Graph Neural Networks for Learning Equivariant Representations of Neural Networks

The 12th International Conference on Learning Representations, ICLR 2024, Oral








Miltiadis Kofinas[1]* Boris Knyazev[2] Yan Zhang[2] Yunlu Chen[3] Gertjan J. Burghouts[4] Efstratios Gavves[1] Cees G. M. Snoek[1] David W. Zhang[1]*

*Joint first and last authors

1 University of Amsterdam

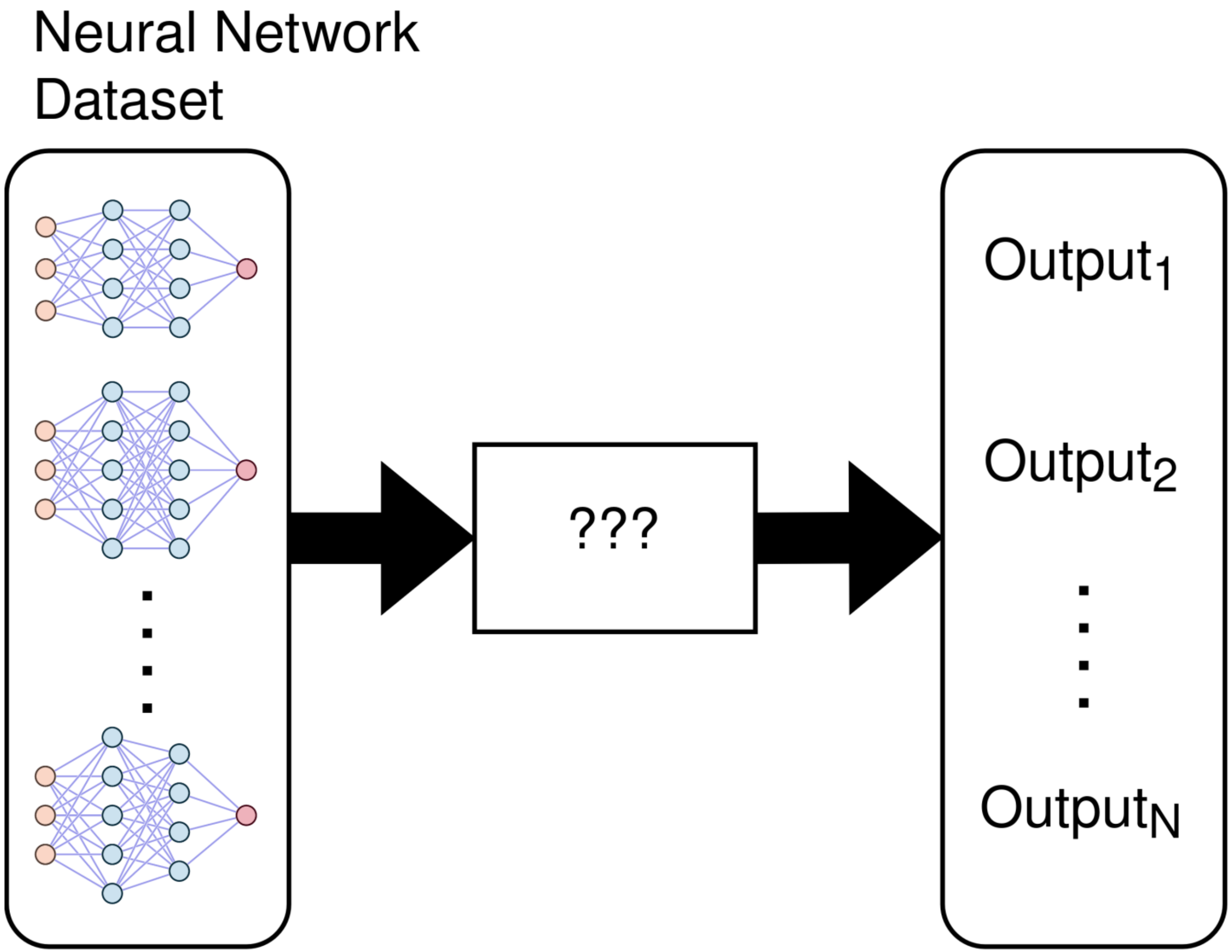
2 SAMSUNG Samsung Advanced Institute of Technology AI Lab Montreal

3 Carnegie Mellon University

4 TNO

ICLR

Problem formulation – Networks for Networks



Implicit Neural Representations (INRs)

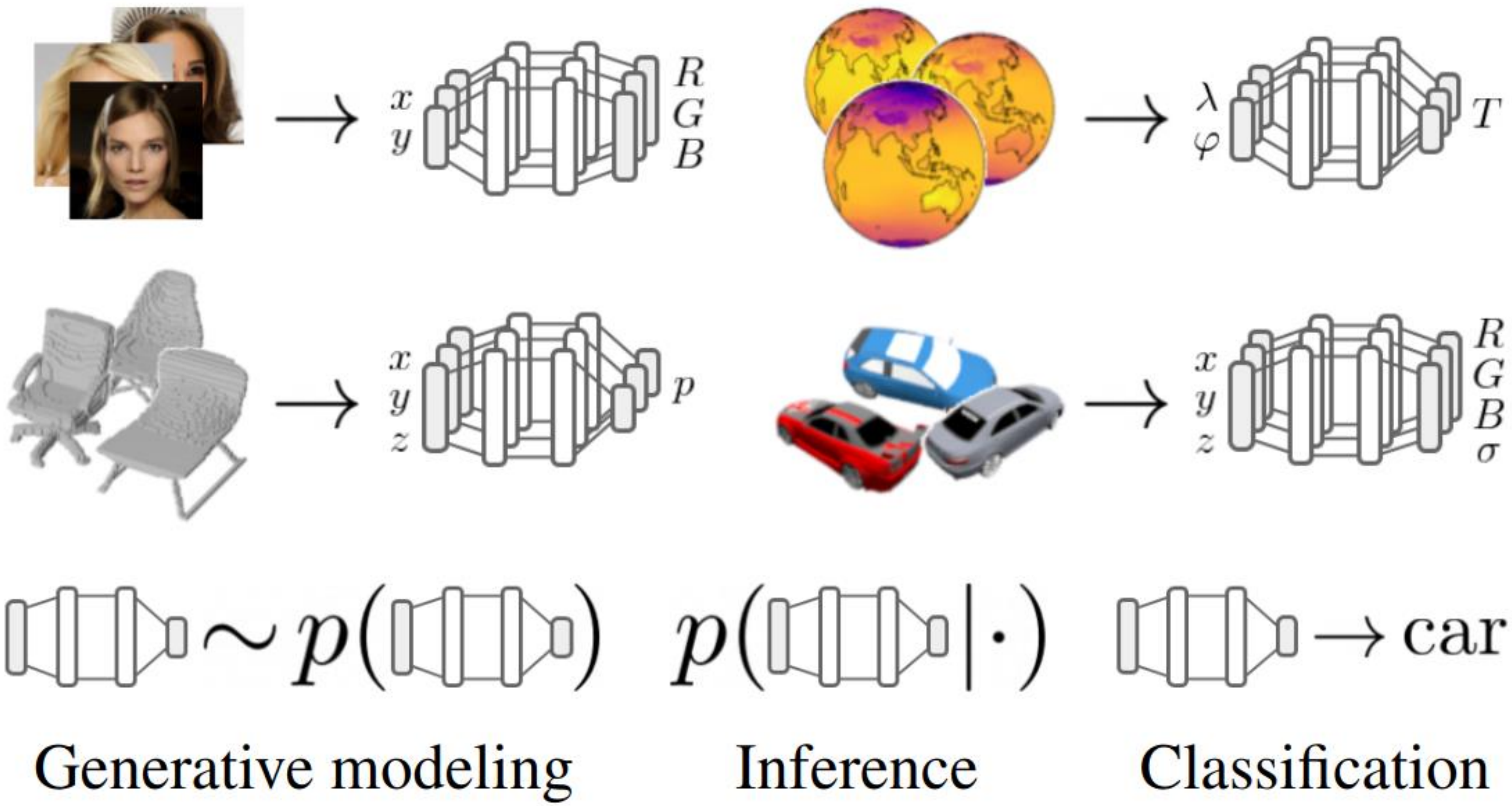
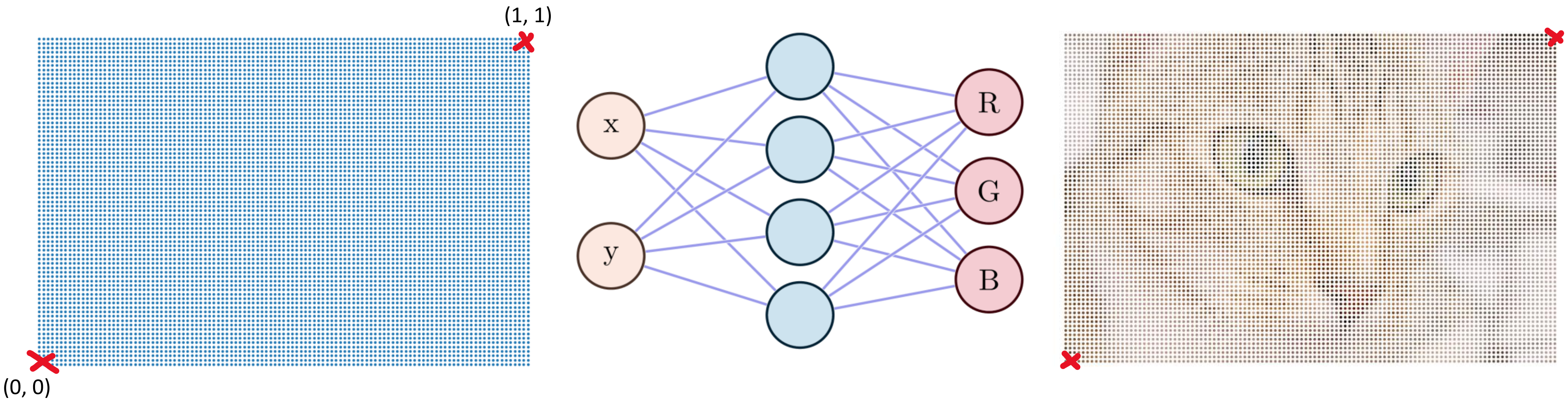


