permutation matrices:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

# Proposal – Re-basin via implicit Sinkhorn differentiation

Permuted model definition with soft permutation matrices

$$\ell_i'(z) = \sigma(S_\tau(P_i) W_i S_\tau(P_{i-1}^T) z + S_\tau(P_i) b_i)$$

with Sinkhorn operator [Mena et al., 2018] defined as:

$$S_\tau^{(0)}(X) = \exp\left(\frac{X}{\tau}\right),$$

$$S_\tau^{(t+1)}(X) = \mathcal{T}_c(\mathcal{T}_r(S_\tau^{(t)}(X))).$$

**Drawback:** Not efficient differentiation.

**Solution:** Implicit differentiation [Eisenberger et al., 2022]

# Proposal – Re-basin via implicit Sinkhorn differentiation

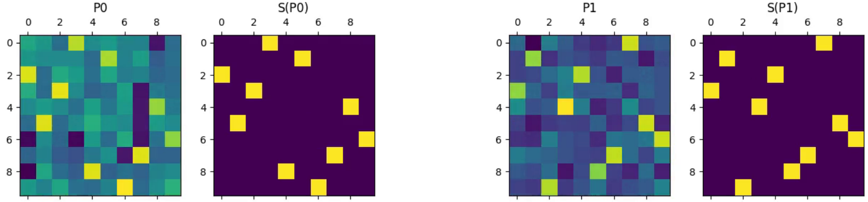



Figure: Soft permutation matrices before – $P_i$ – and after – $S(P_i)$ – applying Sinkhorn.

# Proposal – Re-basin via implicit Sinkhorn differentiation

Learning the permutations using Sinkhorn operator

**Model:** $\ell_i'(z) = \sigma(S_\tau(P_i) W_i S_\tau(P_{i-1}^T) z + S_\tau(P_i) b_i)$

**Cost functions:**

L2 distance loss: $\mathcal{C}_{L2}(\mathcal{P}; \theta_A, \theta_B) = ||\theta_A - \pi_{\mathcal{P}}(\theta_B)||^2$

Middle point loss: $\mathcal{C}_{Mid}(\mathcal{P}; \theta_A, \theta_B) = \mathcal{C}\left(\frac{\theta_A + \pi_{\mathcal{P}}(\theta_B)}{2}\right)$

Random point loss: $\mathcal{C}_{Rnd}(\mathcal{P}; \theta_A, \theta_B) = \mathcal{C}\left((1-\lambda)\theta_A + \lambda \pi_{\mathcal{P}}(\theta_B)\right)$, $\lambda \sim U(0,1)$

**Optimization problem:** $\mathcal{P}^* = \arg\min_{\mathcal{P}} \mathcal{C}(\mathcal{P}; \theta_A, \theta_B)$

Use with your favorite gradient descent-based algorithm!

# Proposal – Re-basin incremental learning

Cost function:

$$\mathcal{C}_{CL}(\delta_i, \mathcal{P}_i; \theta_i) = \mathcal{C}\left(\frac{\theta_i + \pi_{\mathcal{P}_i}(\theta_i)}{2} + \delta_i\right) + \beta||\delta_i||^2$$

Training process solves the optimization problem:

$$\delta_i^*, \mathcal{P}_i^* = \arg\min_{\delta_i, \mathcal{P}_i} \mathcal{C}_{CL}(\delta_i, \mathcal{P}_i; \theta_i)$$

After convergence we choose a model:

$$\theta_{i+1} = (1-\alpha)\theta_i + \alpha\pi_{\mathcal{P}_i}(\theta_i) + \delta_i$$

# Proposal – Re-basin via implicit Sinkhorn differentiation

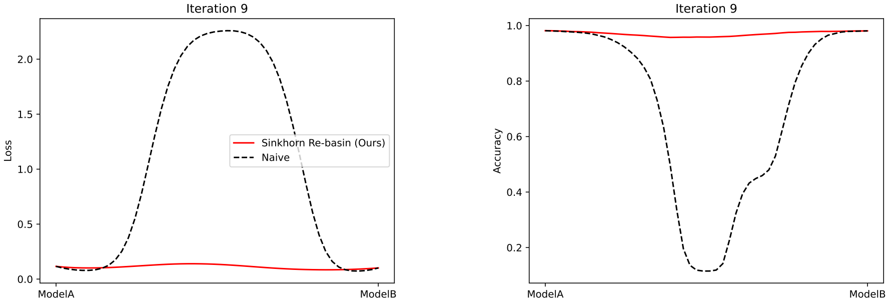
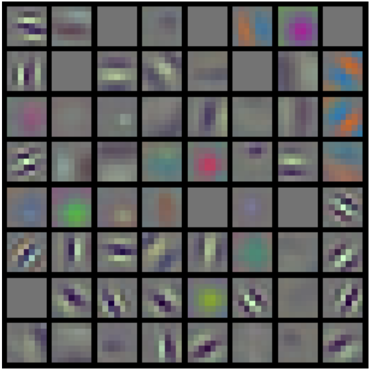



