HELLENIC REPUBLIC
National and Kapodistrian University of Athens
EST. 1837

ARCHIMEDES

Scale Equivariant Graph MetaNetworks

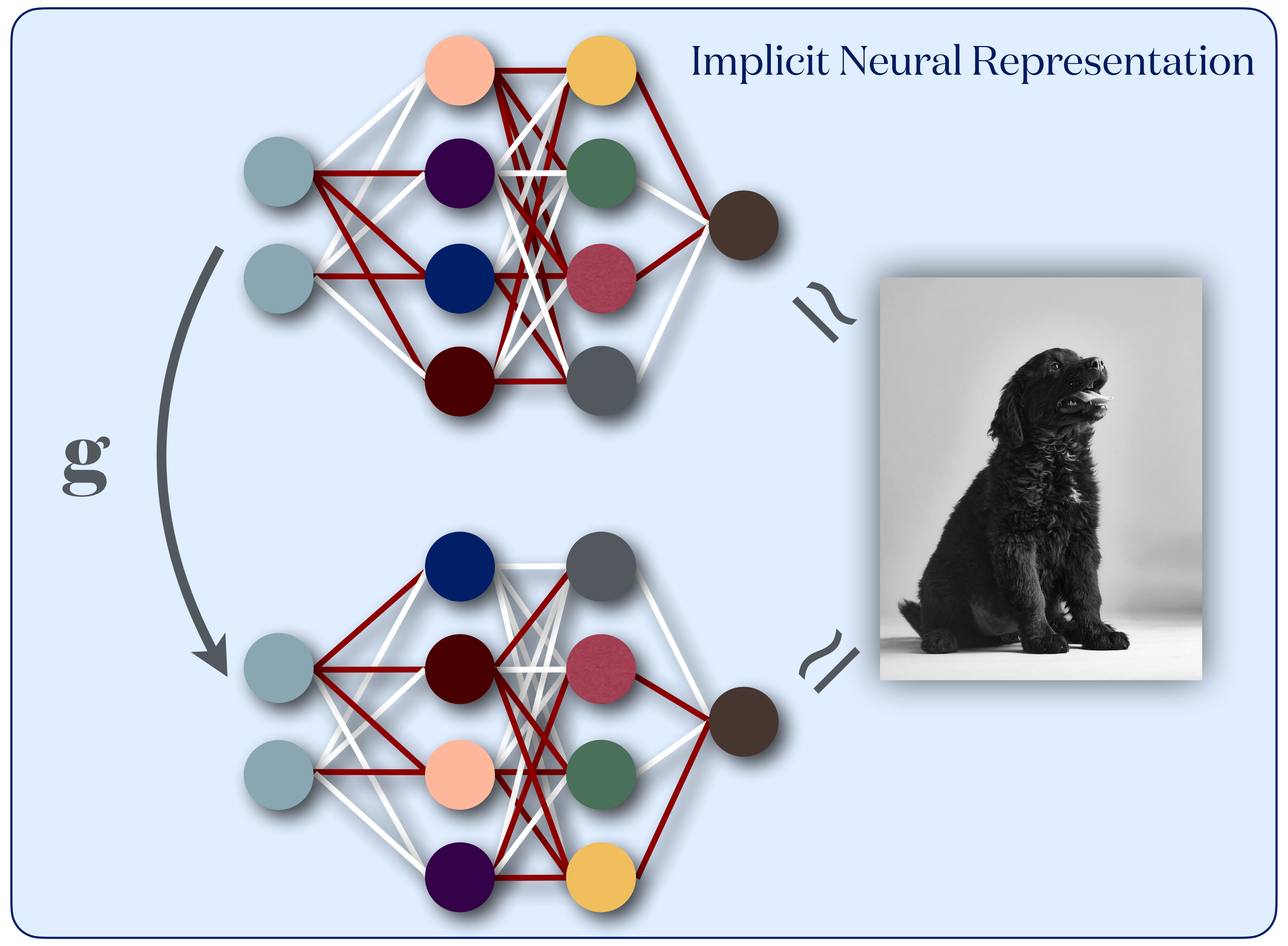
Ioannis Kalogeropoulos[1,2]*, *Giorgos Bouritsas*[1,2]*, *Yannis Panagakis*[1,2]

[1]National and Kapodistrian University of Athens
[2]Archimedes AI/Athena RC, Greece

NeurIPS

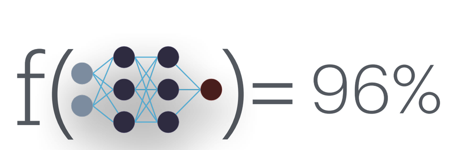
An emerging machine learning paradigm

Learning **higher-order functions**, i.e. functions whose inputs are functions themselves, particularly when these inputs are Neural Networks (NNs).