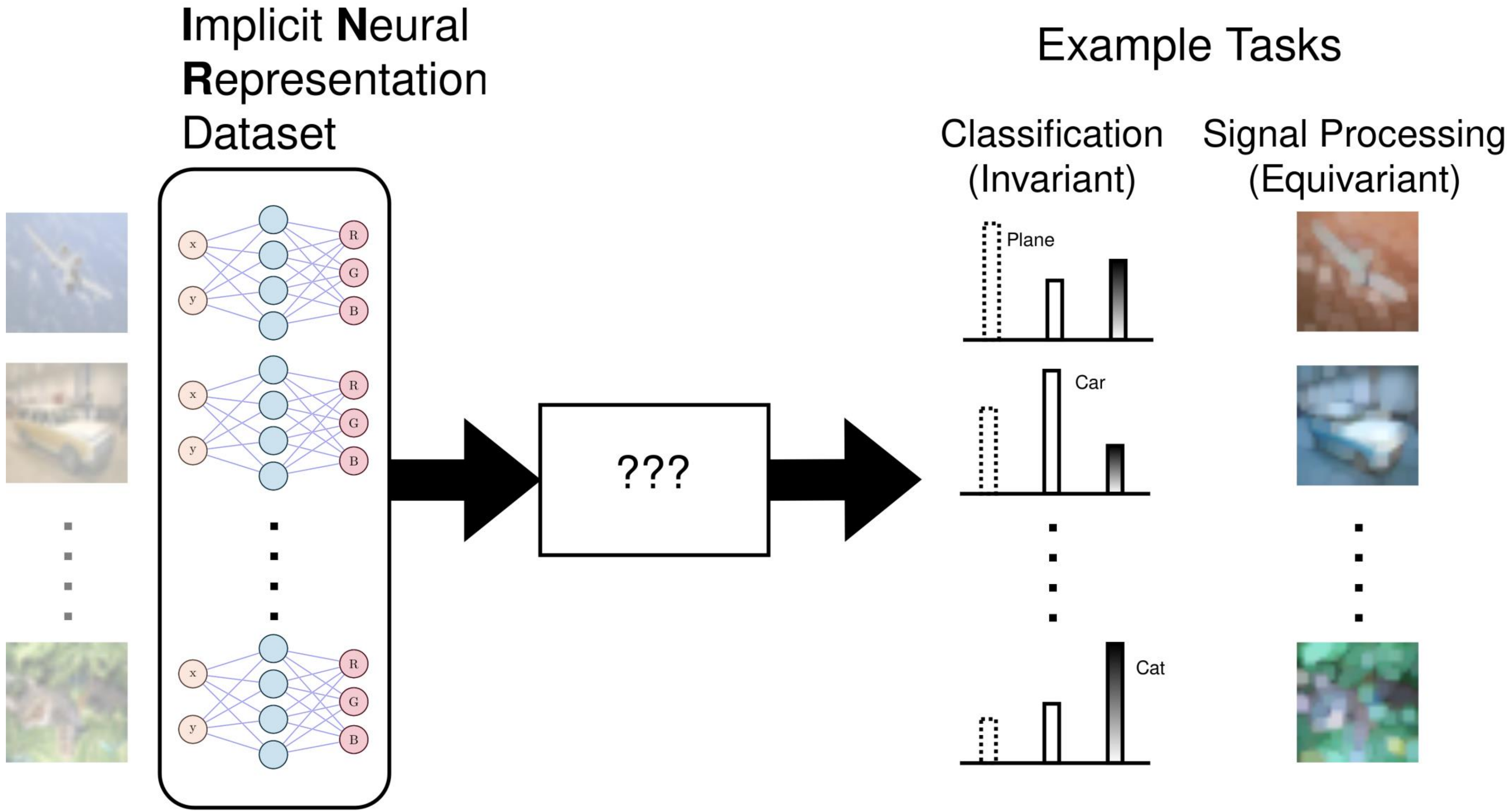
Figure credit: Emilien Dupont*, Hyunjik Kim* et al. “From data to functa: Your data point is a function and you can treat it like one”. In: ICML 2022.

What are INRs?



Query Coordinates

Paradigm shift



Paradigm shift

Traditional Paradigm	Modern Paradigm
• Save signal as an array	• Fit signal with an INR and save its parameters & architecture
• Process with CNNs/ViTs	• Process with parameter space networks

