Symmetry-Aware Graph Metanetwork Autoencoders: Model Merging through Parameter Canonicalization

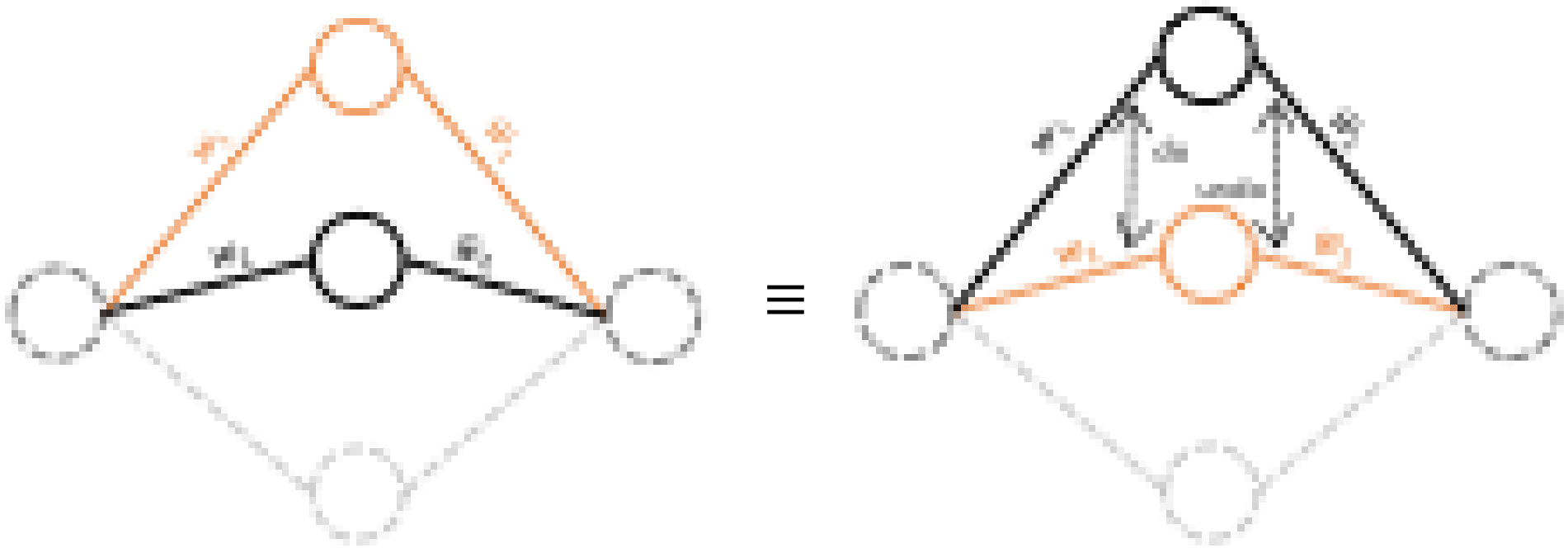
Eduardo Terrés Caballero

eduardo.terres.caballero@student.uva.nl

December 2, 2025

Weight Space Symmetries: *Permutation*

The underlying function that a Neural Network represents allows for **equivalent** representations, derived from the **symmetries of its graph structure**.



$$\theta = (W_l,\ b_l)_{l=1}^{L}$$

$$\theta_P = (P_l W_l P_{l-1}^{-1},\ P_l b_l)_{l=1}^{L}$$

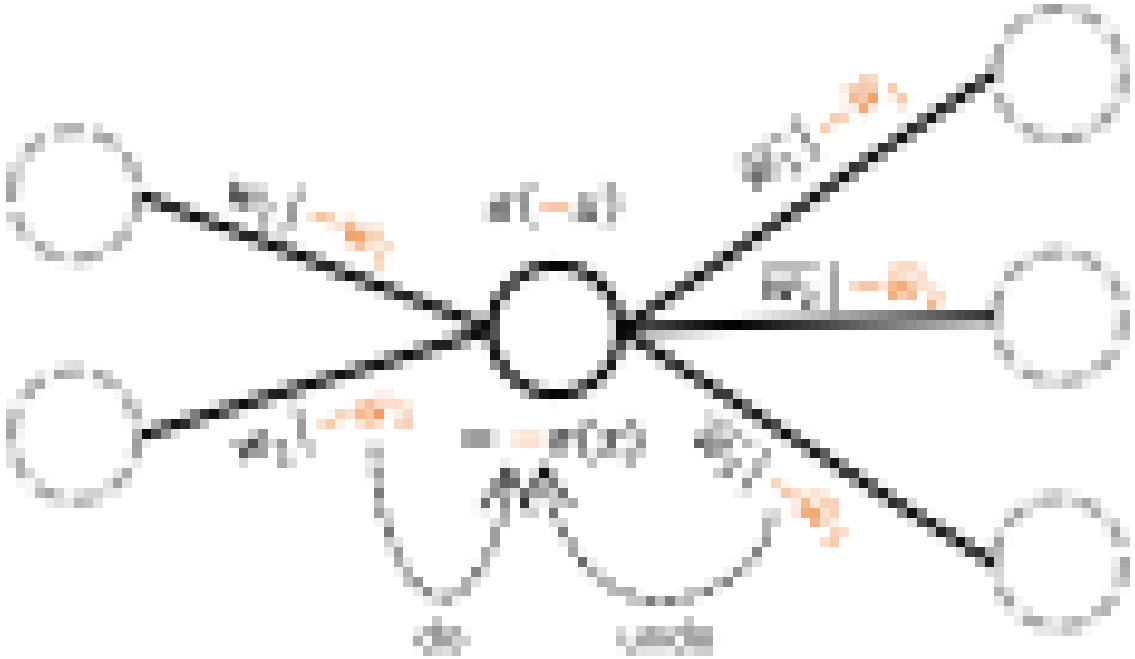
$P_l \in S_{d_l}$, permutation matrix.

Weight Space Symmetries: *Scaling*

Activation functions also induce a set of **equivalent** networks due to scaling symmetries.

ReLU
For $\alpha > 0$, $\mathrm{ReLU}(\alpha \cdot x) = \alpha \cdot \mathrm{ReLU}(x)$.

Tanh
For $\alpha \in \{-1, 1\}$, $\tanh(\alpha \cdot x) = \alpha \cdot \tanh(x)$.



We account for scaling symmetries in conjunction with permutations.

