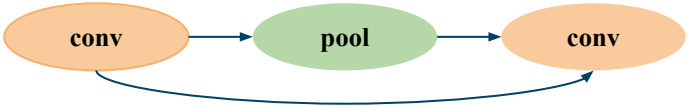
Neural Networks as Graphs

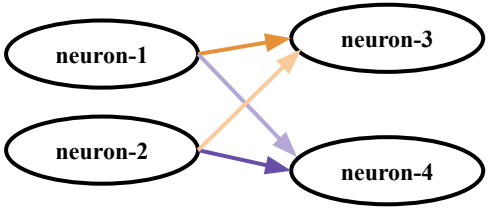
Boris Knyazev

SAMSUNG Université de Montréal

Agenda

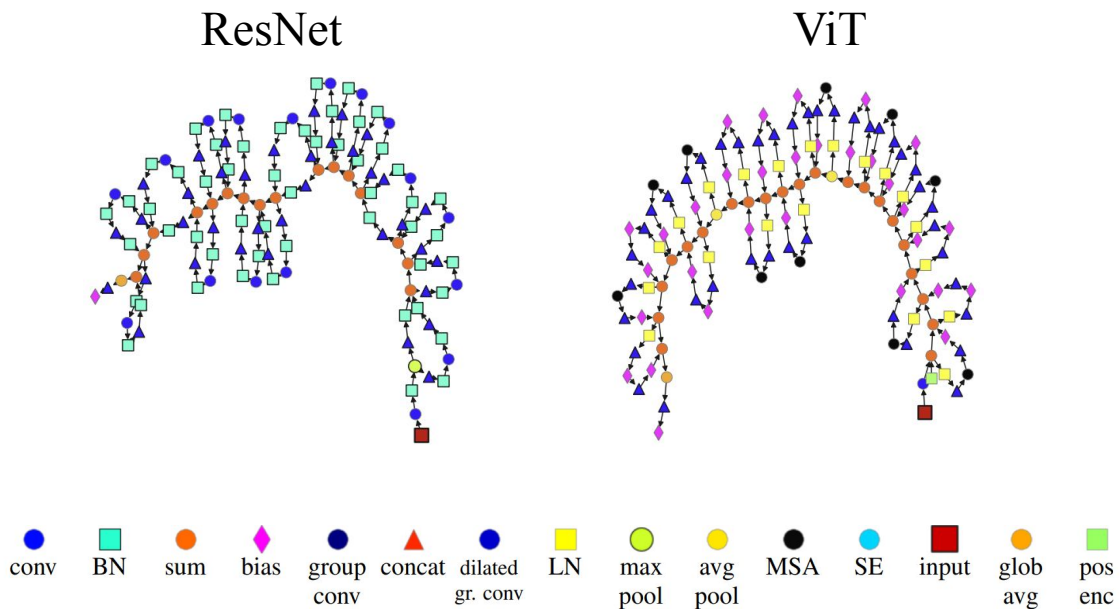
1. Graph of neural architectures
    a. Graph HyperNetworks
2. Graph of parameters
    a. Neural graphs
    b. NiNo
3. Future directions





1. Graph of Architectures

Graph of Architectures



Example:

node 1: [1, 0, 0, 0, 0], input
node 2: [0, 1, 0, 0, 0], conv
node 3: [0, 0, 0, 0, 1], bias
node 4: [0, 0, 0, 1, 0], pooling
…
node 160: [0, 0, 1, 0, 0], fc
node 161: [0, 0, 0, 0, 1], bias

Each graph is represented by:
X - node features
A - adjacency matrix (forward pass flow)

- Graph HyperNetworks for Neural Architecture Search (Zhang et al., ICLR 2019)
- Parameter Prediction for Unseen Deep Architectures (Knyazev et al., NeurIPS 2021)

HyperNetworks

Standard objective:

$w^* = \text{argmin}\, f(x, w)$

Hypernetwork objective (Ha et al., 2016):

$\theta^* = \text{argmin}\, f(x, w{=}H[\theta; \cdot])$

Advantages:

- Reduce the number of params to train and store
- Neural Architecture Search
- Few-shot adaptation, data conditioning
- **Better initialization of larger architectures**