Predict model characteristics

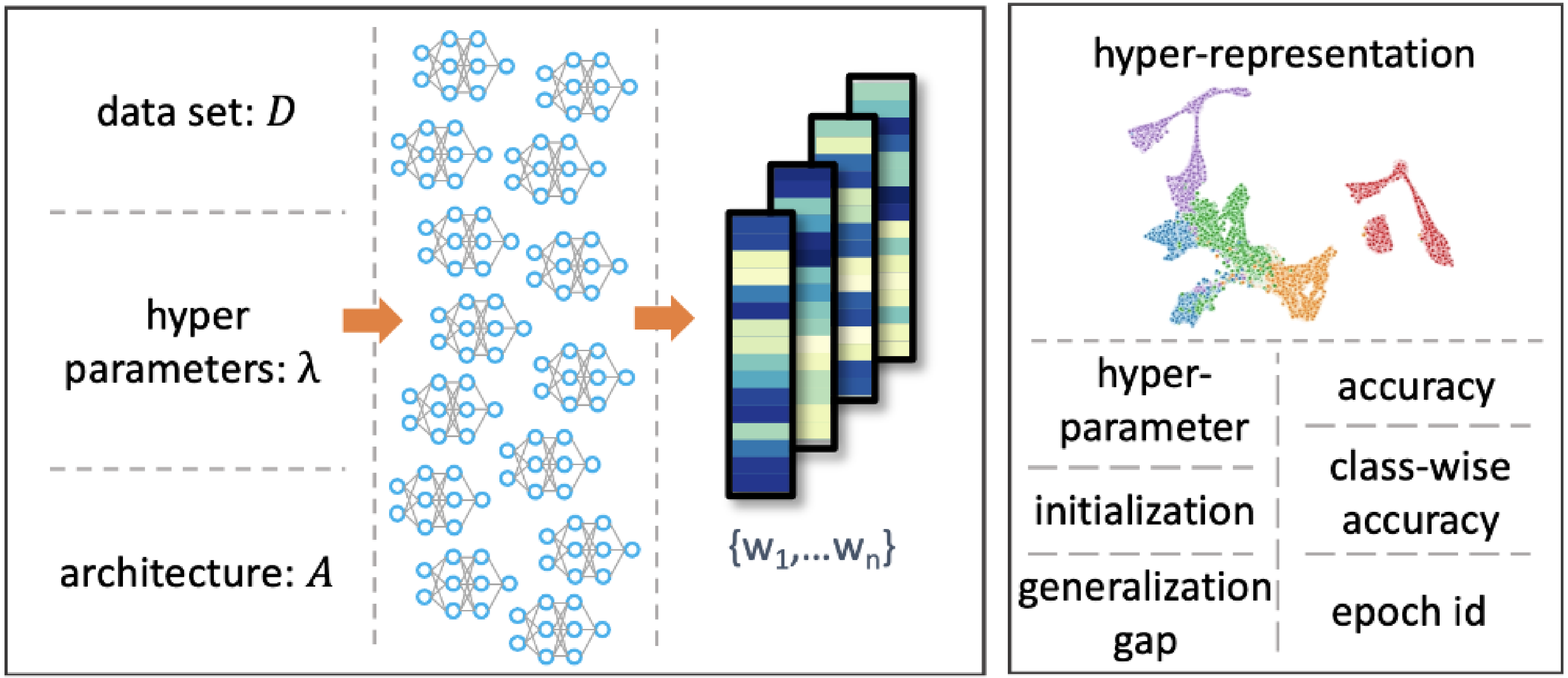


Figure credit: Konstantin Schürholt. “Self-Supervised Representation Learning on Neural Network Weights for Model Characteristic Prediction”. In: NeurIPS 2021.

Generative models of weights

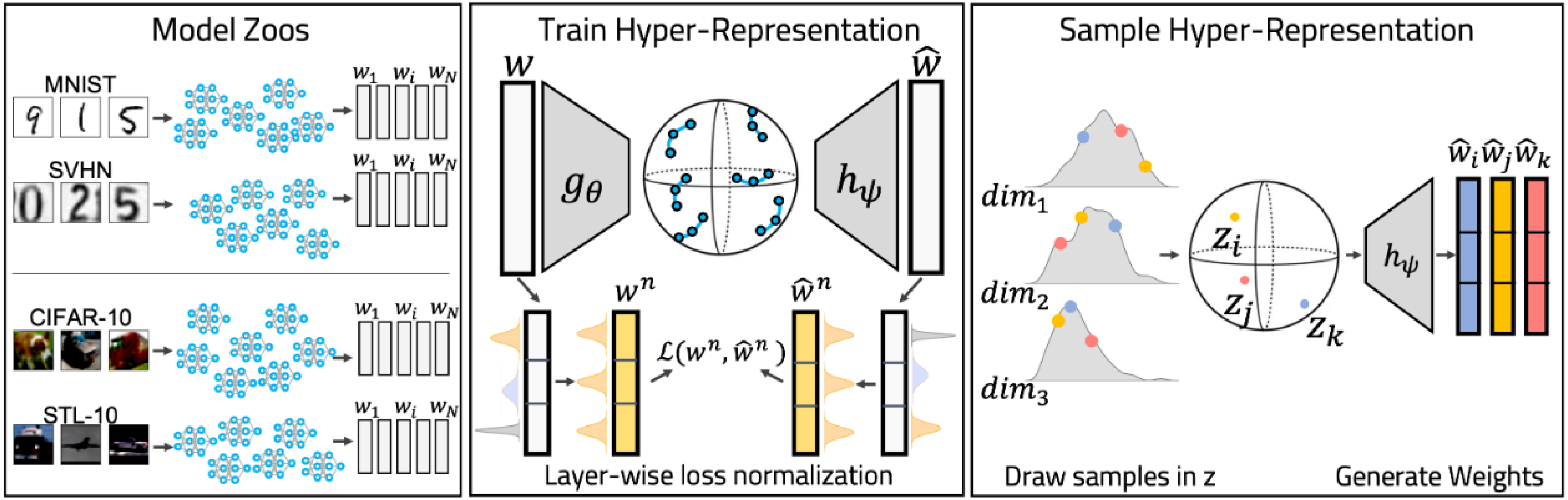
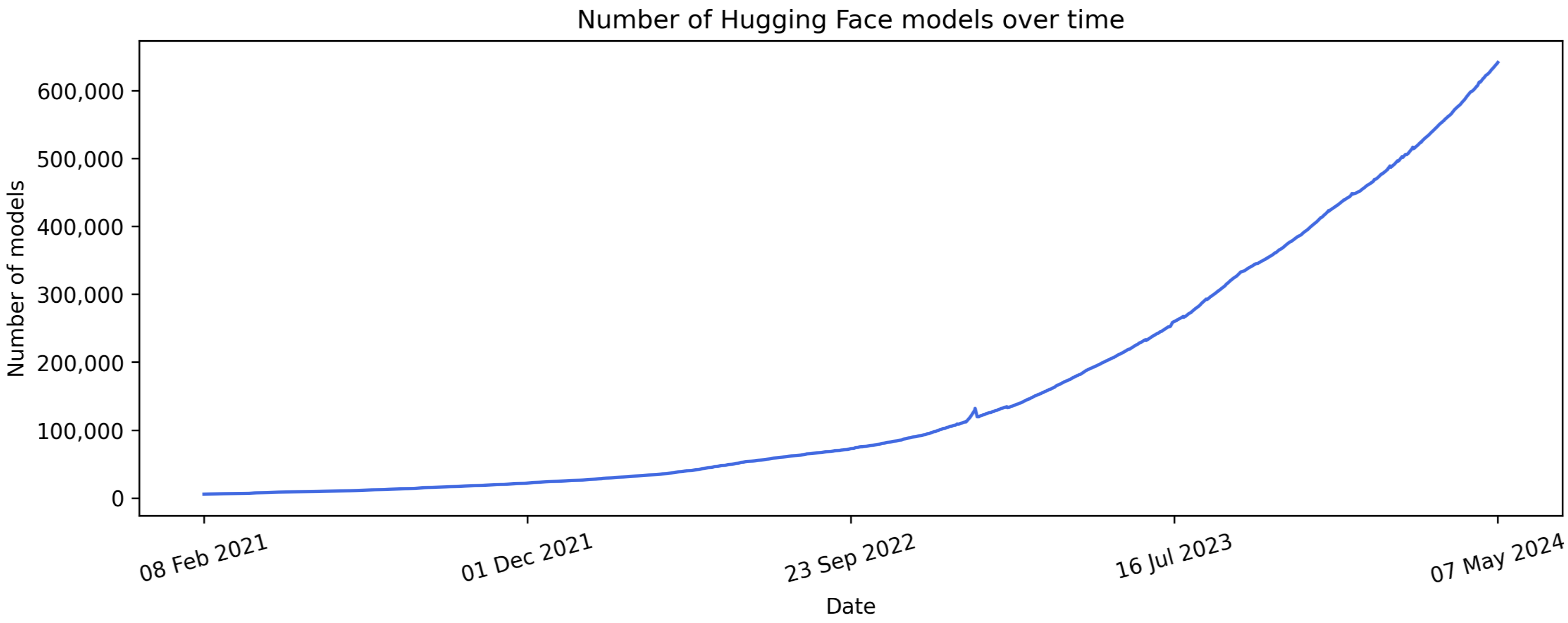


Figure credit: Konstantin Schürholt. “Hyper-Representations as Generative Models: Sampling Unseen Neural Network Weights”. In: NeurIPS 2022.

Neural networks are the new data!



