

Hyperparameter Transfer

A Recipe for Efficient and Robust Scaling

Gage DeZoort

Princeton University

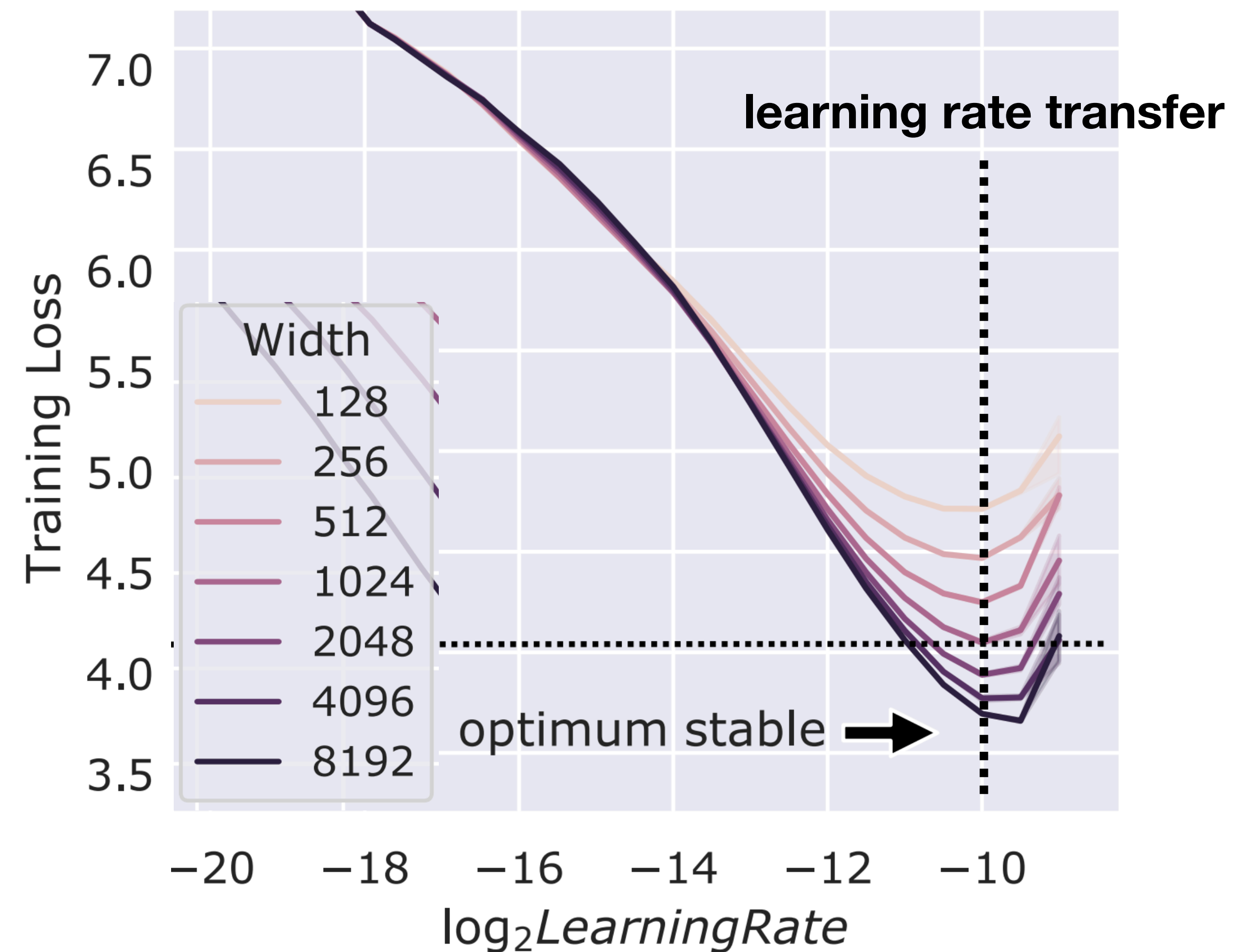
HAMLET-PHYSICS 2026

Copenhagen · 19 August 2026

Re-optimizing big models is painful...