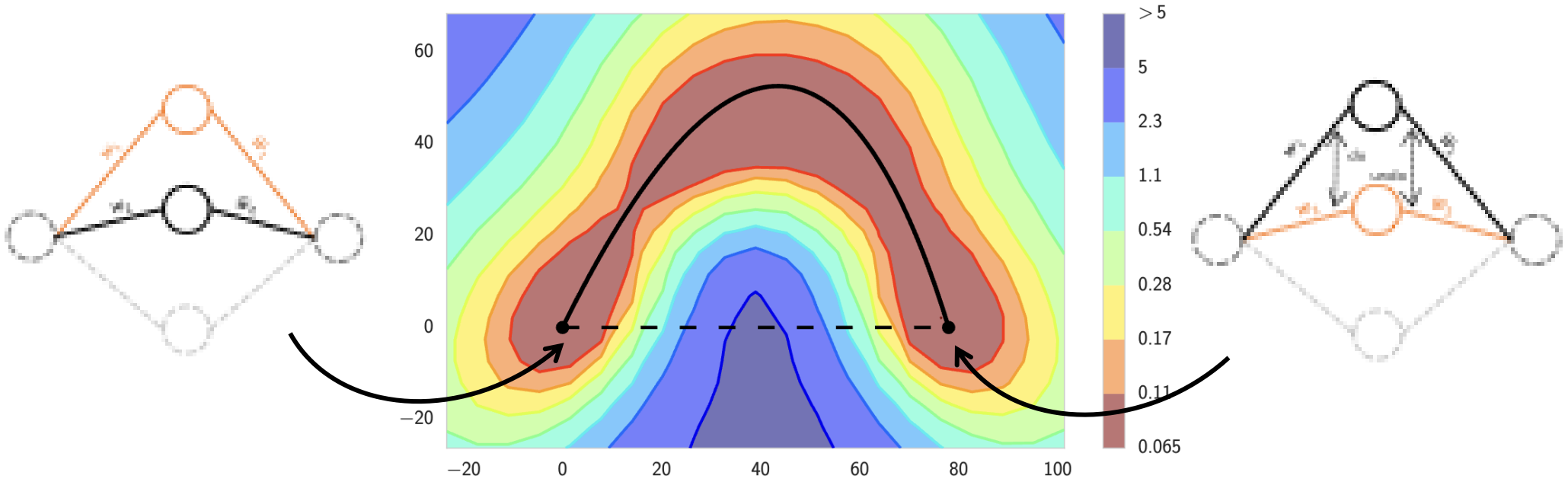
$$\theta_{PQ} = (P_l Q_l W_l Q_{l-1}^{-1} P_{l-1}^{-1},\ P_l Q_l b_l)_{l=1}^{L}$$

$$P_l \in S_{d_l},\ Q_l = \mathrm{diag}(q_1, \ldots, q_{d_l}),\ q_k = \pm 1$$

Model merging

Model merging combines the weights of two models into a single model that inherits their capabilities.

A fundamental problem is that the loss landscape is filled with **equivalent local minima** due to symmetries.



Loss landscape derived from the loss used to train the NN.
Figure 1 from Garipov et al.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In: NeurIPS 2018.

Model merging: **Git Re-Basin**

Git Re-Basin finds a transformation for the weights of network B that brings it into the basin of network A by solving a linear assignment problem using the Hungarian algorithm.

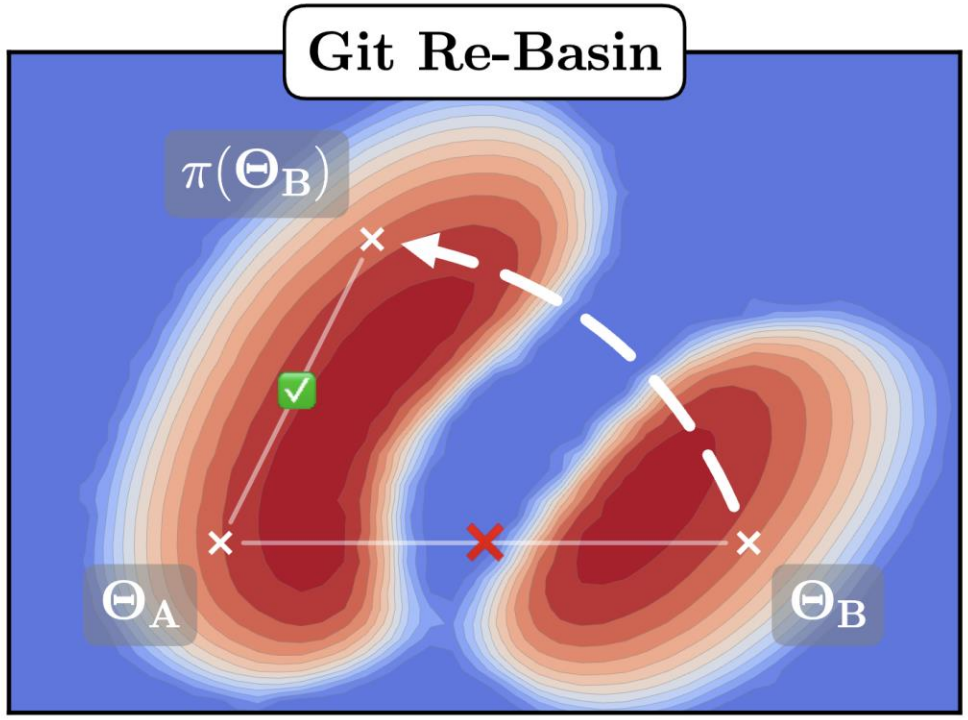


Figure 1 from Ainsworth et al. [3]

Frobenius matrix norm

$$\langle \boldsymbol{A}, \boldsymbol{A} \rangle_F = \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2$$

$$\arg\max_{\boldsymbol{T}_\ell} \left(\begin{array}{l} \langle \boldsymbol{W}_\ell^{(\mathrm{A})}, \boldsymbol{T}_\ell \boldsymbol{W}_\ell^{(\mathrm{B})} \boldsymbol{T}_{\ell-1}^\top \rangle_F + \\ \langle \boldsymbol{W}_{\ell+1}^{(\mathrm{A})}, \boldsymbol{T}_{\ell+1} \boldsymbol{W}_{\ell+1}^{(\mathrm{B})} \boldsymbol{T}_\ell^\top \rangle_F + \\ \langle \boldsymbol{b}_\ell^{(\mathrm{A})}, \boldsymbol{T}_\ell \boldsymbol{b}_\ell^{(\mathrm{B})} \rangle_F \end{array} \right)$$