Source: https://huggingface.co/models

Paradigm shift

Traditional Paradigm	Modern Paradigm
• Train NNs with hyperparameter search	• Model zoos & neural networks are the new data! • Generative parameter space network on the weights

Parameter Space Networks – Naïve Approach

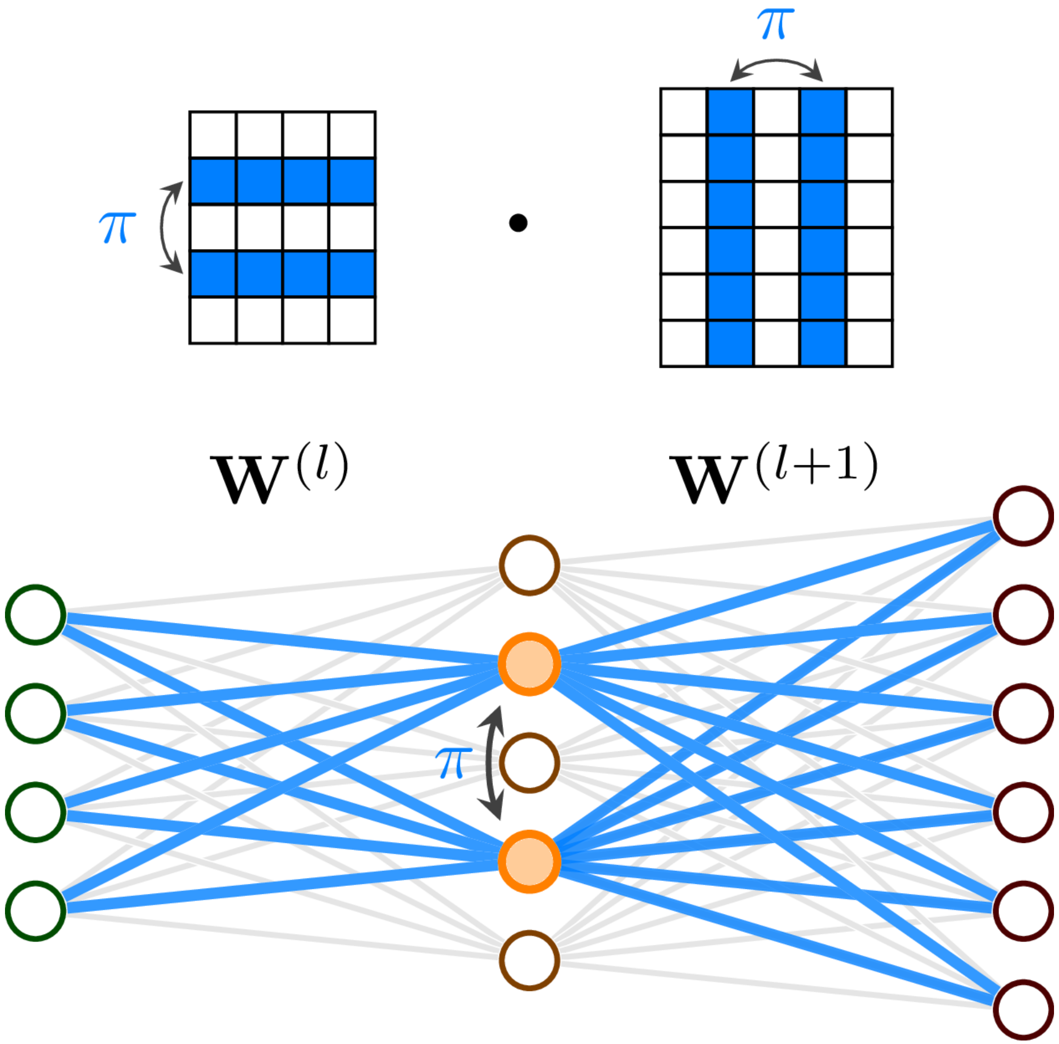
- Flatten parameters (weights/biases) and process them with MLPs

- **Problem**! *Permutation symmetries*

- Naïve MLP achieves 17.6% accuracy on MNIST[1]

[1]Aviv Navon*, Aviv Shamsian* et al. “Equivariant architectures for learning in deep weight spaces”. In: ICML 2023.

Permutation Symmetries



Related works

- Overlook the inherent permutation symmetry

- Rely on intricate weight-sharing patterns to achieve equivariance

- Ignore the network architecture itself, limited to a single architecture

[1] Aviv Navon*, Aviv Shamsian* et al. “Equivariant architectures for learning in deep weight spaces”. In: ICML 2023.
[2] Allan Zhou et al. “Permutation Equivariant Neural Functionals”. In: NeurIPS 2023.

Our approach – Neural Graphs

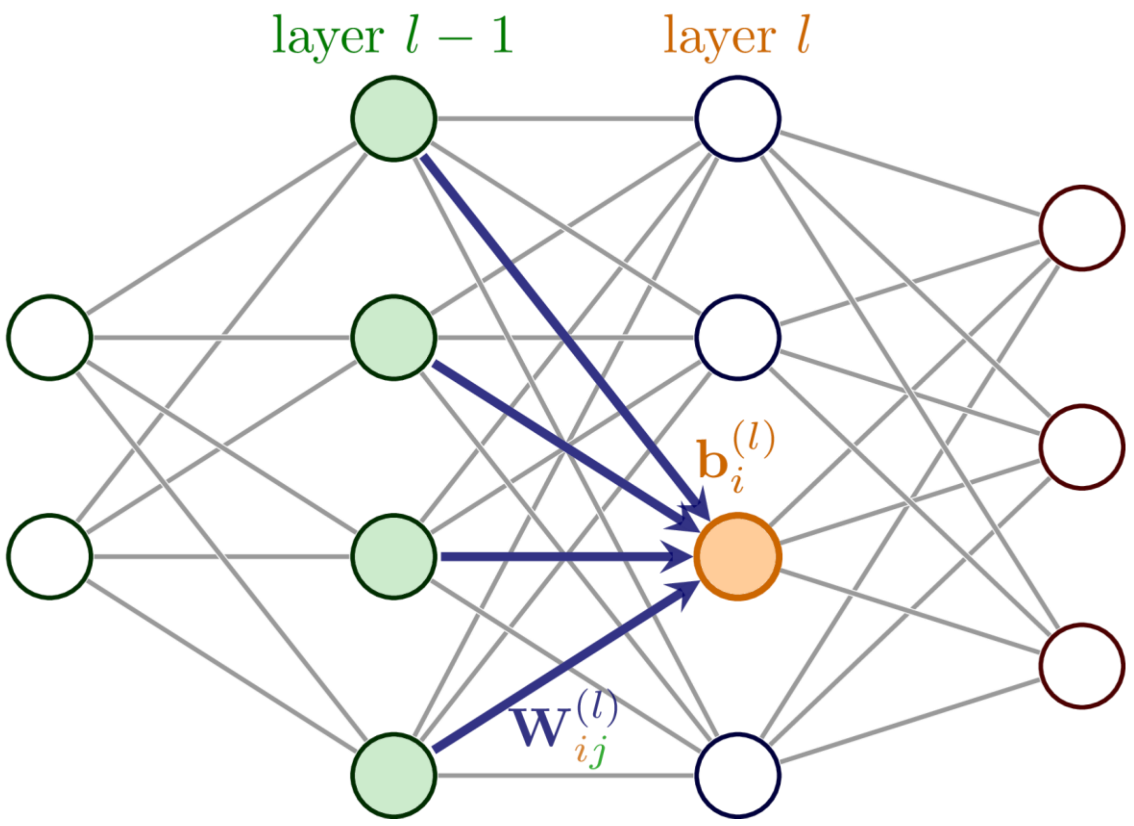
Neural network feedforward activation
(neuron i in layer l)

$$\mathbf{x}_i^{(l)} = \sigma\left(\mathbf{b}_i^{(l)} + \sum_j \mathbf{W}_{ij}^{(l)} \mathbf{x}_j^{(l-1)}\right)$$

Neural network as **neural graph**:

Node i feature: $\boldsymbol{V}_i^{(l)} \leftarrow \mathbf{b}_i^{(l)}$

Edge $j \rightarrow i$ feature: $\boldsymbol{E}_{ij}^{(l)} \leftarrow \mathbf{W}_{ij}^{(l)}$