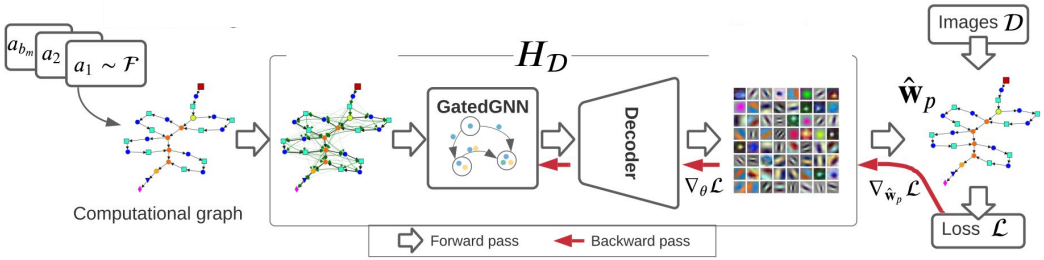
Graph HyperNetworks

Standard objective:

$w^* = \text{argmin}\, f(x, w)$

GHN objective:

$\theta^* = \text{argmin}\, f(x, w{=}H_D[\theta; a])$



- We generated a dataset of 1M graphs (no need to train the parameters!)
- We use the dataset for training a **parameter prediction** model (GHN)

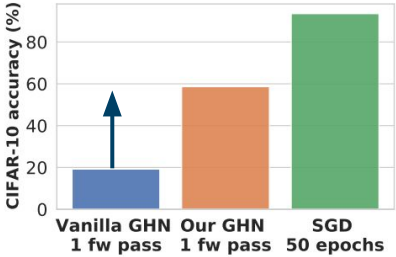
- Parameter Prediction for Unseen Deep Architectures (Knyazev et al., NeurIPS 2021)
- Can We Scale Transformers to Predict Parameters of Diverse ImageNet Models? (Knyazev et al., ICML 2023)

Key Example

```
# Pseudo-code

ghn = GHN('imagenet')  # 1. pretrained GHN
model = resnet50()     # 2. can be almost any model
model = ghn(model)     # 3. returns model with predicted parameters
y = model(test_images) # 4. model can achieve high acc right away
```

Example of evaluating on an unseen architecture $a \notin \mathcal{F}$ (ResNet-50)

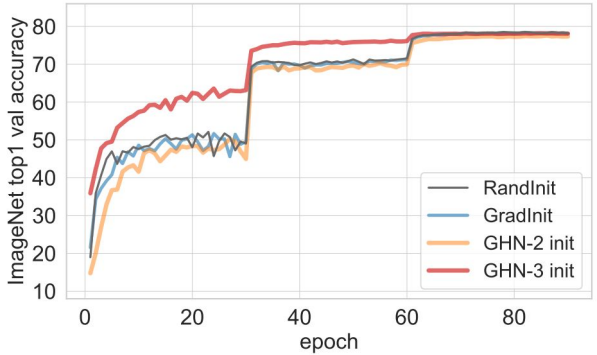


- Predicted parameters ≡ strong initialization
- BUT works only for the pretraining dataset

Graph of Architectures – Summary

1. Advantages:
   a. Represents many neural architectures
   b. Compact representation
   c. Synergizes with hypernetworks:
      i. Predicting parameters (for initialization)
      ii. Neural Architecture Search
      iii. Few-shot adaptation

2. Limitations (or opportunities):
   a. Does not represent the parameters
   b. Architecture/layer type constraints
   c. Dataset specific

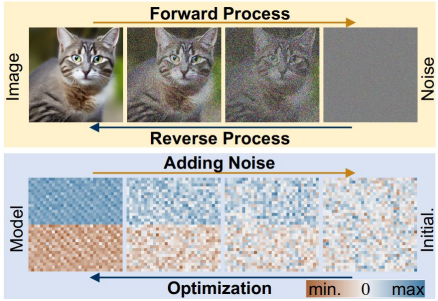


2. Graph of Parameters

Neural Network Parameters vs Images

Neural networks:

1. Different and very large sizes
2. Different symmetries
3. Lack of intuitive structure
4. Different range of values
5. Applications are early but promising



Images:

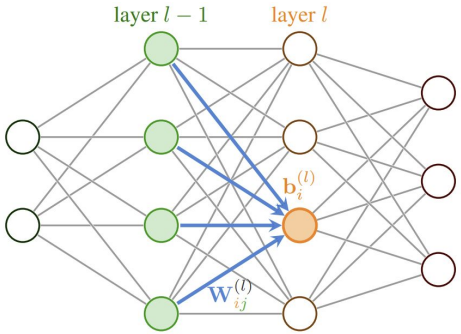
1. Resize/crop images
2. Augmentations
3. Spatial locality, compositionality
4. 0-255
5. Clear applications

- Neural Network Diffusion (Wang et al., 2024)
- Learning to Learn with Generative Models of Neural Network Checkpoints (Peebles et al., 2022)
- Hyper-Representations, SANE (Konstantin Schürholt, 2021-2024)

Graph of Parameters

$$\mathbf{y} = \text{softmax}(\mathbf{W}^{(3)}\sigma(\mathbf{W}^{(2)}(\sigma(\mathbf{W}^{(1)}\mathbf{x}))))$$

$$\theta = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}\}$$



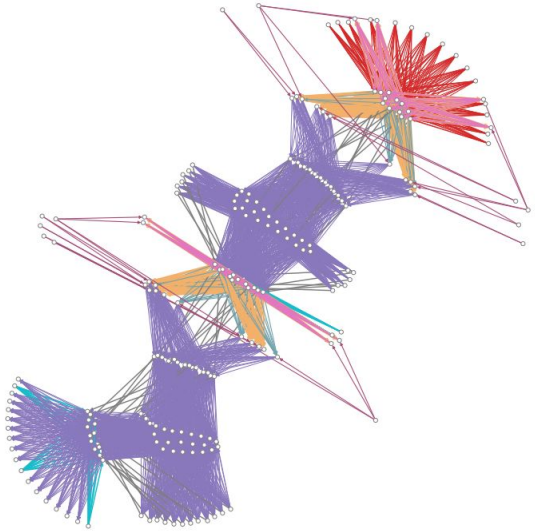
Neural Graph

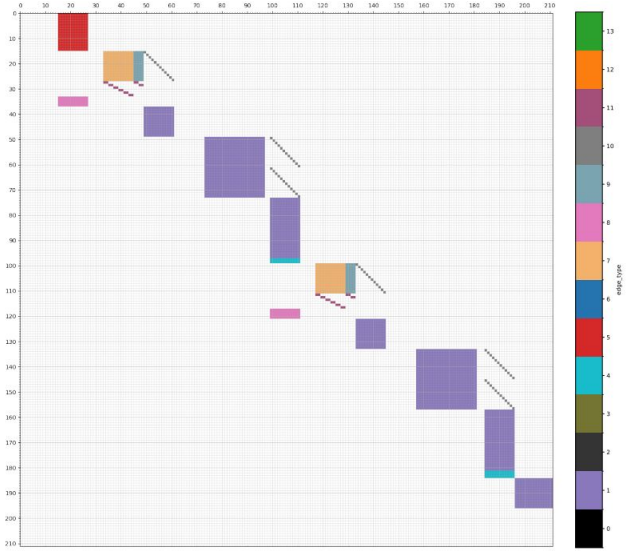
$$\mathcal{G}(\theta) = (\mathbf{V}, \mathbf{E})$$

$$\mathbf{E} = \begin{pmatrix} \mathbf{W}^{(1)} & & & \\ & \mathbf{W}^{(2)} & & \mathbf{0} \\ & & \ddots & \\ & \mathbf{0} & & \mathbf{W}^{(L)} \end{pmatrix}$$

Graph Neural Networks for Learning Equivariant Representations of Neural Networks (ICLR 2024)
M. Kofinas, B. Knyazev, Y. Zhang, Y. Chen, C.J. Burghouts, E. Gavves, C.G.M. Snoek, D.W. Zhang

Graph of Parameters – Llama3*



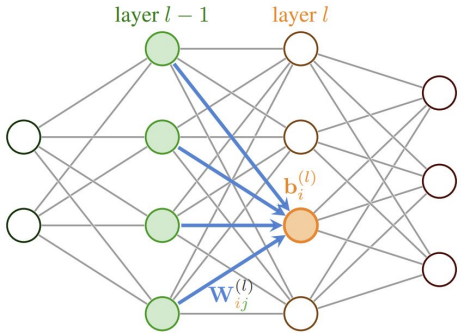


Accelerating Training with Neuron Interaction and Nowcasting Networks
(Knyazev et al., ICLR 2025)