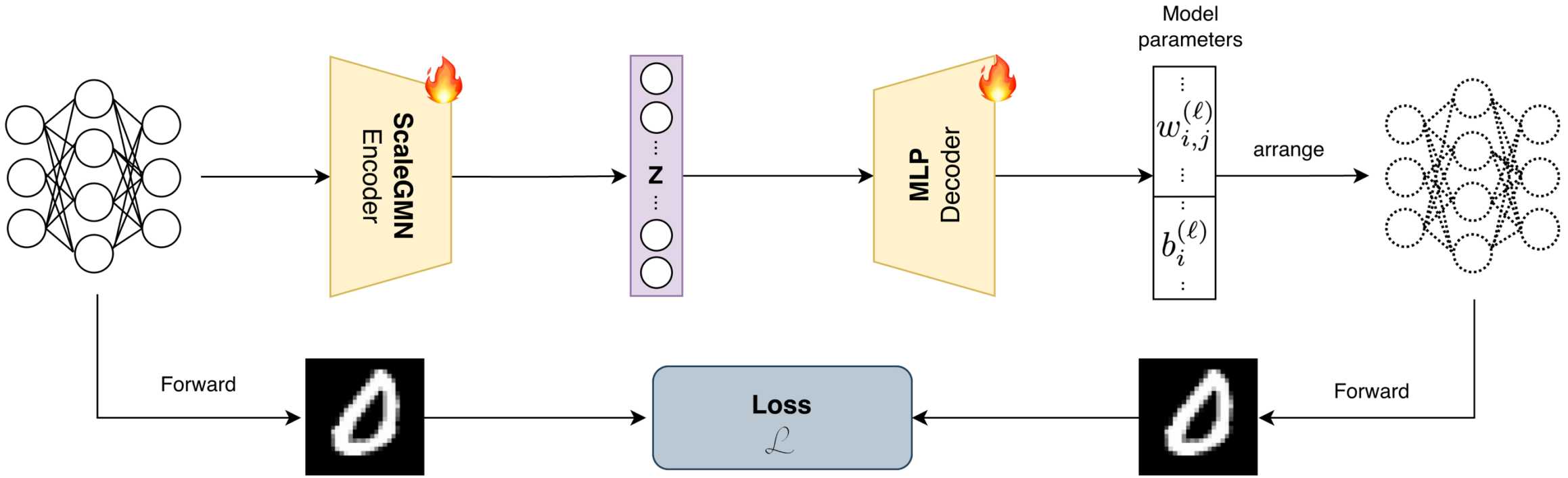
We extend Git Re-Basin to also account for ± 1 scaling symmetries (i.e. $\mathbf{T}_\ell = \mathbf{P}_\ell \mathbf{Q}_\ell$).

Samuel K. Ainsworth et al. Git re-basin: Merging models modulo permutation symmetries, 2023.

Model merging: **An autoencoding approach**

We propose an autoencoding approach to model merging.

- The **encoder** maps all networks of a group orbit to the same latent vector.
- The **decoder** provides a learned canonical representation of this latent vector



The input: CNNs

Symmetries in CNNs[1]

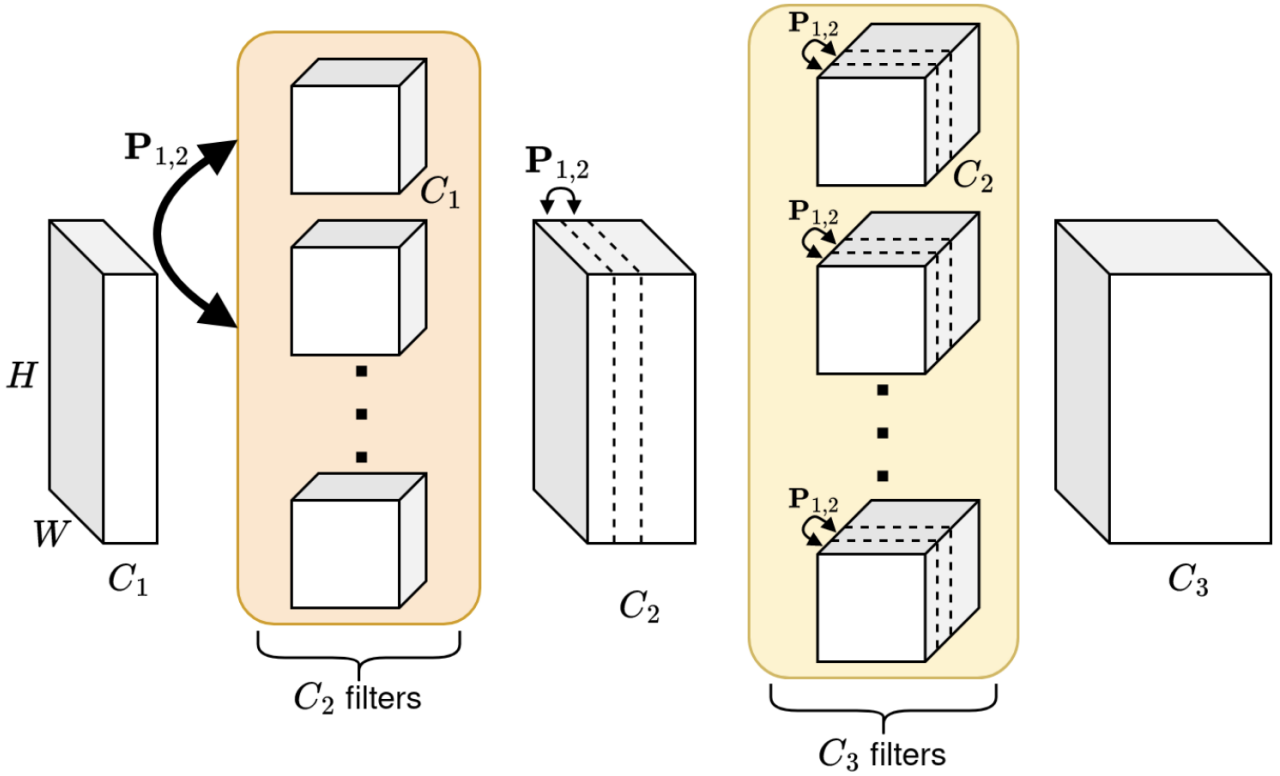


Figure 5, Appendix C, from Kofinas et al. (2024) [2].