Node & Edge features

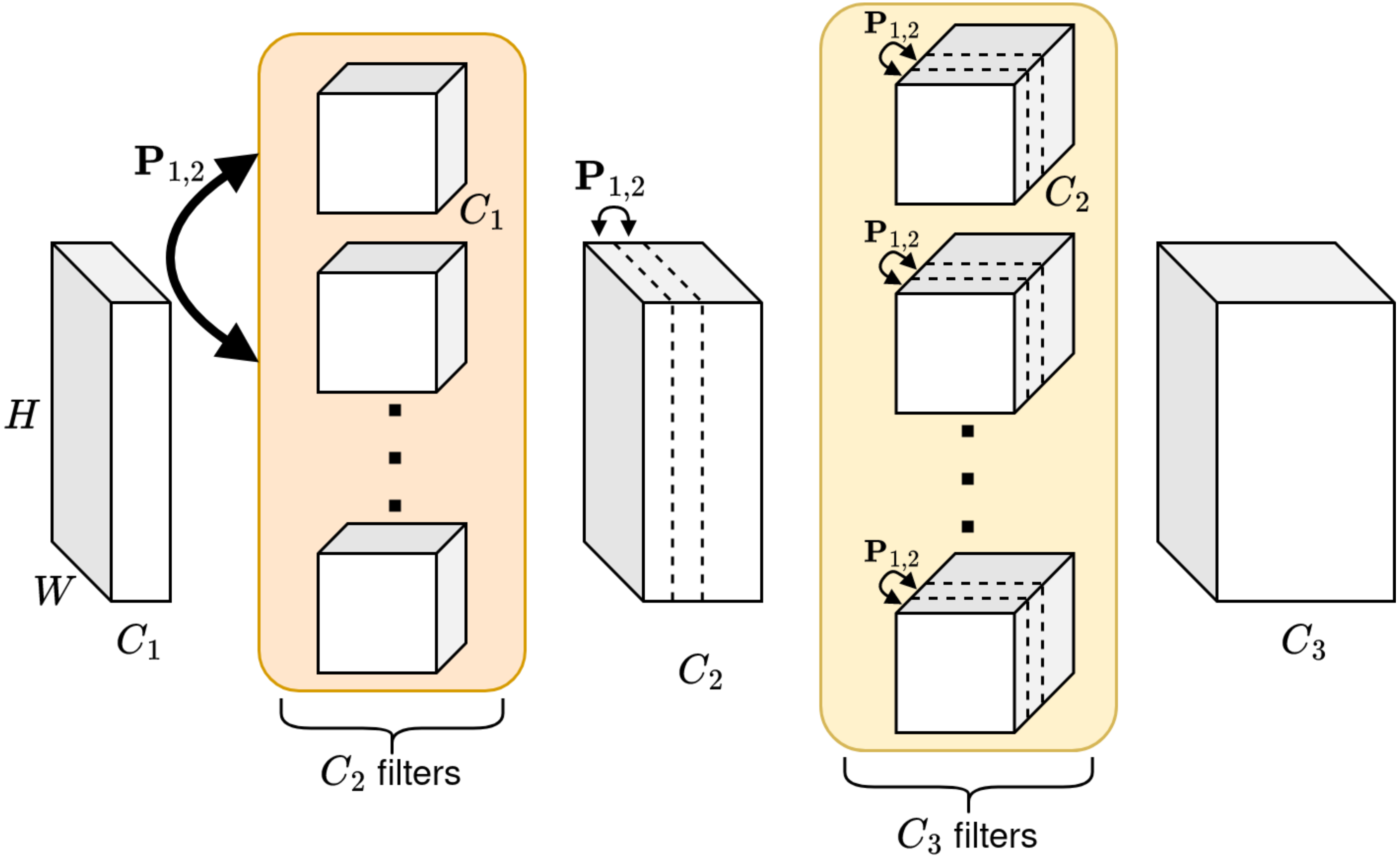
$$
\boldsymbol{E} = \begin{array}{c}
\begin{array}{ccccc} d_0 & d_1 & d_2 & \cdots & d_L \end{array} \\
\begin{pmatrix}
 & \mathbf{W}^{(1)\top} & \ddots & & \\
 & & \mathbf{W}^{(2)\top} & \mathbf{0} & \ddots \\
 & \ddots & & \ddots & \\
 & \mathbf{0} & & & \mathbf{W}^{(L)\top} \\
 & & \ddots & &
\end{pmatrix}
\end{array}
\begin{array}{c} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_L \end{array}
\quad , \quad
\boldsymbol{V} = \begin{pmatrix} \mathbf{0}_{d_0} \\ \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \\ \vdots \\ \mathbf{b}^{(L)} \end{pmatrix}
$$

Our approach – Neural Graphs

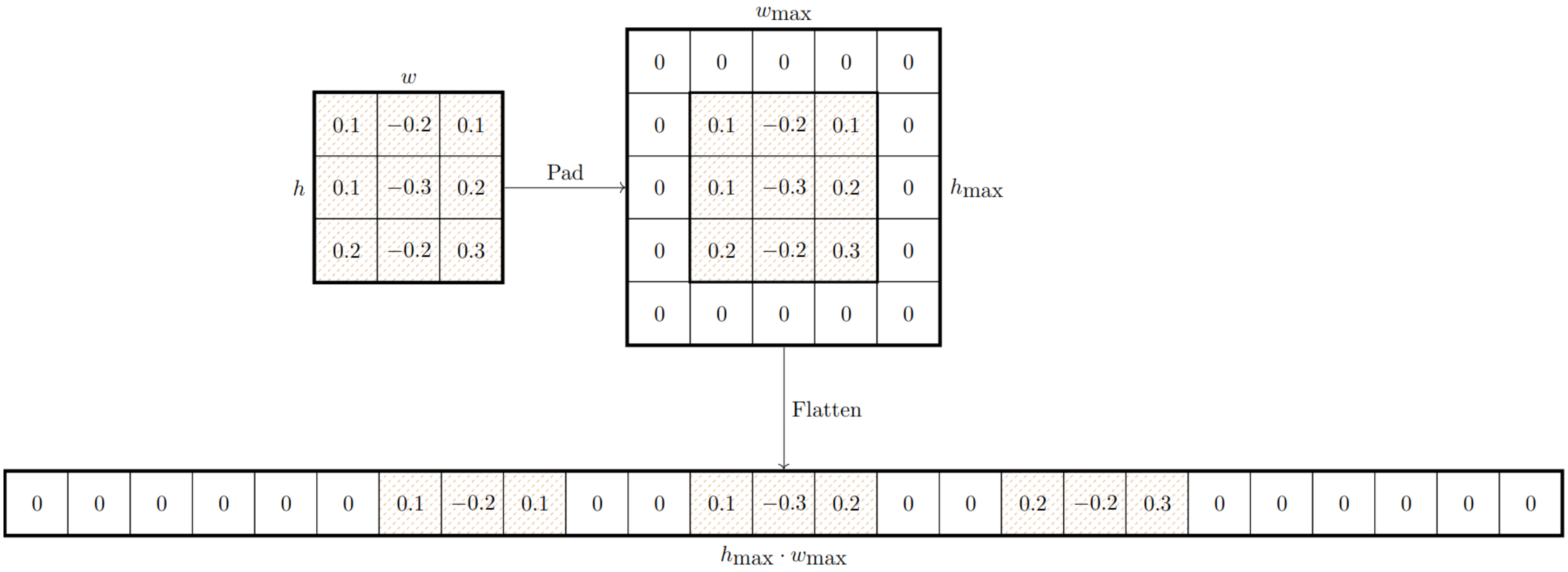
We can process heterogeneous architectures:

- ✓ Architectures with varying computational graphs
- ✓ Different numbers of layers
- ✓ Different number of hidden dimensions
- ✓ Different non-linearities
- ✓ Different network connectivities, such as residual connections