Graph of Parameters – Properties

$$f(\mathbf{x}, \theta) = f(\mathbf{x}, \pi^{\text{good}}(\theta)) \qquad \mathcal{G}(\theta) \cong \mathcal{G}(\pi^{\text{good}}(\theta))$$

$$f(\mathbf{x}, \theta) \neq f(\mathbf{x}, \pi^{\text{bad}}(\theta)) \qquad \mathcal{G}(\theta) \ncong \mathcal{G}(\pi^{\text{bad}}(\theta))$$



Neural graphs model **permutation symmetry**:

- Permutation (π) of nodes = permutation (π) neurons
- Graphs are isomorphic when the **neural network's function** does not change, and vice versa

Graph of Parameters – Why to use it?

$$\mathbf{V}^m, \boldsymbol{\mathcal{E}}^m = \mathrm{GNN}_{\phi_m}(\mathbf{V}^{m-1}, \boldsymbol{\mathcal{E}}^{m-1})$$

Representation learned by GNNs:

1. Expressive
2. Equivariant/invariant to neuron permutations
3. Equivariant/invariant *to input size*

Predicting accuracy of weights (correlation)

Method	CIFAR10-GS	CIFAR10 Wild Park
$\mathrm{NFN}_{\mathrm{HNP}}$ (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)	$0.931_{\pm 0.002}$	$0.804_{\pm 0.009}$
NG-T (Ours)	$\mathbf{0.935}_{\pm \mathbf{0.000}}$	$\mathbf{0.817}_{\pm \mathbf{0.007}}$

Graph Neural Networks for Learning Equivariant Representations of Neural Networks (ICLR 2024)
M. Kofinas, B. Knyazev, Y. Zhang, Y. Chen, C.J. Burghouts, E. Gavves, C.G.M. Snoek, D.W. Zhang

Graph of Parameters – GNNs

Many works use such or similar graphs

- **Accelerating Training with Neuron Interaction and Nowcasting Networks** (ICLR 2025)
  B. Knyazev, A. Moudgil, G. Lajoie, E. Belilovsky, S. Lacoste-Julien
- **Graph Neural Networks for Learning Equivariant Representations of Neural Networks** (ICLR 2024)
  M. Kofinas, B. Knyazev, Y. Zhang, Y. Chen, C.J. Burghouts, E. Gavves, C.G.M. Snoek, D.W. Zhang
- **Universal Neural Functionals** (arXiv 2024)
  A. Zhou, C. Finn, J. Harrison
- **Graph Metanetworks for Processing Diverse Neural Architectures** (ICLR 2024)
  D. Lim, H. Maron, M.T. Law, J. Lorraine, J. Lucas.
- **Scale Equivariant Graph Metanetworks** (NeurIPS 2024)
  I. Kalogeropoulos, G. Bouritsas, Y. Panagakis
- **Improved Generalization of Weight Space Networks via Augmentations** (ICML 2024)
  A. Shamsian, A. Navon, D.W. Zhang, Y. Zhang, E. Fetaya, G. Chechik, H. Maron
- **Equivariant Architectures for Learning in Deep Weight Spaces** (ICML 2023)
  A. Navon, A. Shamsian, I. Achituve, E. Fetaya, G. Chechik, H. Maron
- **Deep Neural Network Fusion via Graph Matching with Applications to Model Ensemble and Federated Learning** (ICML 2020)
  C. Liu, C. Lou, R. Wang, A.Y. Xi, L. Shen, J. Yan
- **Graph Structure of Neural Networks** (ICML 2020)
  J. You, J. Leskovec, K. He, S. Xie

Neural graphs + GNNs:

1. Different and very large sizes ✔
2. Different symmetries ✔
3. Lack of intuitive structure
4. Different range of values
5. **Applications – optimization** ✔

NiNo

Neuron Interaction and Nowcasting Networks

Accelerating Training with Neuron Interaction and Nowcasting Networks

```
# Adam
for t in range(10^6):
    θ_{t+1} = adam.step(θ_t, ∇θ_t, ...)
```

- Slow convergence

```
# Learnable optimizers (L2O)
for t in range(10^6):
    θ_{t+1} = f.predict(θ_t, ∇θ_t, ...)
```

- Faster convergence
- Overhead
- Generalization issues
- Tricky to integrate

- **µLO: Compute-Efficient Meta-Generalization of Learned Optimizers** (under review)
  B. Thérien, C.É. Joseph, B. Knyazev, E. Oyallon, I. Rish, E. Belilovsky
- **Learning Versatile Optimizers on a Compute Diet** (under review)
  A. Moudgil, B. Knyazev, G. Lajoie, E. Belilovsky
- **VeLO: Training Versatile Learned Optimizers by Scaling Up** (arXiv 2022)
  L. Metz, J. Harrison, C.D. Freeman, A. Merchant, L. Beyer, J. Bradbury, N. Agrawal et al.