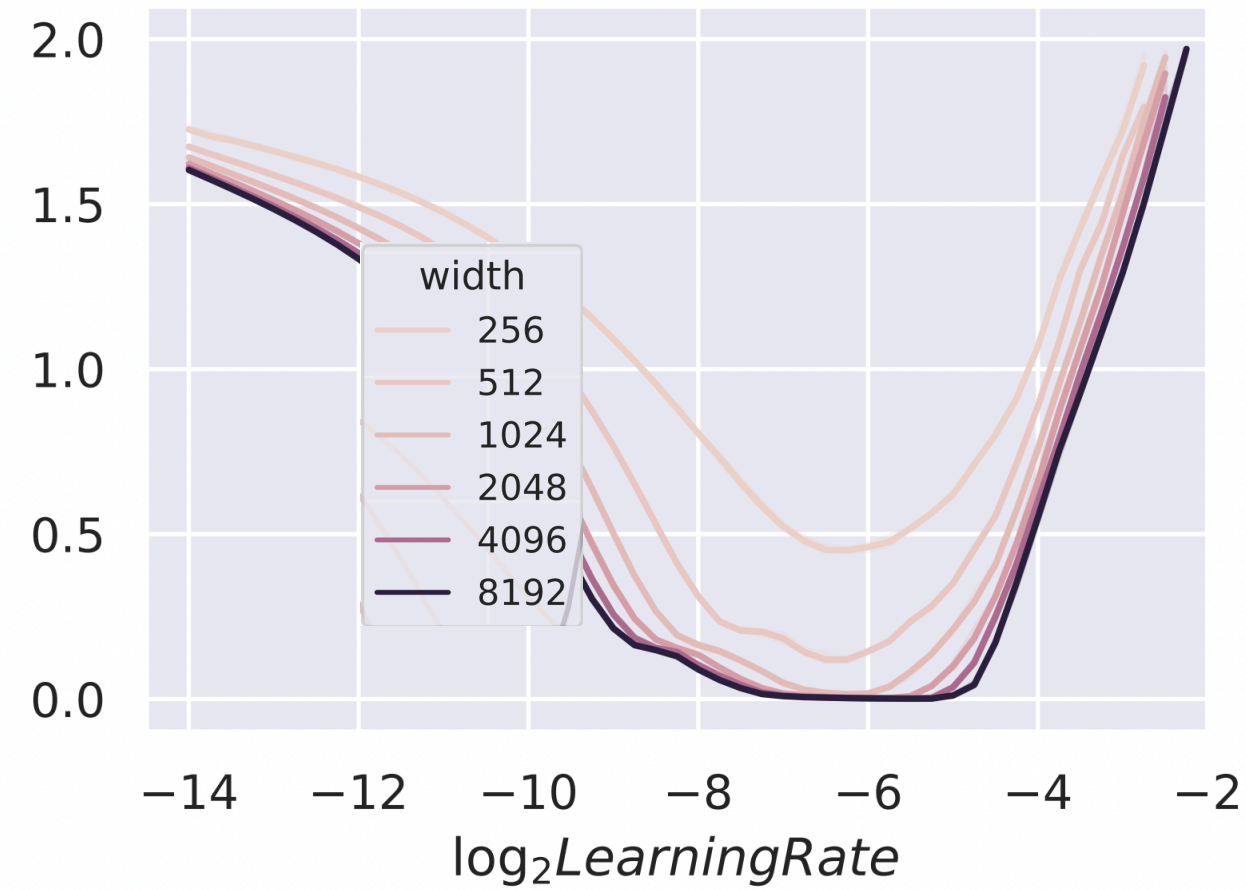


What if we could just tune smaller models?