CNN permutation symmetries



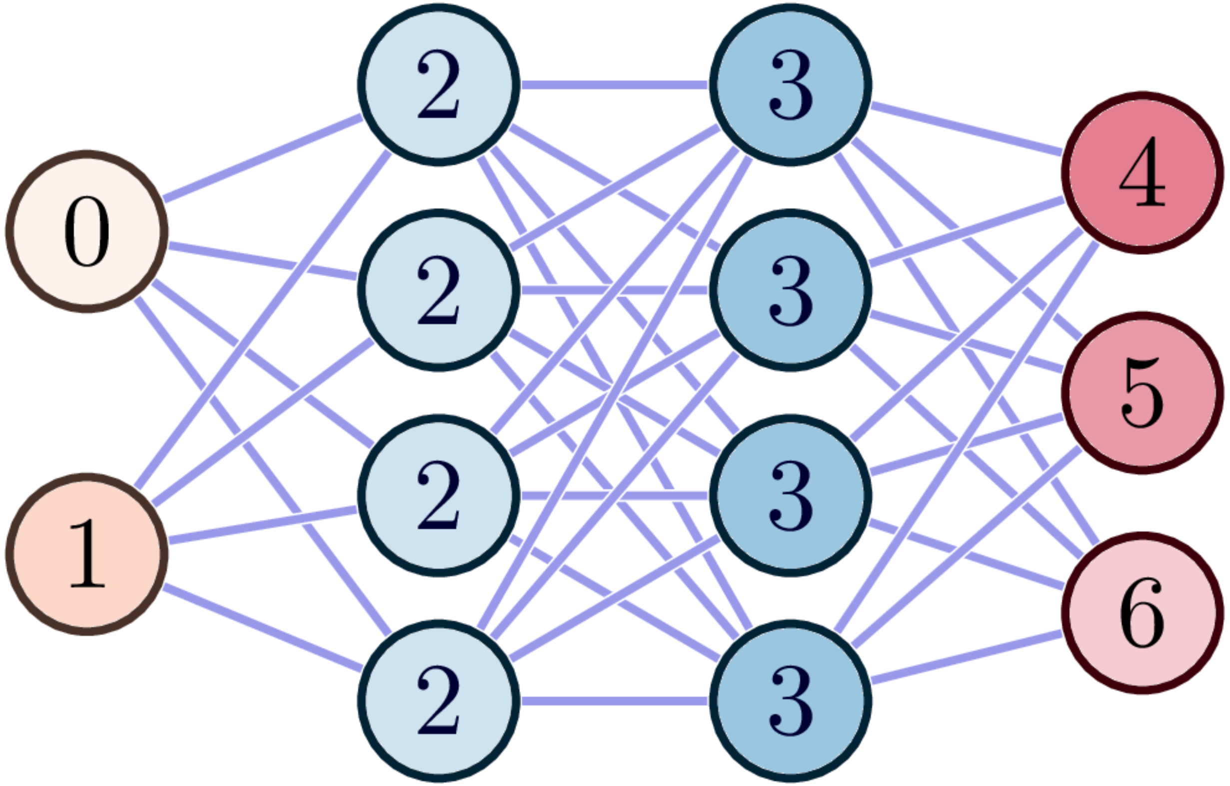
Convolutional kernels as edge features



More neural network modules

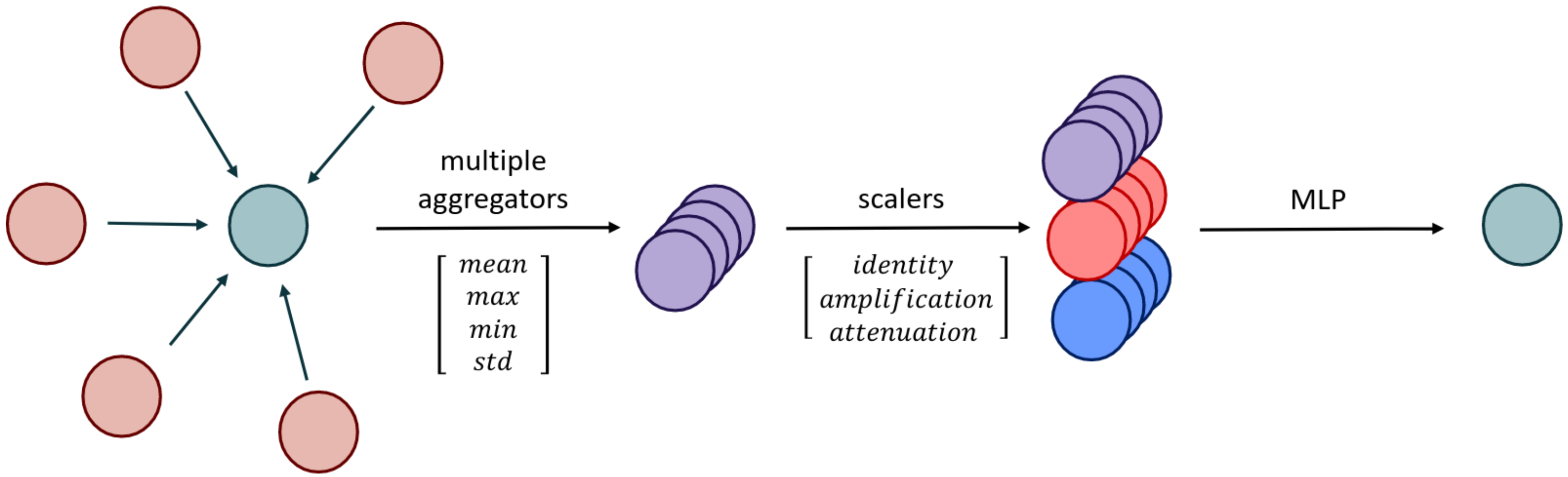
- Residual connections
- Activation functions
- Normalization layers
- Self-attention

Positional embeddings



- One positional embedding per input
- Shared positional embedding per layer
- One positional embedding per output

Neural Graph Graph Network (NG-GNN)

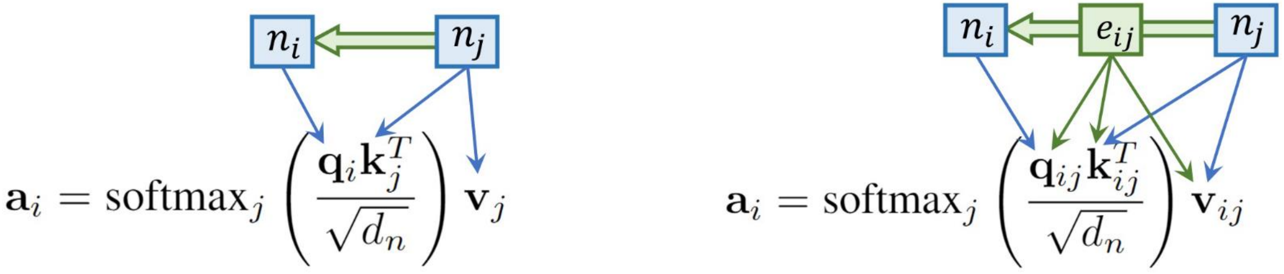


We extend PNA with an MLP that updates the edge features given the incident nodes' features and the previous layer's edge features.

Figure credit: Gabriele Corso et al. "Principal Neighbourhood Aggregation for Graph Nets". In: NeurIPS 2020.

Neural Graph Transformer (NG-T)