Figure: Linear mode connectivity using two VGG models trained over Mnist.

# Experimental results

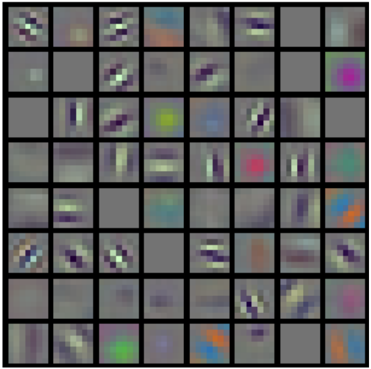
# Experiment 1 – Models alignment

**conv1** of pre-trained Resnet18



$\theta_A$

**conv1** of randomly permuted pre-trained Resnet18:



$\theta_B = \pi_{\mathcal{P}}(\theta_A)$

Can we find the permutations that align the models?

# Experiment 1 – Models alignment

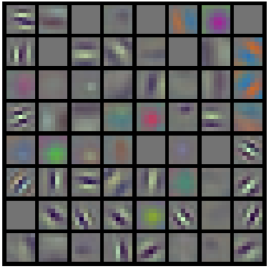
| Method | Init | 2 hidden ↓ | 4 hidden ↓ | 8 hidden ↓ |
|---|---|---|---|---|
| WM | Rnd | 6.05±9.17 | 4.12±6.58 | 0.50±1.55 |
| $\mathcal{C}_{L2}$ (Ours) | | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 |
| WM | Pol3 | 0.57±2.84 | 0.07±0.46 | 0.01±0.10 |
| $\mathcal{C}_{L2}$ (Ours) | | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 |
| WM | Pol1 | 0.27±0.94 | 0.00±0.00 | 0.00±0.00 |
| $\mathcal{C}_{L2}$ (Ours) | | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 |

Table: L1 distance between the estimated and expected re-basing with different network initialization and depth. Distances are scaled $\times 10^3$.

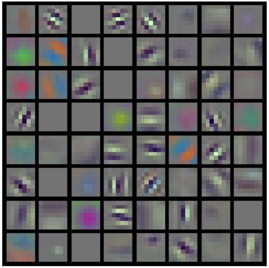
# Experiment 1 – Models alignment

Resnet18 models alignment

$\theta_A$



$\pi_{\mathcal{P}}(\theta_A)$



# Experiment 1 – Models alignment

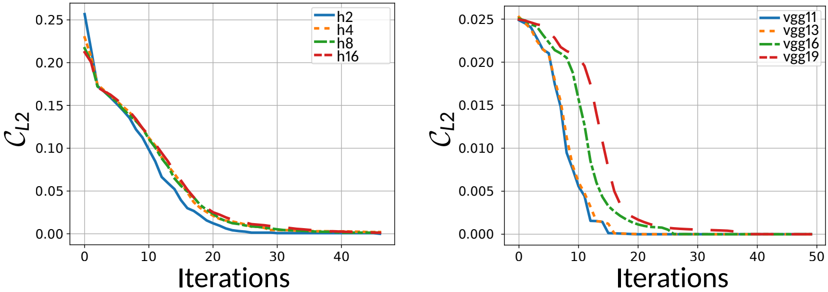
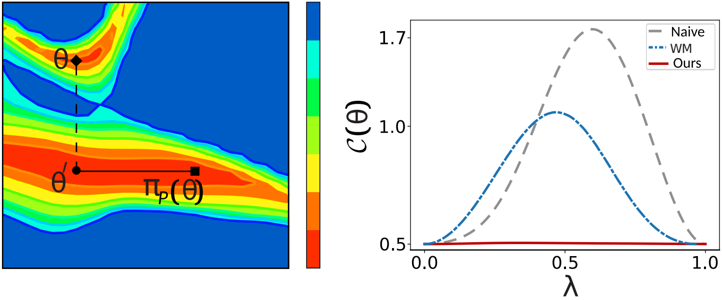



Figure: Validation loss during Sinkhorn re-basin training for feedforward neural networks with a different number of hidden layers (left panel) and VGG with increasing depth (right panel).