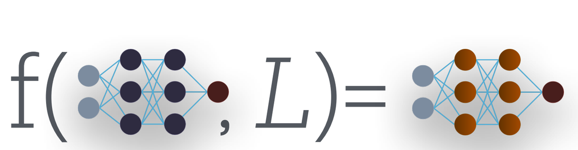
Motivation

1. Vast ecosystem of **millions of publicly available trained models**.
2. **Discrete data as trainable continuous functions**:
   - Trained **NN parameters** to represent **datapoint signals**, such as images or 3D shapes (INR/NeRF).
3. **Neural Network processing**:
   - Analysis/Interpretation:



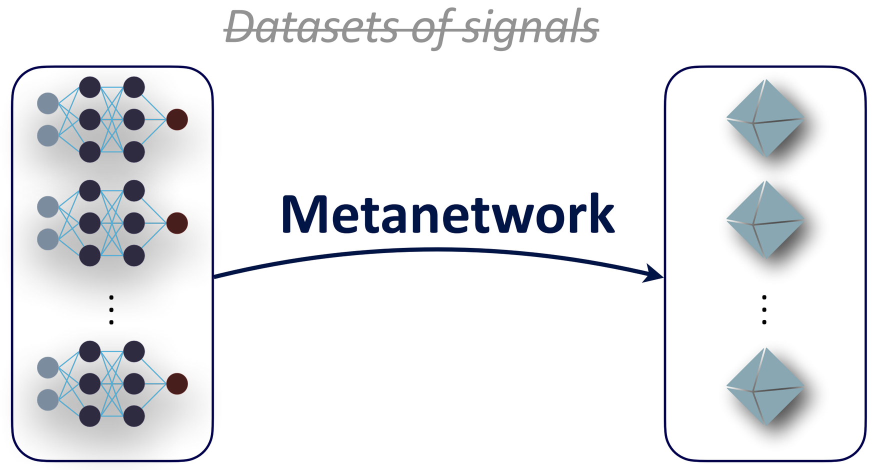
   - Editing:

   - Synthesis:



Datasets of NNs

~~Datasets of signals~~



TLDR: We aim to learn functions of neural networks. We study their symmetries and create Scale Equivariant Graph Metanetworks.

Neural Network symmetries

- Network architecture → *Permutation* symmetries: Hidden neurons do not possess any inherent ordering
- Activation functions → *Scaling* symmetries: Multiplying hidden neurons' biases and all incoming weights with a constant a results in scaling its output with a corresponding constant b.

For several activation functions we have:

$$\left(\mathbf{P}_\ell \mathbf{Q}_\ell \mathbf{W}_\ell \mathbf{Q}_{\ell-1}^{-1} \mathbf{P}_{\ell-1}^{-1}, \mathbf{P}_\ell \mathbf{Q}_\ell \mathbf{b}_\ell\right)_{\ell=1}^{L} = \boldsymbol{\theta}' \simeq \boldsymbol{\theta} = \left(\mathbf{W}_\ell, \mathbf{b}_\ell\right)_{\ell=1}^{L}$$

ReLU	sine (INR), tanh
Generalised permutation matrices with positive entries of the form: $\mathbf{PQ}, \mathbf{Q} = \mathrm{diag}(q_1, \dots, q_d), q_i > 0$	Signed permutation matrices of the form: $\mathbf{PQ}, \mathbf{Q} = \mathrm{diag}(q_1, \dots, q_d), q_i = \pm 1$

For more details on scaling symmetries, refer to the paper.

ScaleGMN

Consisted of 3 building blocks.