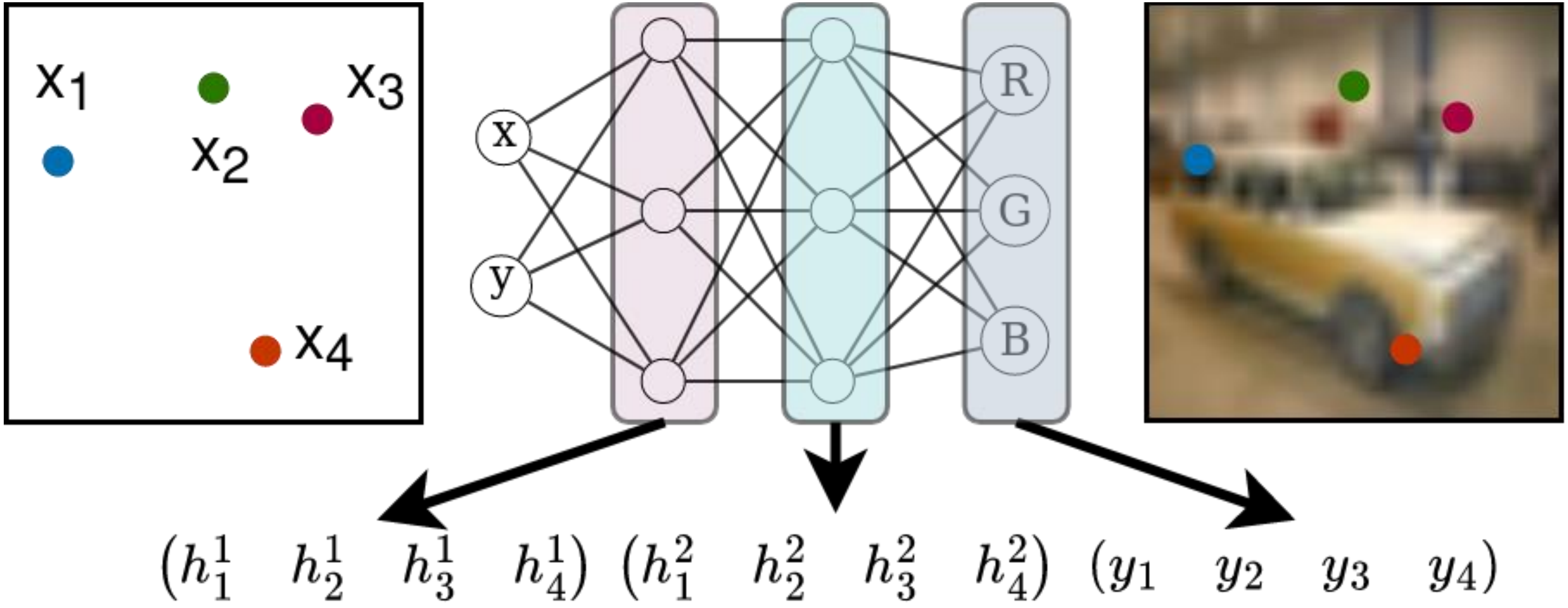
$$\mathbf{q}_{ij} = \left(\mathbf{n}_i \mathbf{W}_n^Q + \mathbf{e}_{ij} \mathbf{W}_e^Q\right) \qquad \mathbf{k}_{ij} = \left(\mathbf{n}_j \mathbf{W}_n^K + \mathbf{e}_{ij} \mathbf{W}_e^K\right) \qquad \mathbf{v}_{ij} = \left(\mathbf{n}_j \mathbf{W}_n^V + \mathbf{e}_{ij} \mathbf{W}_e^V\right)$$



We extend Relational Transformer with multiplicative interactions between node and edge features to algorithmically align it with the forward-pass of a neural network.

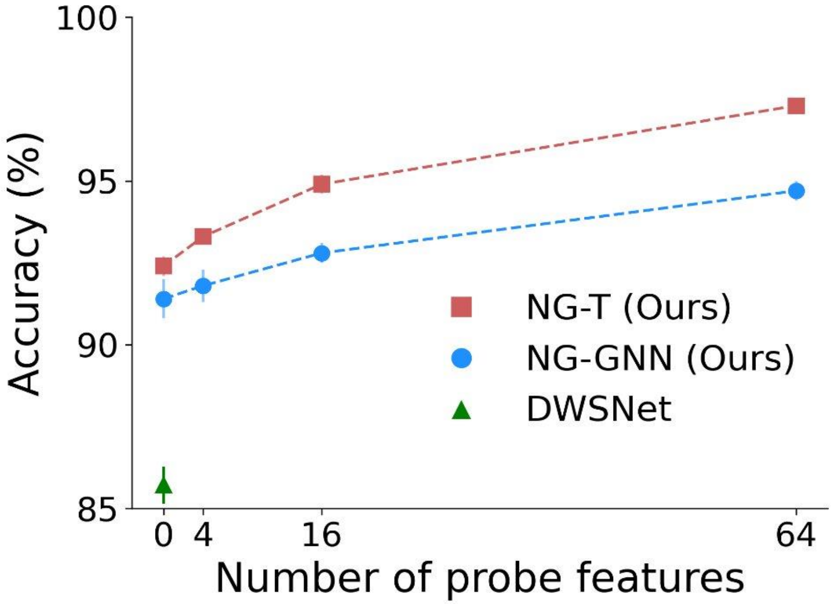
Figure credit: Cameron Diao and Ricky Loynd. “Relational Attention: Generalizing Transformers for Graph-Structured Tasks”. In: ICLR 2023.