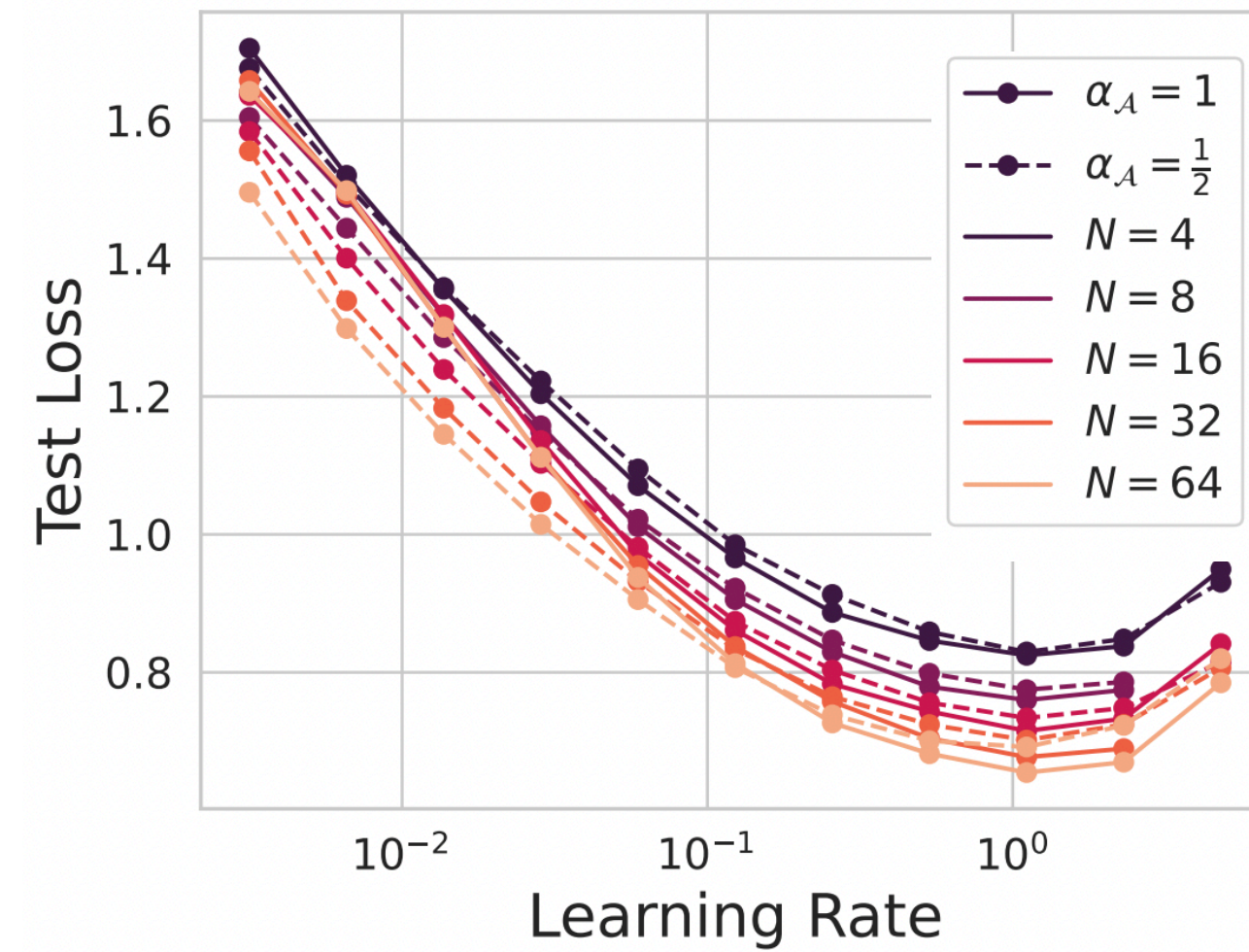
μ P Width Scaling

<https://arxiv.org/abs/2203.03466>



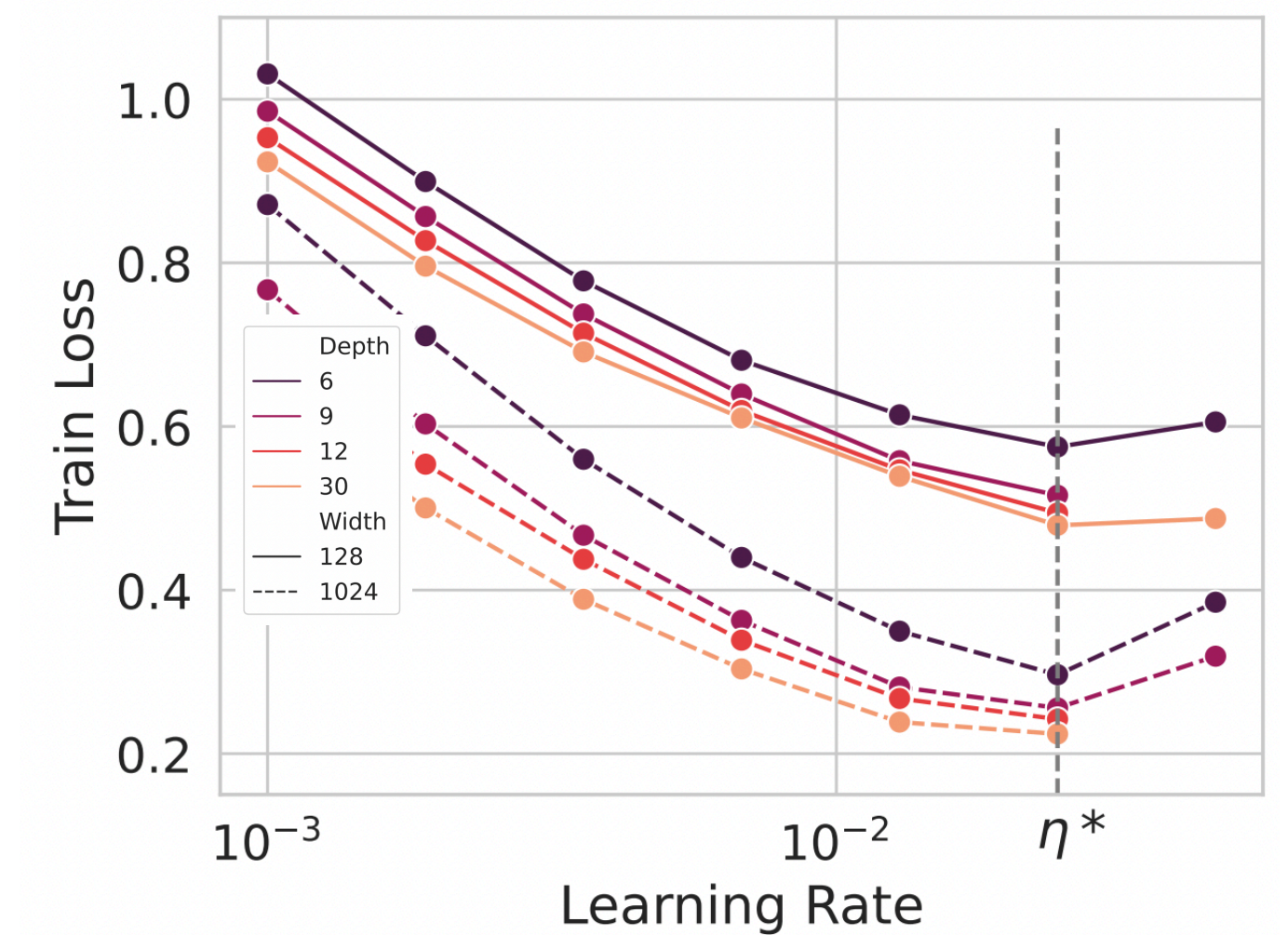