Accelerating Training with Neuron Interaction and Nowcasting Networks

```
# Adam
for t in range(10^6):
    θ_{t+1} = adam.step(θ_t, ∇θ_t, ...)
```

- Slow convergence

```
# Learnable optimizers (L2O)
for t in range(10^6):
    θ_{t+1} = f.predict(θ_t, ∇θ_t, ...)
```

- Faster convergence
- Overhead
- Generalization issues
- Tricky to integrate

```
# Periodically predict future parameters
for t in range(10^6):
    If t % 1000 == 0:
        θ_{t+K} = f.predict(θ_t, θ_{t-1}, ...)
    else:
        θ_{t+1} = adam.step(θ_t, ∇θ_t, ...)
```

Accelerating Training with Neuron Interaction and Nowcasting Networks

```
# Adam
for t in range(10^6):
    θ_{t+1} = adam.step(θ_t, ∇θ_t, ...)
```

- Slow convergence

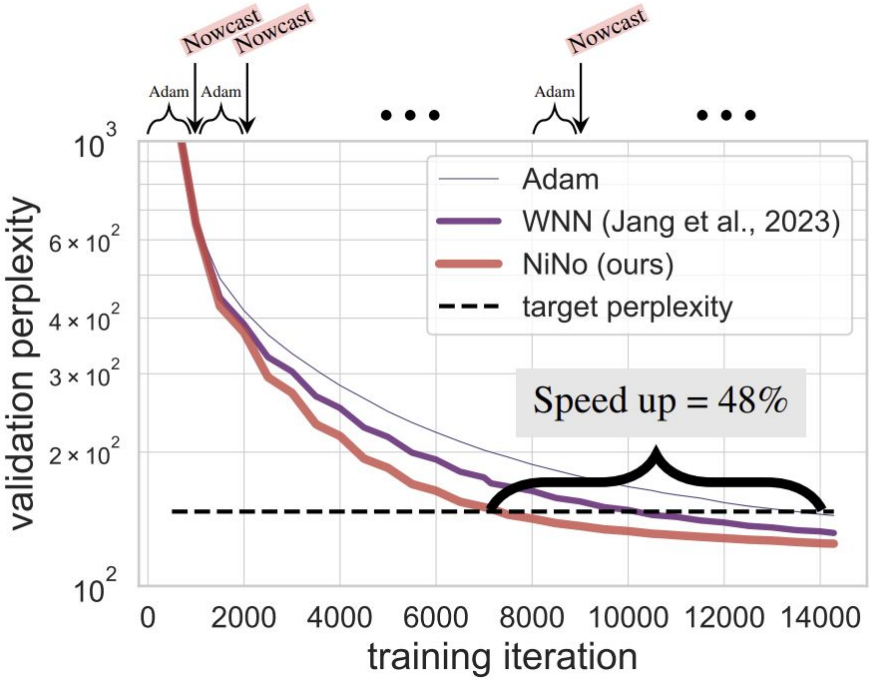
```
# Learnable optimizers (L2O)
for t in range(10^6):
    θ_{t+1} = f.predict(θ_t, ∇θ_t, ...)
```

- Faster convergence
- Overhead
- Generalization issues
- Tricky to integrate

```
# Periodically predict future parameters
for t in range(10^6):
    If t % 1000 == 0:
        θ_{t+K} = f.predict(θ_t, θ_{t-1}, ...)
    else:
        θ_{t+1} = adam.step(θ_t, ∇θ_t, ...)
```

- Faster convergence
- Less overhead
- Better generalization
- Easier to integrate

Key Example


Nowcast
Nowcast
Nowcast
Adam
Adam
Adam
validation perplexity
10^3
6×10^2
4×10^2
3×10^2
2×10^2
10^2
Adam
WNN (Jang et al., 2023)
NiNo (ours)
target perplexity
Speed up = 48%
0
2000
4000
6000
8000
10000
12000
14000
training iteration

Is optimization cost really a big issue?