[1] Miltiadis Kofinas et al. Graph neural networks for learning equivariant representations of neural networks. In: ICLR, 2024.
[2] Thomas Unterthiner et al. Predicting neural network accuracy from weights, 2021.

The input: CNNs as GNNs

Symmetries in CNNs

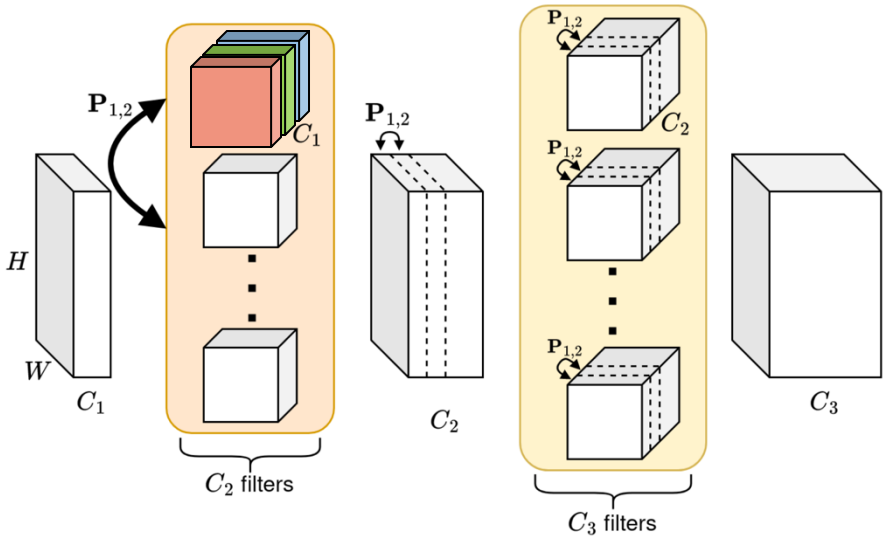
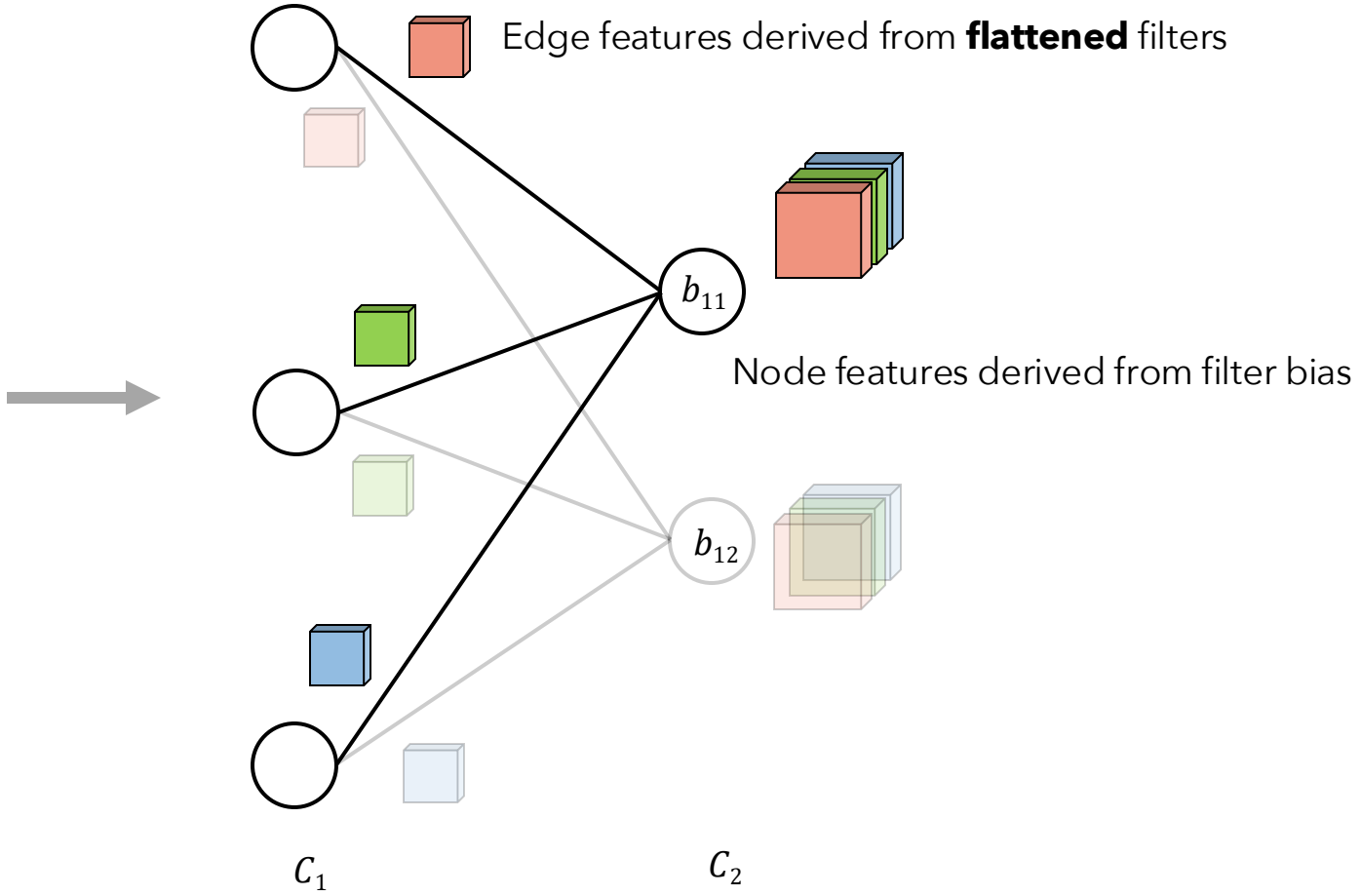


Figure 5, Appendix C, from Kofinas et al. (2024).