DMFT for Attention

<https://arxiv.org/pdf/2405.15712>



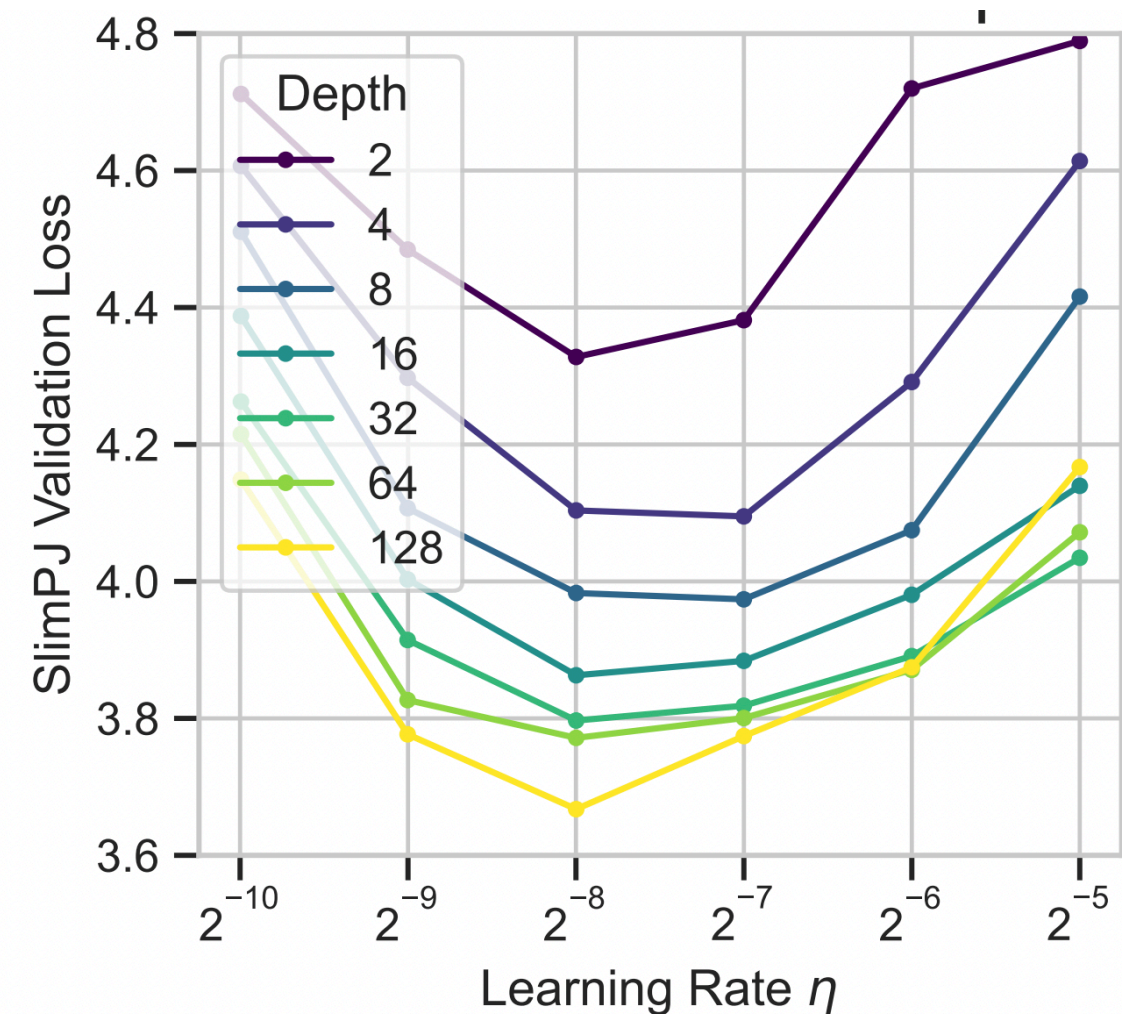
Residual Depth Scaling

<https://arxiv.org/pdf/2309.16620>