# Experiment 2 – Linear mode connectivity




Can we find the permutations that minimize the barrier in the linear path?

# Experiment 2 – Linear mode connectivity

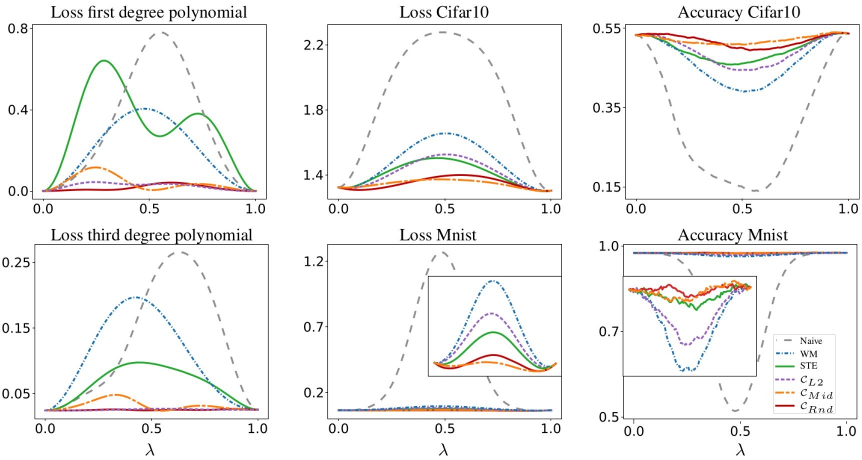



Figure: Linear mode connectivity achieved by WM [Ainsworth et al., 2022], STE [Ainsworth et al., 2022], and our Sinkhorn re-basin with $\mathcal{C}_{L2}$, $\mathcal{C}_{Mid}$, and $\mathcal{C}_{Rnd}$ costs for a NN with two hidden layers.

# Experiment 2 – Linear mode connectivity

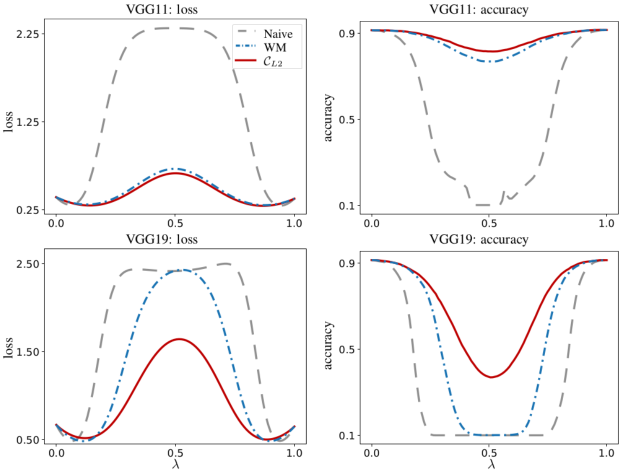



Figure: Linear mode connectivity using VGG11 and VGG19 networks trained over Cifar10 dataset.

# Experiment 2 – Linear mode connectivity

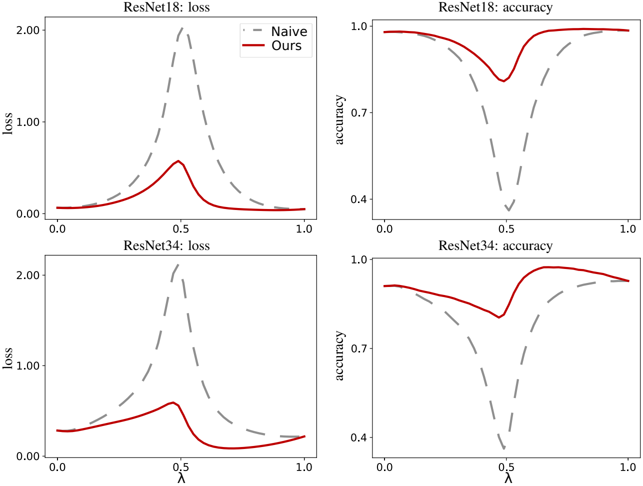
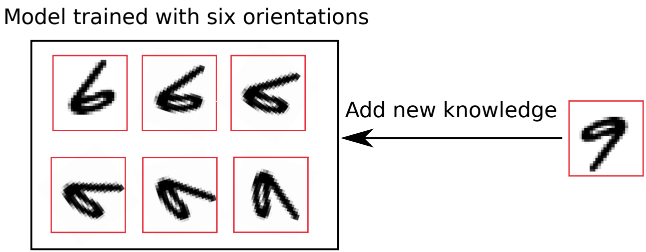



Figure: Linear mode connectivity using ResNet18 and ResNet34 trained over Imagenette dataset.