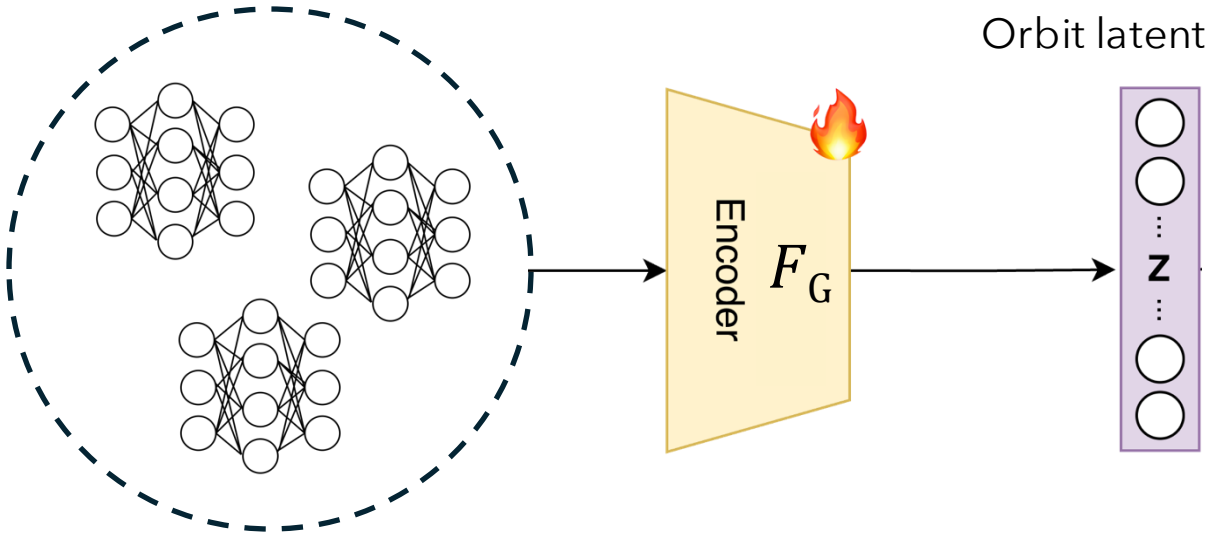
[1] Miltiadis Kofinas et al. Graph neural networks for learning equivariant representations of neural networks. In: ICLR, 2024.

The encoder: Symmetry-aware Graph Metanetworks

Graph MetaNetwork: A neural network that uses a GNN to process another network's structure.

Symmetry aware

Let G be a fixed computation graph.
Define the encoder as $F_G : \{(G, \theta) : \theta \text{ arbitrary model params.}\} \to \mathbb{R}^D$



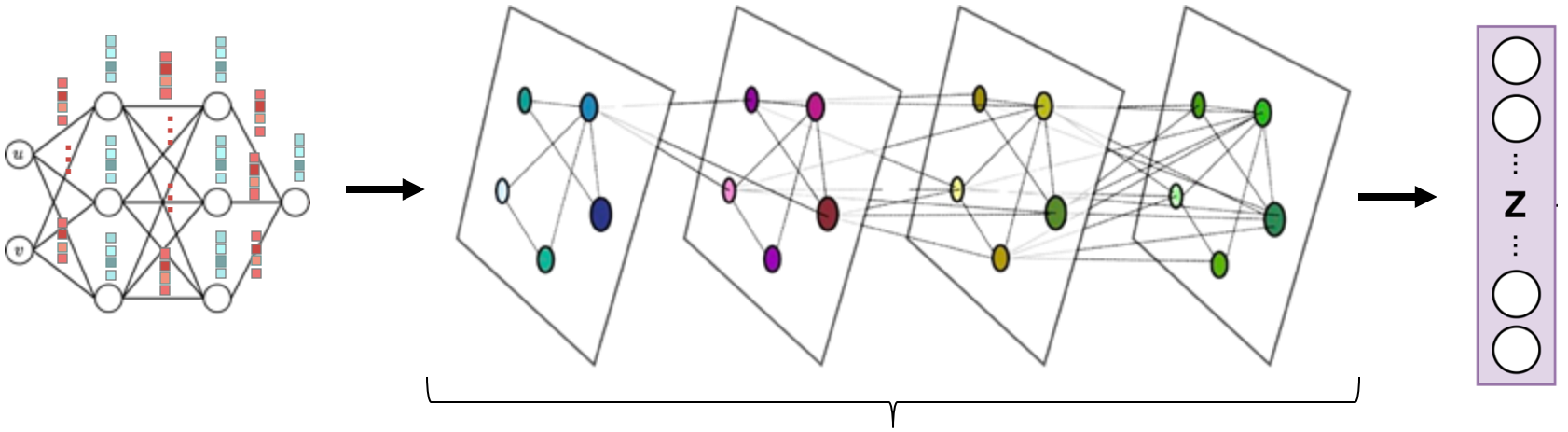
Let $\mathcal{G}$ be the group of symmetries of interest.
Group Orbit: $\mathrm{Orb}_{\mathcal{G}}(G, \theta) = \{(G, g \cdot \theta) \mid \forall g \in \mathcal{G}\}$

The encoder: Symmetry-aware Graph Metanetworks

We use 2 symmetry aware encoders:

① Neural Graphs[1] (Permutation)

② ScaleGMN[2] (Permutation + Scaling)