How to be Scale Invariant?

$$\mathsf{ScaleInv}(\mathbf{X}) = \rho\left(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n\right)$$

(Scale Invariant Net)

$$\tilde{\mathbf{x}}_i = \mathsf{canon}(\mathbf{x}_i) \ \text{ or } \ \mathsf{symm}(\mathbf{x}_i)$$

How to be ReScale Equivariant?

Input of MSG **scaled by different multipliers**:

$$g \curvearrowright \mathrm{MSG}(h_i, h_j, e_{ji}))$$

$$\mathrm{MSG}(\underbrace{q_i\, h_i}_{\text{central vertex}}, \underbrace{q_j\, h_j}_{\text{neighbor}}, \underbrace{\tfrac{q_i}{q_j}\, e_{ji}}_{\text{edge}})$$

The output should be scaled by q_i .

$$\mathrm{ReScaleEq}(\mathbf{x}_1, \mathbf{x}_2) = \boldsymbol{\Gamma}_1 \mathbf{x}_1 \odot \boldsymbol{\Gamma}_2 \mathbf{x}_2$$

(ReScale Equivariant Net)

$$\mathrm{MSG}(h_i, \mathrm{ReScaleEq}(h_j, e_{ji}))$$

How to be Scale Equivariant?

$$\mathsf{ScaleEq} = \mathsf{f}^K \circ \dots \circ \mathsf{f}^1$$

$$\mathsf{f}^k(\mathbf{X}) = \left(\boldsymbol{\Gamma}_1^k \mathbf{x}_1, \dots, \boldsymbol{\Gamma}_n^k \mathbf{x}_n\right) \odot \mathsf{ScaleInv}^k(\mathbf{X})$$

(Scale Equivariant Net)

Using the above, we create ScaleGMN:

1. Map the input NN to a **graph**, based on its architecture.
2. Define **scale equivariant** MSG and UPD functions:

$$\mathrm{MSG}\left(\mathbf{x}, \mathbf{y}, \mathbf{e}\right) = \mathsf{ScaleEq}\left(\left[\mathbf{x}, \mathsf{ReScaleEq}\left(\mathbf{y}, \mathbf{e}\right)\right]\right)$$

$$\mathrm{UPD}\left(\mathbf{x}, \mathbf{m}\right) = \mathsf{ScaleEq}\left(\left[\mathbf{x}, \mathbf{m}\right]\right)$$

3. Define a scale and permutation-invariant READ function:

$$\mathrm{READ}\left(\mathbf{X}\right) := \mathsf{DeepSets}\left(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n\right)$$

The above method refers to the *forward* ScaleGMN variant. Refer to the paper for the bidirectional one.

Theoretical Outcomes

Proposition

ScaleGMN is *permutation* & *scale equivariant*. Additionally, ScaleGMN is *permutation* & *scale invariant* when using a readout with the same symmetries.

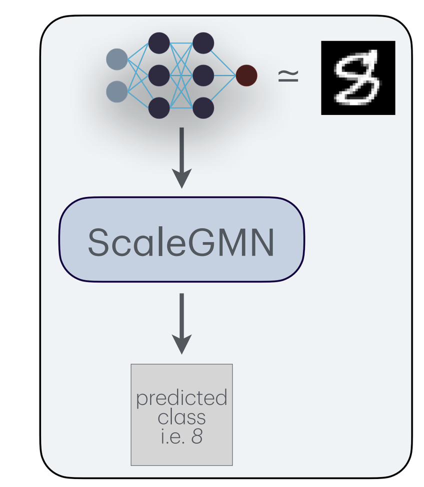
Theorem

Bidirectional ScaleGMN with sufficiently expressive MSG and vertex *UPD* functions can *simulate the forward and backward pass* of any input FFNN, when ScaleGMN's depth are equal or double the length of the input neural network.