# Experiment 3 – Incremental learning




Can we find the permutation that allows us to finetune the model and add new knowledge while avoiding catastrophic forgetting?

# Experiment 3 – Incremental learning

| | Rotated Mnist | | Split Cifar100 | |
|---|---|---|---|---|
| Method | Accuracy ↑ | Forgetting ↓ | Accuracy ↑ | Forgetting ↓ |
| Finetune | 46.28±1.01 | 0.52±0.01 | 35.41±0.95 | 0.49±0.01 |
| EWC | 59.92±1.71 | 0.34±0.02 | 50.50±1.33 | 0.24±0.02 |
| LwF | 61.86±3.66 | 0.29±0.06 | 41.43±4.06 | 0.51±0.01 |
| A-GEM | 68.47±0.90 | 0.28±0.01 | 44.42±1.46 | 0.36±0.01 |
| Rebasin (Ours) | 78.14±0.50 | 0.12±0.01 | 51.34±0.74 | 0.07±0.02 |
| Joint training | 90.84±4.30 | 0.00 | 60.48±0.54 | 0.00 |

Table: Performance of our proposed and state-of-art methods on incremental learning benchmarks over 20 episodes.

# Experiment 3 – Incremental learning

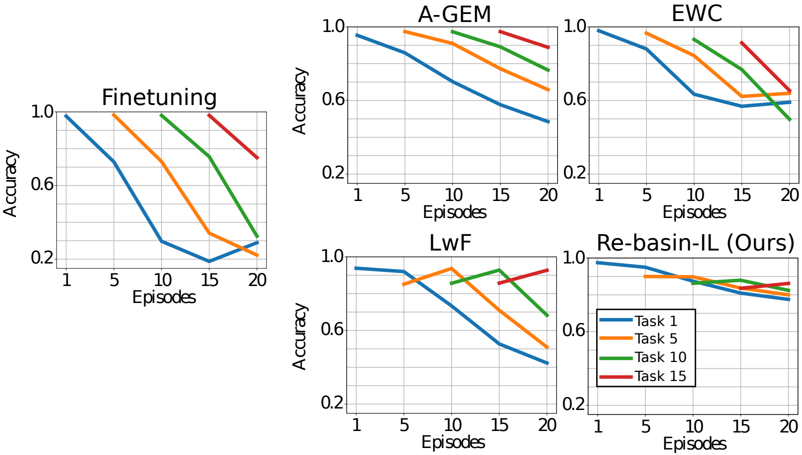



Figure: Evolution of task's accuracy during the incremental learning experience on Rotated Mnist with 20 episodes. Only tasks 1, 5, 10, and 15 are shown.

# Conclusions

# Conclusions

- Gradient descent-based re-basin
- Easy to adapt to new objectives
- New loss functions for neuron alignment, linear mode connectivity, and incremental learning
- Simple to use (PyTorch only!)

```
from rebasin import RebasinNet
modelA = MLP(input_size = 28 * 28, num_classes = 10)
pi_modelA = RebasinNet(modelA, input_shape = (1, 28 * 28))
```

# Thank you!

Visit our project website!

[Ainsworth et al., 2022] Ainsworth, S. K., Hayase, J., and Srinivasa, S. (2022).
Git re-basin: Merging models modulo permutation symmetries.
*arXiv preprint arXiv:2209.04836*.

[Eisenberger et al., 2022] Eisenberger, M., Toker, A., Leal-Taixé, L., Bernard, F., and Cremers, D. (2022).
A unified framework for implicit sinkhorn differentiation.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 509–518.

[Mena et al., 2018] Mena, G., Belanger, D., Linderman, S., and Snoek, J. (2018).
Learning latent permutations with gumbel-sinkhorn networks.
*arXiv preprint arXiv:1802.08665*.