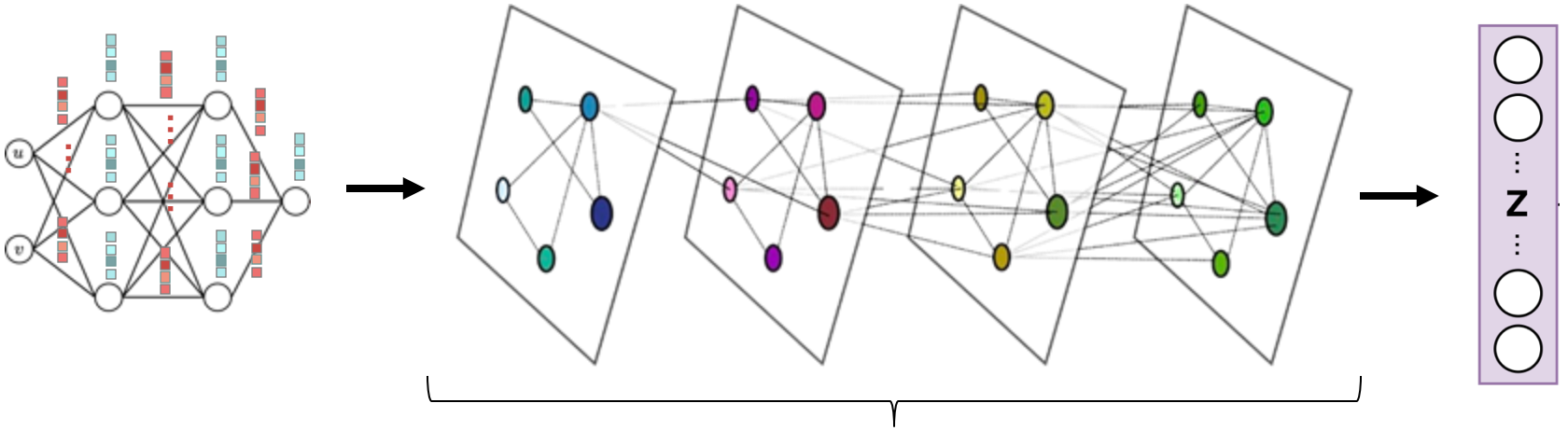
[1] Miltiadis Kofinas et al. Graph neural networks for learning equivariant representations of neural networks. In: ICLR, 2024.
[2] Ioannis Kalogeropoulos, Giorgos Bouritsas, and Yannis Panagakis. Scale equivariant graph metanetworks. In: NeurIPS 2024.

The encoder: Symmetry-aware Graph Metanetworks

We use 2 symmetry aware encoders:

① Neural Graphs[1] (Permutation)

② ScaleGMN[2] (Permutation + Scaling)

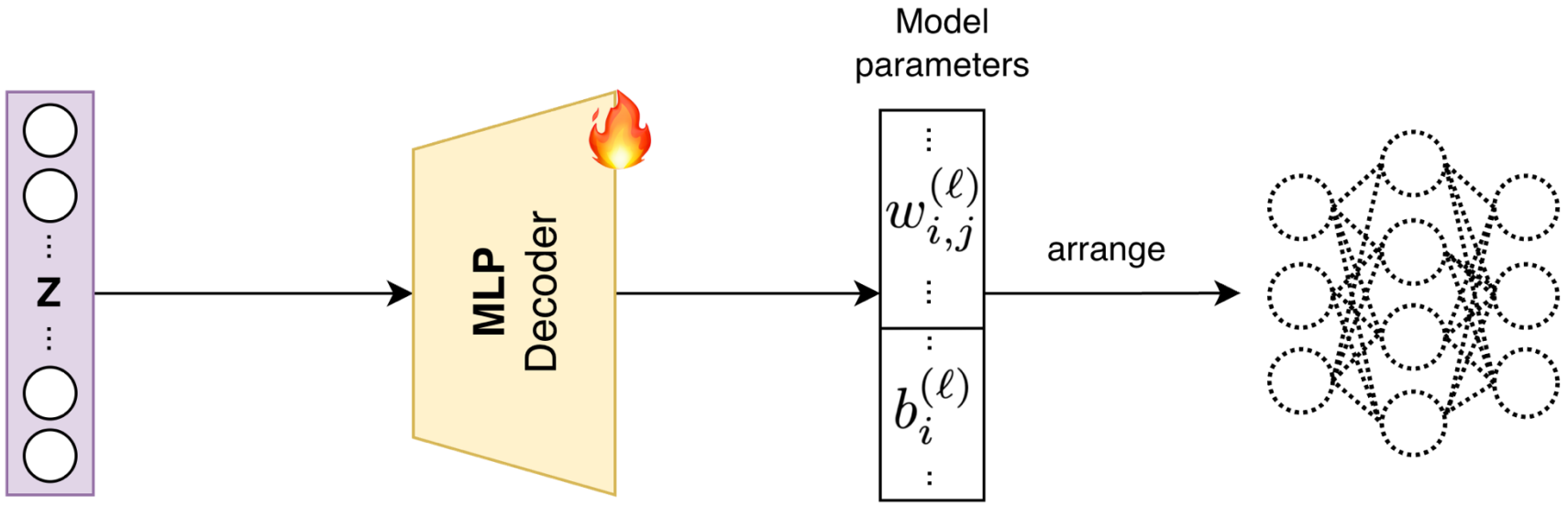


How do ScaleGMNs achieve scale invariance?

- For sign symmetries, compute both $canon(x) = MLP(x) + MLP(-x) = canon(-x)$.
- For continuous scaling symmetries, canonicalization $canon(x) = x / ||x||$.

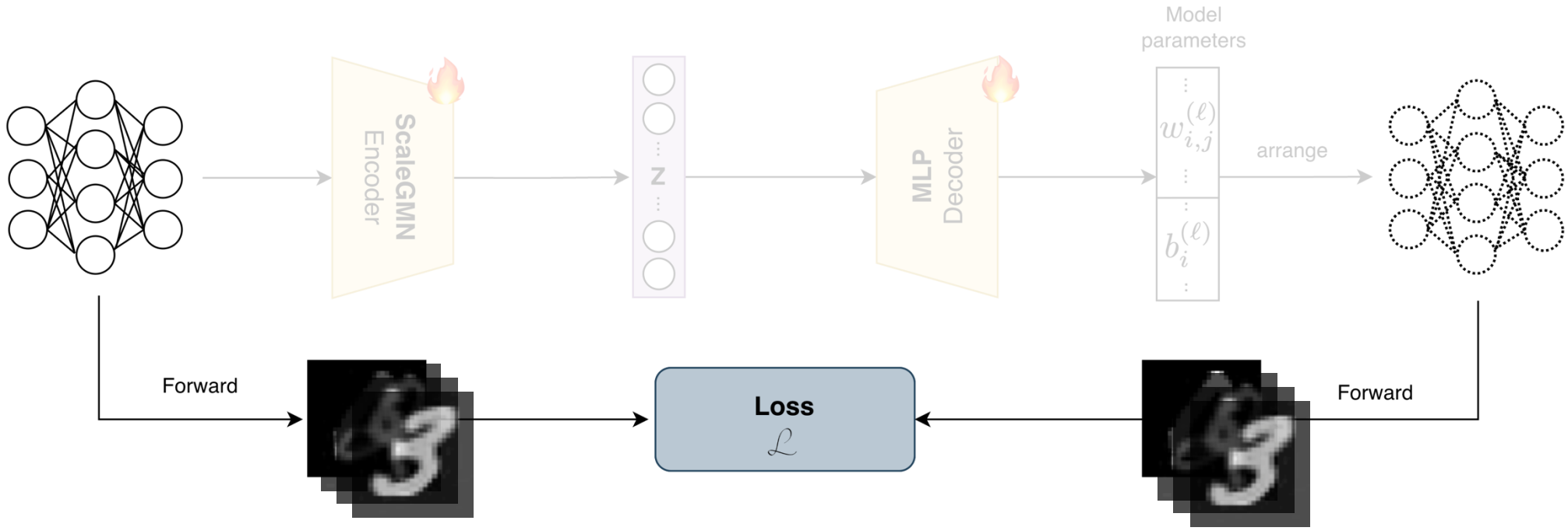
The decoder: MLP

We use a simple MLP decoder to obtain a canonicalized version of the weights.