Experiments

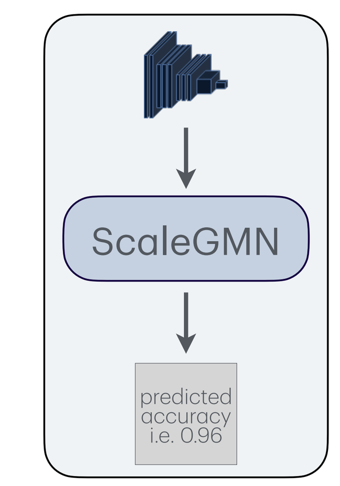
1. INR Classification

Method	MNIST	F-MNIST	CIFAR-10	Augmented CIFAR-10
MLP	17.55 ± 0.01	19.91 ± 0.47	11.38 ± 0.34*	16.90 ± 0.25
Inr2Vec [2]	23.69 ± 0.10	22.33 ± 0.41	-	-
DWS [3]	85.71 ± 0.57	67.06 ± 0.29	-	-
$\mathrm{NFN_{NP}}$ [5]	78.50 ± 0.23*	68.19 ± 0.28*	33.41 ± 0.01*	46.60 ± 0.07
$\mathrm{NFN_{HNP}}$ [5]	79.11 ± 0.84*	68.94 ± 0.64*	28.64 ± 0.07*	44.10 ± 0.47
NG-GNN [1]	91.4 ± 0.60	68.0 ± 0.20	36.04 ± 0.44*	45.70 ± 0.20*
ScaleGMN (Ours)	96.57 ± 0.10	80.46 ± 0.32	36.43 ± 0.41	56.62 ± 0.24
ScaleGMN-B (Ours)	**96.59 ± 0.24**	**80.78 ± 0.16**	**38.82 ± 0.1**	**56.95 ± 0.57**



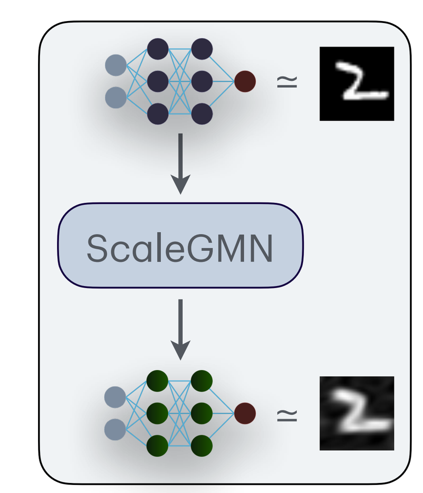
2. Generalization prediction

Method	CIFAR-10-GS ReLU	SVHN-GS ReLU	CIFAR-10-GS Tanh	SVHN-GS Tanh
StatNN [4]	0.9140 ± 0.001	0.8463 ± 0.004	0.9140 ± 0.000	0.8440 ± 0.001
$\mathrm{NFN_{NP}}$ [5]	0.9190 ± 0.010	0.8586 ± 0.003	0.9251 ± 0.001	0.8580 ± 0.004
$\mathrm{NFN_{HNP}}$ [5]	0.9270 ± 0.001	0.8636 ± 0.002	0.9339 ± 0.000	0.8586 ± 0.004
NG-GNN ([1])	0.9010 ± 0.060	0.8549 ± 0.002	0.9340 ± 0.001	0.8620 ± 0.003
ScaleGMN (Ours)	0.9276 ± 0.002	**0.8689 ± 0.003**	0.9418 ± 0.005	**0.8736 ± 0.003**
ScaleGMN-B (Ours)	**0.9282 ± 0.003**	0.8651 ± 0.001	**0.9425 ± 0.004**	0.8655 ± 0.004



3. INR Editing

Method	MSE in 10^{-2}
MLP	5.35 ± 0.00
DWS [3]	2.58 ± 0.00
$\mathrm{NFN_{NP}}$ [5]	2.55 ± 0.00
$\mathrm{NFN_{HNP}}$ [5]	2.65 ± 0.01
NG-GNN-0 [1]	2.38 ± 0.02
NG-GNN-64 [1]	2.06 ± 0.01
ScaleGMN (Ours)	2.56 ± 0.03
ScaleGMN-B (Ours)	**1.89 ± 0.00**



References

[1] Kofinas, M., et al. "Graph neural networks for learning equivariant representations of neural networks". In: ICLR2024
[2] De Luigi, L., et al. "Deep learning on implicit neural representations of shapes". In: ICLR 2023
[3] Navon, A., et al. "Equivariant architectures for learning in deep weight spaces". In: ICML 2023
[4] Unterthiner, T., et al. "Predicting neural network accuracy from weights". 2020
[5] Zhou, A., et al. "Permutation equivariant neural functionals". In: NeurIPS 2024