Probe features

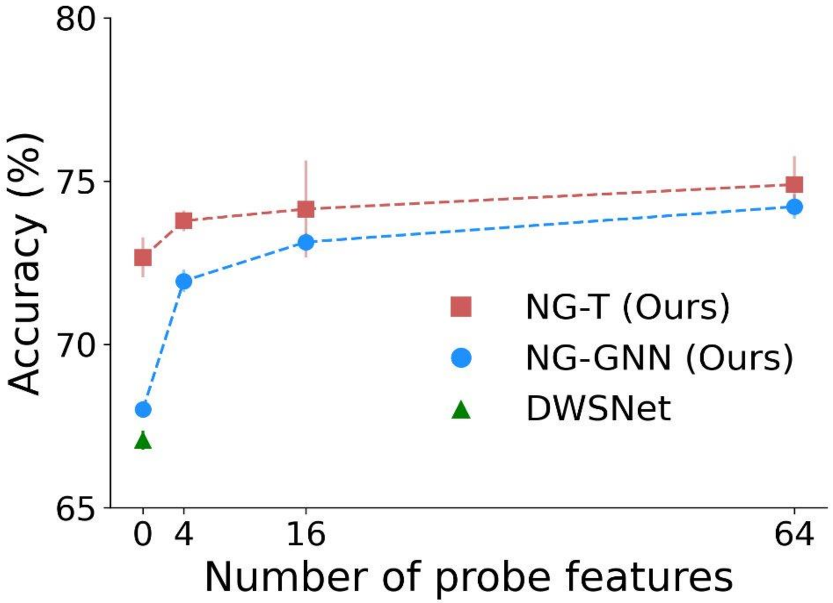


Experiments – INR Classification

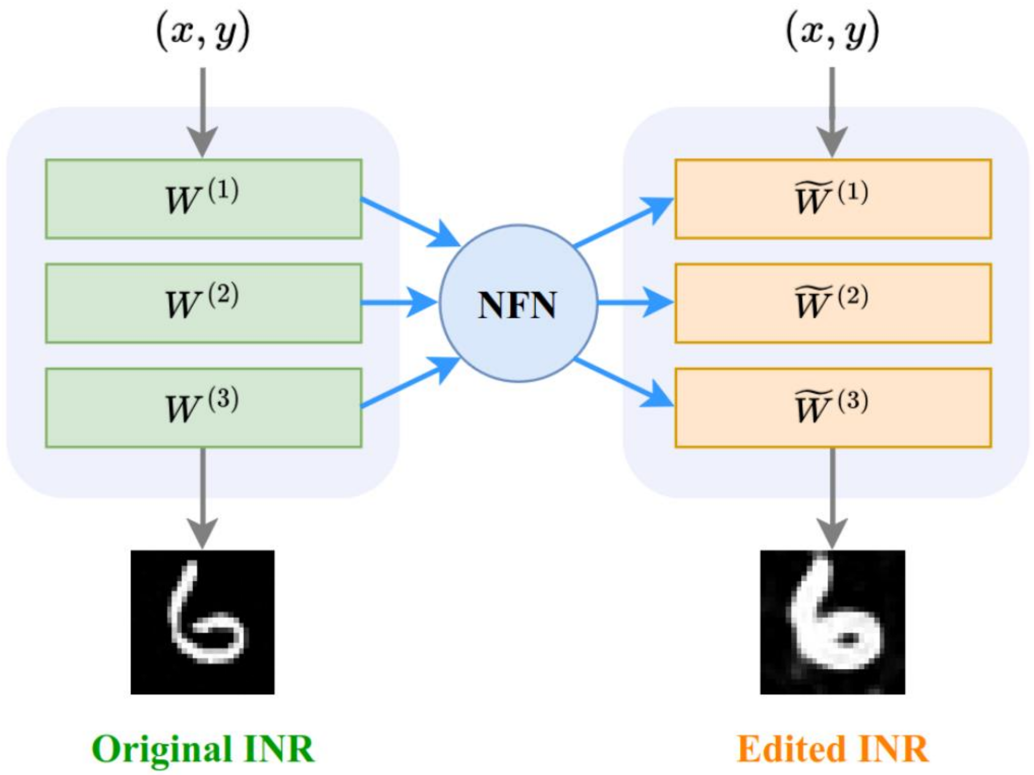
MNIST



Fashion MNIST



Experiments – INR Style Editing



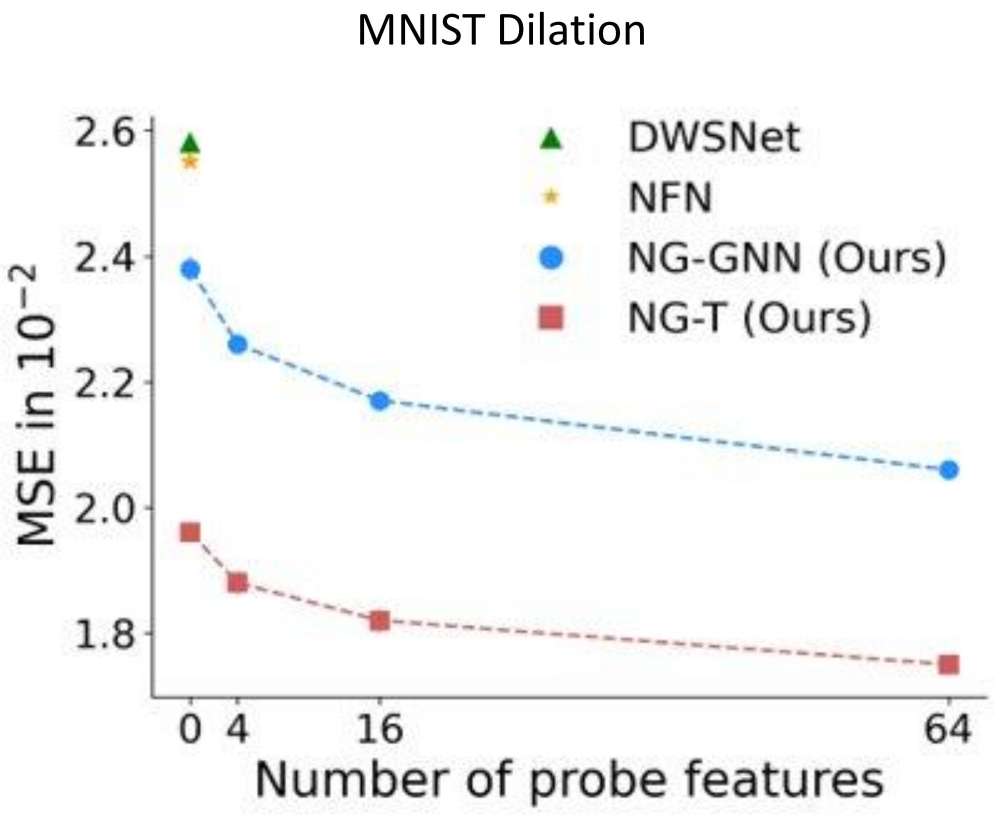


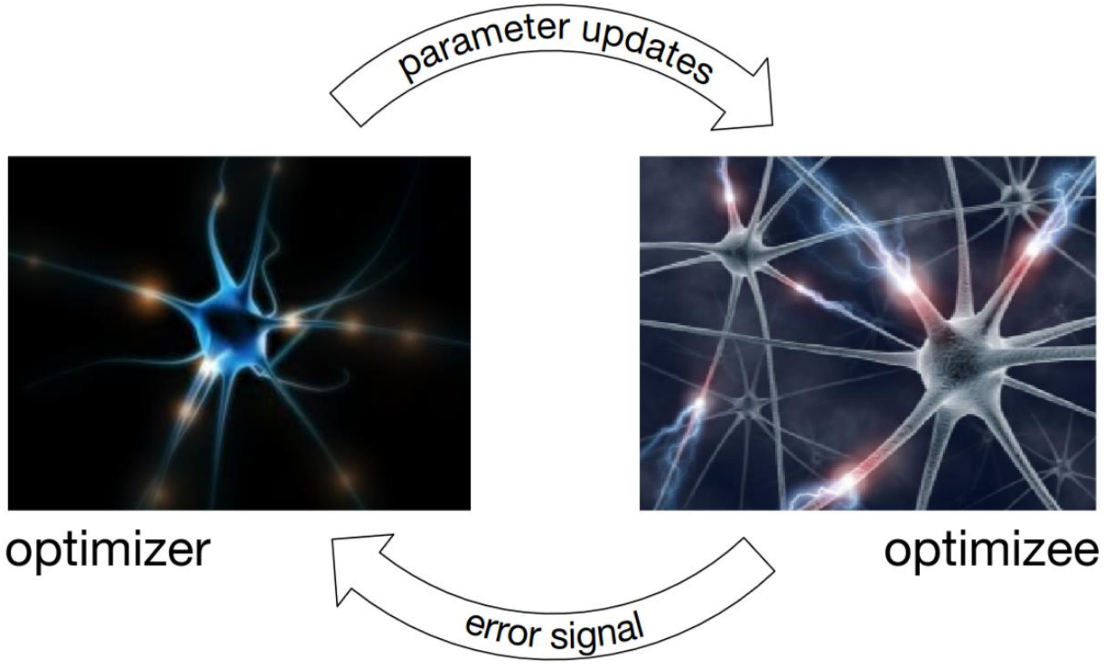
Figure credit (left): Allan Zhou et al. "Permutation Equivariant Neural Functionals". In: NeurIPS 2023.

Experiments – Predict CNN Generalization

- Predict the generalization performance of CNN classifiers based on their parameters
- We introduce *CNN Wild Park,* a dataset of heterogeneous CNNs that vary in the number of layers, kernel sizes, activation functions, and residual connections

Method	CIFAR10-GS	CIFAR10 Wild Park
NFN_{HNP} (Zhou et al., 2023a)	$0.934_{\pm 0.001}$	—
StatNN (Unterthiner et al., 2020)	$0.915_{\pm 0.002}$	$0.719_{\pm 0.010}$
NG-GNN (Ours)	$0.930_{\pm 0.001}$	$0.804_{\pm 0.009}$
NG-T (Ours)	$\mathbf{0.935}_{\pm 0.000}$	$\mathbf{0.817}_{\pm 0.007}$

Exciting application – Learning to optimize



Train a neural network (optimizer) that can optimize the weights of other neural networks (optimizee)

Figure credit: Marcin Andrychowicz et al. “Learning to learn by gradient descent by gradient descent”. In: NeurIPS 2016.

Experiments – Learning to Optimize

- Leverage neural network graph structure
- Train optimizer on Fashion MNIST, evaluate on Fashion MNIST & CIFAR10

Optimizer	FashionMNIST (validation task)	CIFAR-10 (test task)
Adam (Kingma & Ba, 2014)	$80.97_{\pm 0.66}$	$54.76_{\pm 2.82}$
FF (Metz et al., 2019)	$85.08_{\pm 0.14}$	$57.55_{\pm 1.06}$
LSTM (Metz et al., 2020)	$85.69_{\pm 0.23}$	$59.10_{\pm 0.66}$
NFN (Zhou et al., 2023a)	$83.78_{\pm 0.58}$	$57.95_{\pm 0.64}$
NG-GNN (Ours)	$85.91_{\pm 0.37}$	$\mathbf{64.37}_{\pm 0.34}$
NG-T (Ours)	$\mathbf{86.52}_{\pm 0.19}$	$60.79_{\pm 0.51}$

Conclusion

- Processing neural networks with neural networks is an exciting new research avenue
- Novel representation of neural networks as neural graphs
- Introduce Graph networks for processing neural networks
- Applications in INRs, CNN generalization, learning to optimize
- Neural graphs constitute a *new benchmark for graph networks*

Resources

- Source code: https://github.com/mkofinas/neural-graphs
- Arxiv: https://arxiv.org/abs/2403.12143

- Visit our poster today
    - Poster **#77**, poster session 4, Halle B

- Contact us:
    - mkofinas@gmail.com
    - david0w0zhang@gmail.com