The Loss: Functional Reconstruction

To ensure maintaining performance of the reconstructed network we use a functional loss.



For CNN classifiers we forward the **input and reconstructed networks** over a dataset and **compare the class logits** (KL divergence).

Experiment: CNN

We want to show that the autoencoder maps independently trained CNNs to a common basin, allowing for a low-loss-barrier linear interpolation.

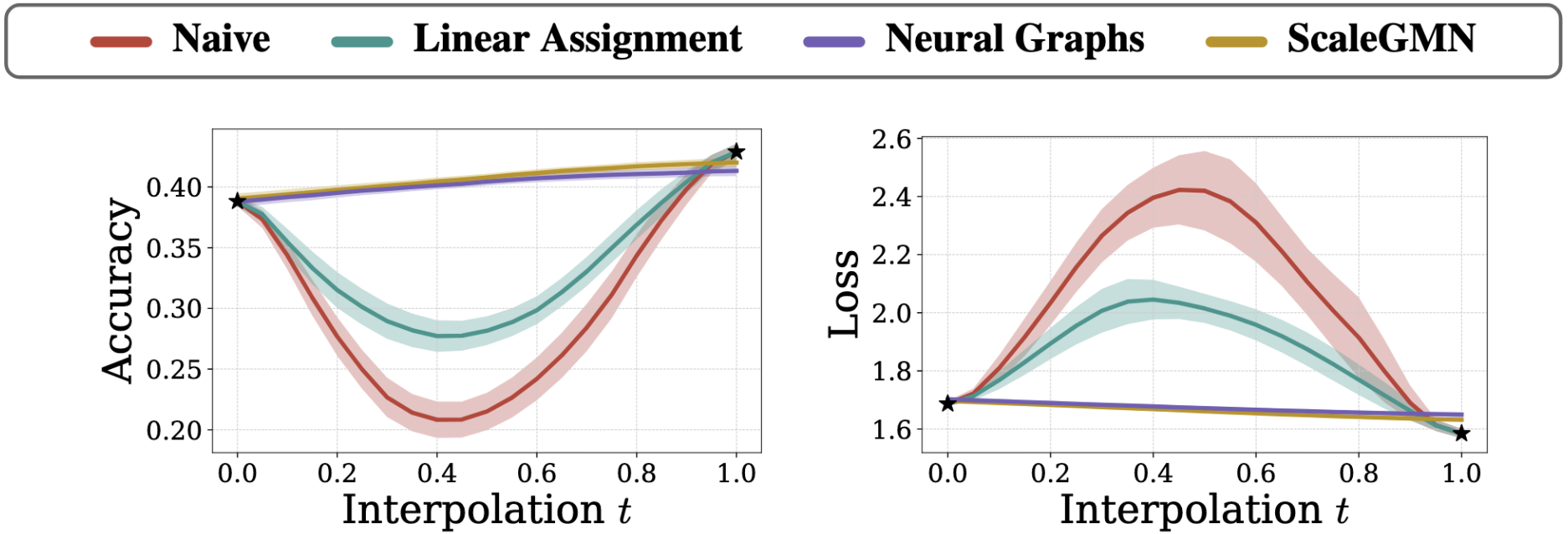
Steps of the experiment:

1. We take **two** test set **CNNs** trained on a CIFAR train split.
2. **Forward pass** them through the autoencoder architecture.
3. Perform **linear interpolation** in weight space.
4. Evaluate canonicalized CNNs on the CIFAR test split.

Experiment: CNN

Naïve interpolation and Linear assignment incur in a high loss barrier

Autoencoding allows for a straight interpolation curve but provides worse reconstruction.



(a) Accuracy (b) Cross Entropy Loss

Average interpolation curves (and standard deviation) over 20 pairs of distinct CNN models with Tanh activation.

Recap and Pros&Cons

We propose a symmetry-aware autoencoding framework that canonicalizes neural networks by collapsing symmetry orbits.

Pros

1. The autoencoder has a linear inference cost w.r.t. the number of parameters of the input. Git Re-Basin has super-linear cost.
2. The interpolation barrier is lower with autoencoding.

Cons

1. Requires training a model, hence enough data is required.
2. Reconstruction loss must keep performance.

Future Work: Extend the framework (encoder) to larger architectures (e.g. Transformers).

For Additional Questions
Please Contact:

a.garciacastellanos@uva.nl

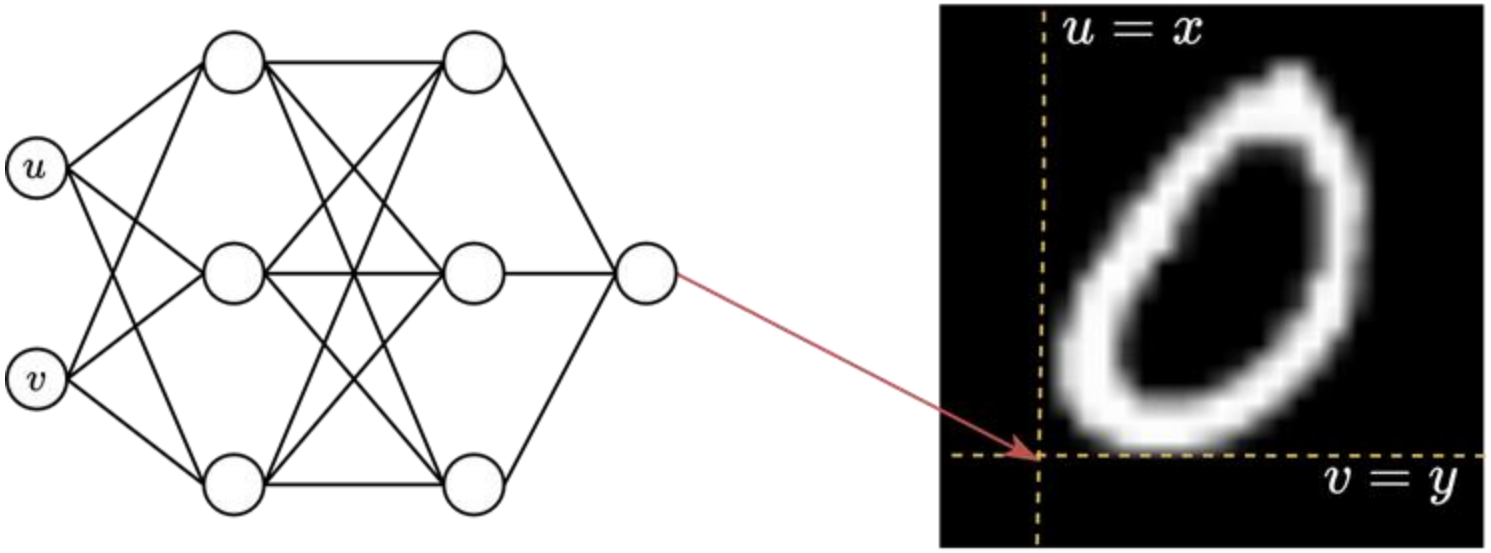
Appendix

The input: Graph Neural Networks

INR: A Neural Network that gives the pixel intensity (output) on a pixel (input).

INRs

(Implicit Neural Representations)



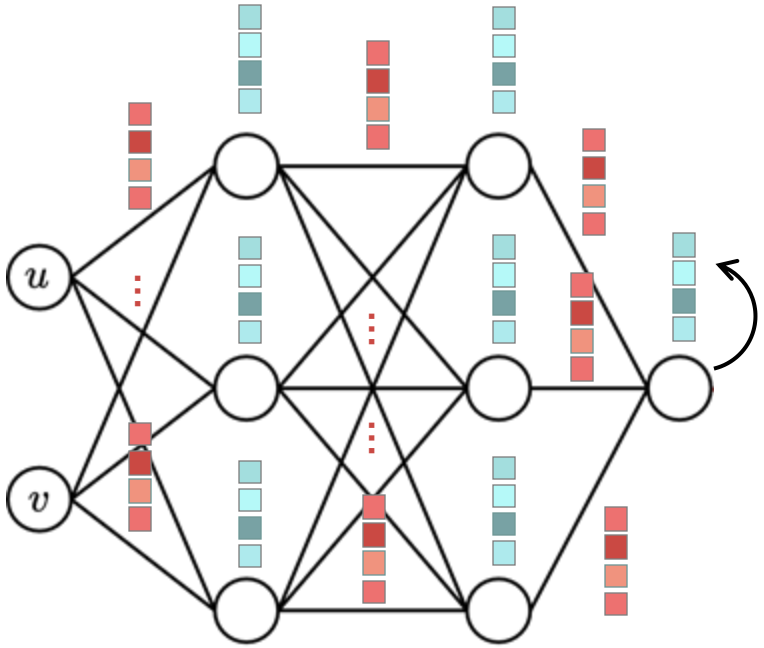
The input: INRs

INR*: A Neural Network that gives the pixel intensity (output) on a pixel (input).

We transform the INR to a GNN by deriving **edge features** from weights and **node features** from biases.

INR as a GNN: (G, θ)

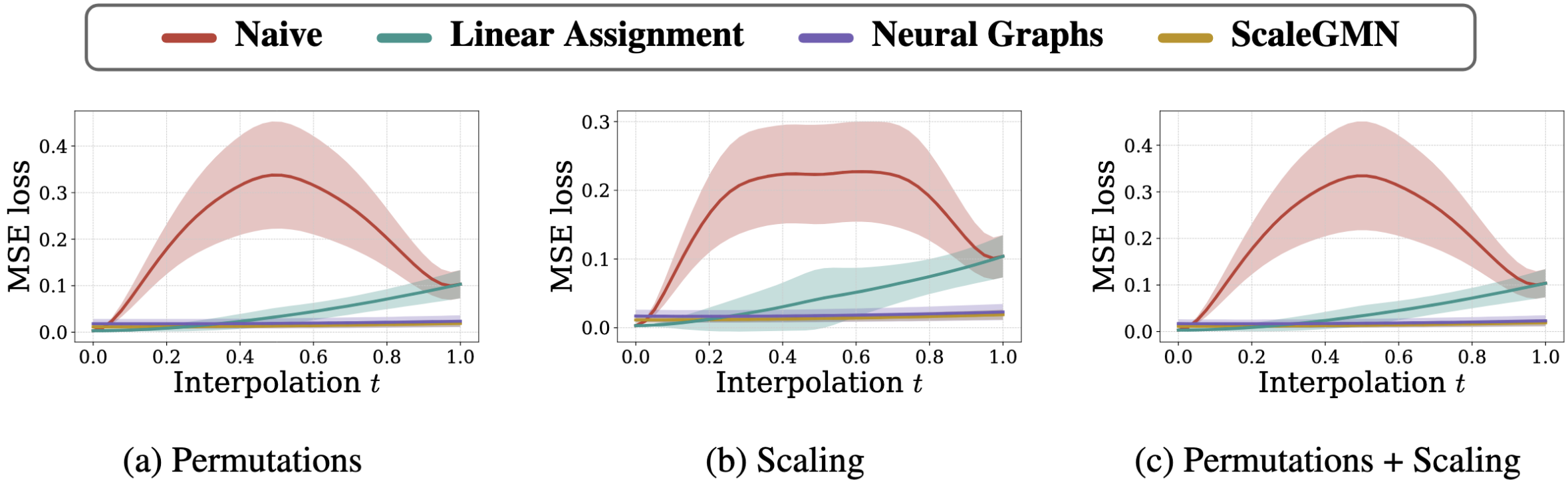
Denote G for the computation graph, θ for the parameters.



(*) Matthias Bauer et al. Spatial functa: Scaling functa to imagenet classification and generation.

We use an **upscaling layer** to convert weights as edge features, biases as node features.

Results: INRs



(a) Permutations

(b) Scaling

(c) Permutations + Scaling