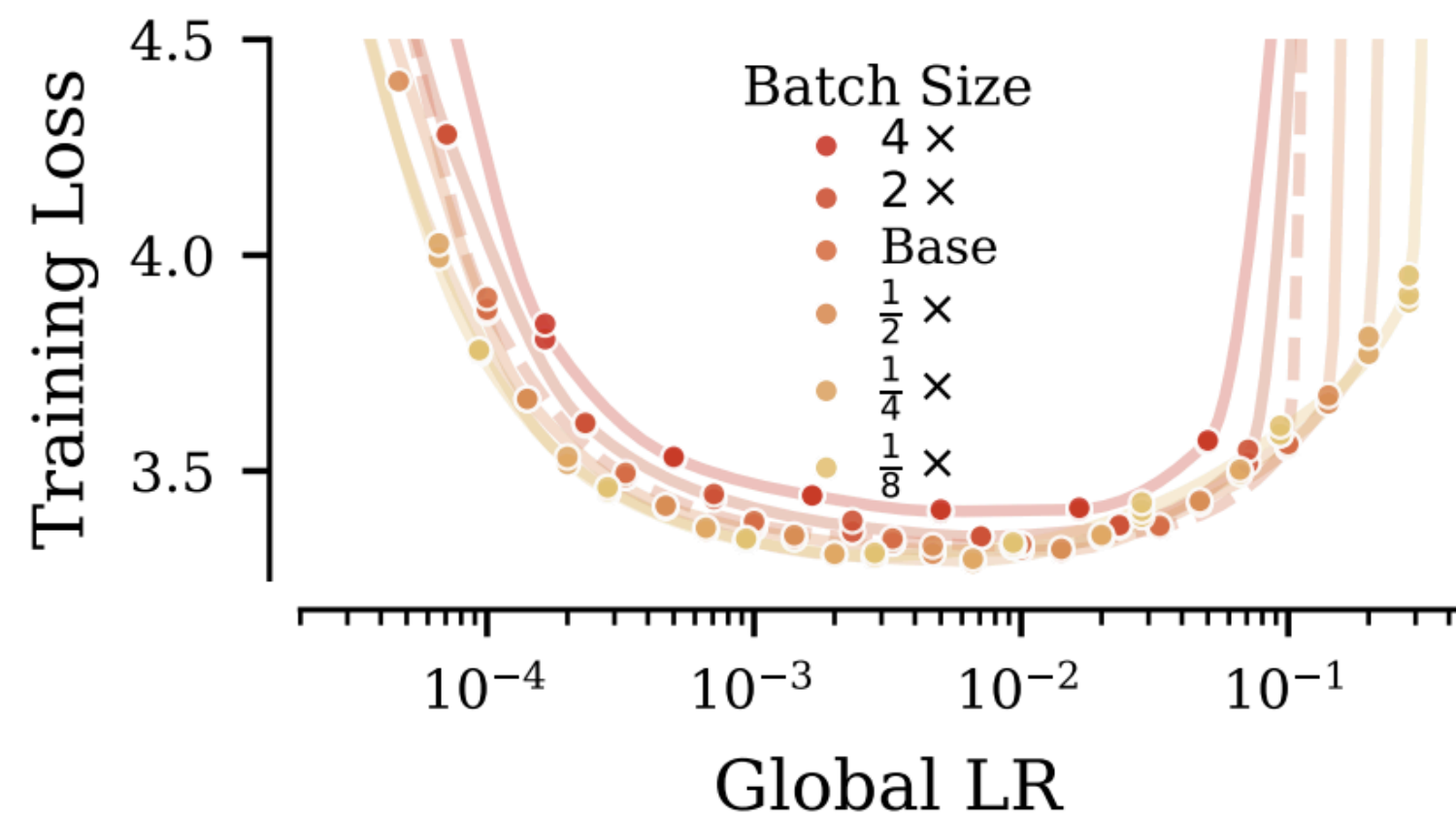
CompleteP

<https://arxiv.org/pdf/2505.01618>



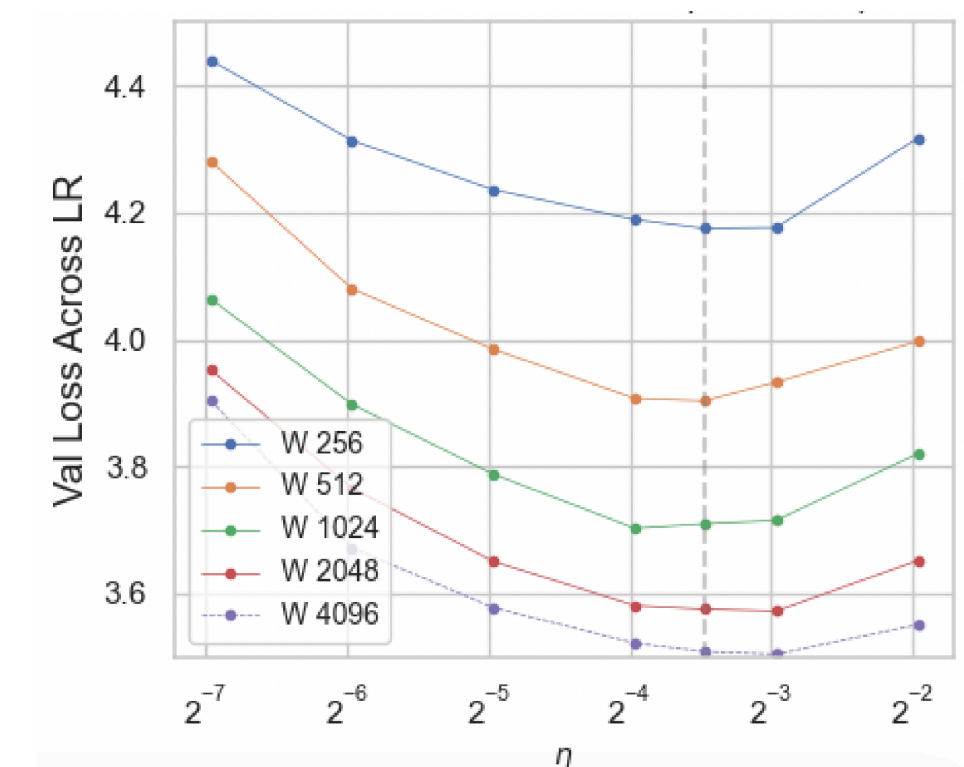
CompletedP

<https://arxiv.org/pdf/2512.22382>



MoE Models

<https://arxiv.org/pdf/2601.20205>



Scientific ML is already scaling up! e.g.