- Many open-source models are already available
- GPUs are getting more powerful
- Optimization methods like Adam are good enough

Is optimization cost really a big issue?

- Many open-source
- GPUs are getting more powerful
- Optimization methods like Adam

License restrictions, harmful bias, backdoor attacks

high intra/inter company competition and price

1000 GPUs × 90 days × 24 h × $1.60/h = $3.5mln

Is optimization cost really a big issue?

- Many open-source
- GPUs are getting more powerful
- Optimization methods like Adam

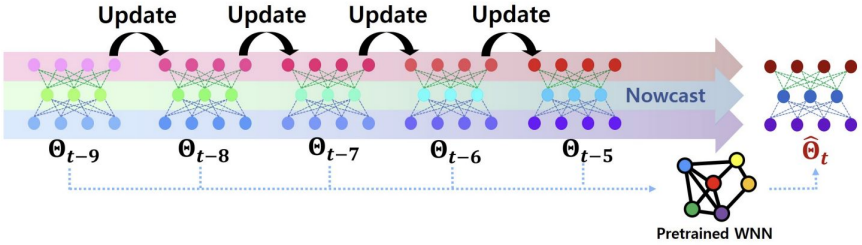
License restrictions, harmful bias, backdoor attacks

high intra/inter company competition and price

1000 GPUs × 90 days × 24 h × $1.60/h = $3.5mln

- Optimization remains the biggest cost in many pipelines
- BUT it is still hard to justify a new optimization algorithm if it saves only 10% **because of engineering overhead and unexpected outcomes**

Background: Weight Nowcaster Networks



J. Jang et al. (ICML 2023)

Training and evaluation pipeline:

1. Collect many checkpoints D={θ} in some tasks
2. Train WNN on D $\text{argmin}_\phi ||\Delta\tilde{\theta}_{\tau+k} - \Delta\hat{\theta}_{\tau+k}||_1$
3. Use the trained WNN on any new task

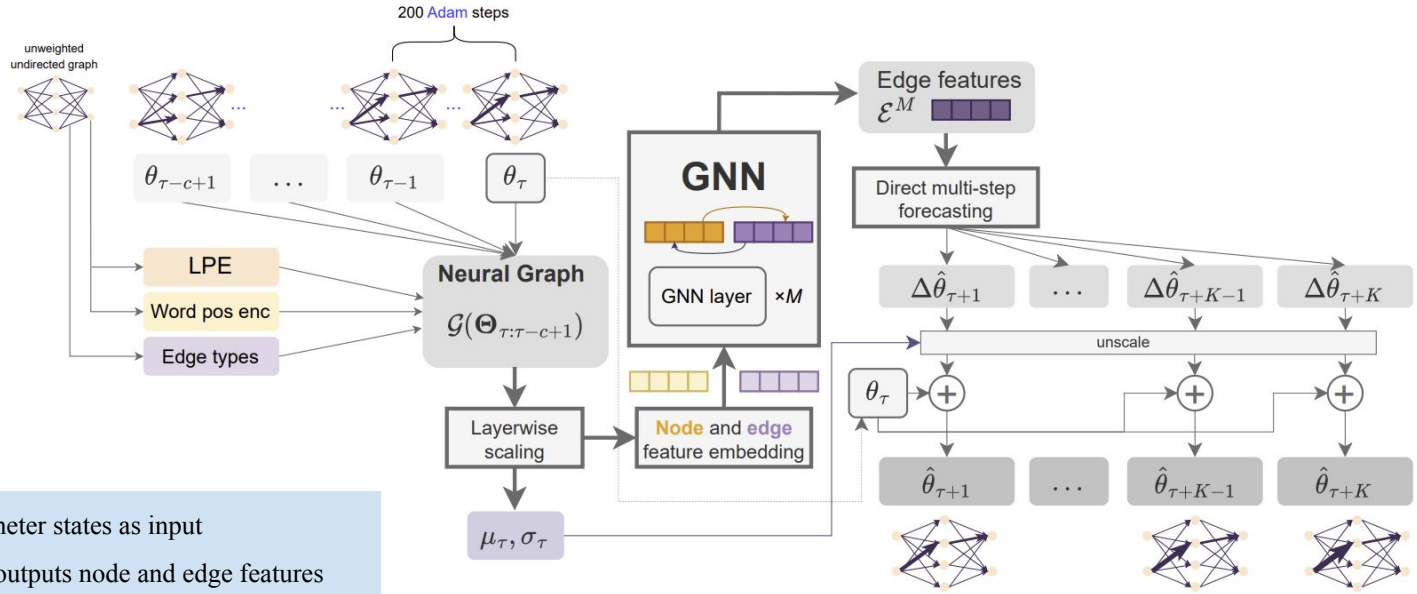
```
# Periodically predict future parameters
for t in range(10^6):
    If t % 1000 == 0:
        θ_{t+K} = f.predict(θ_t , θ_{t-1} , ...)
    else:
        θ_{t+1} = adam.step(θ_t , ∇θ_t , ...)
```

From WNN to NiNo

```
# WNN
for t in range(10^6):
    If t % 1000 == 0:
        θ_{t+K, i} = f.predict(θ_{t, i} , θ_{t-1, i}  , ...)  # ∀ i
    else:
        θ_{t+1, i} = adam.step(θ_{t, i} , ∇θ_{t, i} , ...)  # ∀ i
```

```
# NiNo
for t in range(10^6):
    If t % 1000 == 0:
        θ_{t+K} = f.predict(θ_t , θ_{t-1}  , ...; G)  # jointly for all params
    else:
        θ_{t+1, i} = adam.step(θ_{t, i} , ∇θ_{t, i} , ...)  # ∀ i
```

NiNo



- Take 5 past parameter states as input
- GNN inputs and outputs node and edge features
- Predict parameter deltas at multiple future horizons

Training and evaluation pipeline

- Collect many checkpoints D={θ} in some tasks (in-distribution tasks)
- Train NiNo on D
- Use the trained NiNo on any new task task

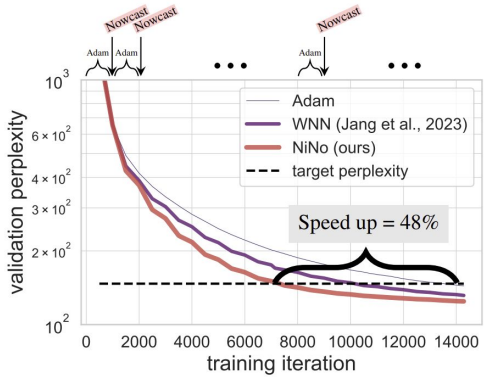


Table 1: **In-distribution and out-of-distribution tasks.**

	In-distribution tasks				Out-of-distribution tasks				
	FM/16	C10/16	LM1B/3-24	LM1B/2-32	FM/32	C10/32	C100/32	LM1B/3-64	Wiki/3-64
Final training loss	0.25±0.06	0.91±0.1	5.94±0.03	5.85±0.04	–	–	–	–	–
#models	300	300	200	200	–	–	–	–	–
#params	14K	15K	1.2M	1.6M	56K	57K	63K	3.3M	3.3M
Target (validation) metric	Acc	Acc	Perplexity	Perplexity	Acc	Acc	Acc	Perplexity	Perplexity
Target value	89.5%	66.0%	352	319	90.5%	72.5%	39%	181	147
Adam #steps	8606	8732	23000	23500	8269	8607	8341	23500	13500
NiNo #steps	4582	3775	11500	12000	4395	4323	4646	12000	7000

- Speedup = (13500-7000)/13500=48%
- Scaled to 100M models

3. Future directions

Future directions

1. Generative vs deterministic models
2. Scalability vs equivariance
3. Handling diverse ranges of parameter values
4. Generalization (new architectures, new tasks)
5. More compelling applications needed
    a. Optimization (l2o, nowcasting)
    b. Model alignment
    c. Model editing

Our Lab in Montréal

Samsung AI Lab (SAIL) Montréal
https://www.sait.samsung.co.kr/saithome/about/labs.do

Interns for Fall 2025-Winter 2026: jobs.sail.montreal@gmail.com

Looking for potential students to join UdeM: boris.knyazev@umontreal.ca

SAMSUNG

Université de Montréal