Particle Physics:

OmniLearned with ~1B jets and 20-500M params

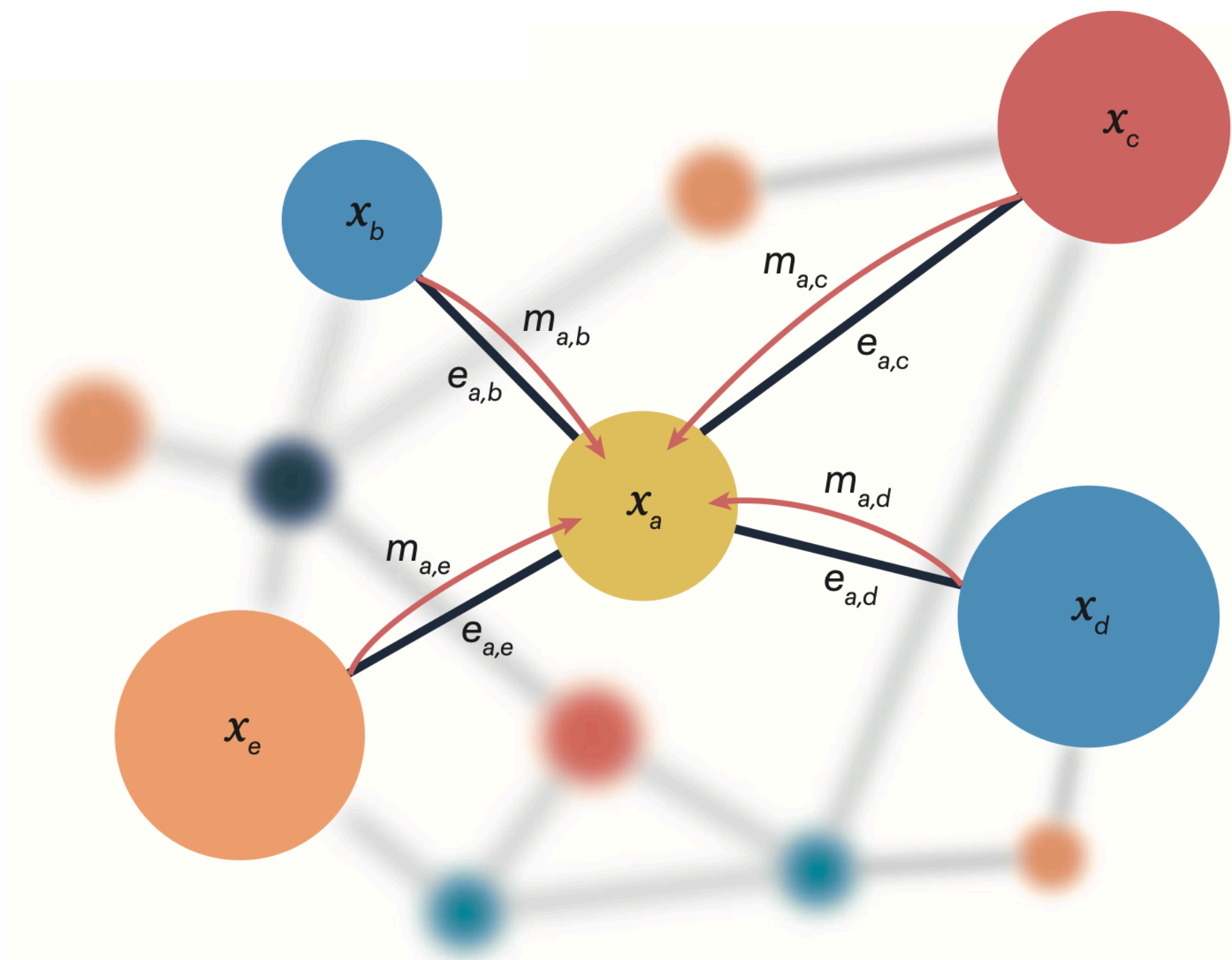
EveNet with 500M events and 20M params

Carpe Datum scaling laws on ~8B jets

Astronomy:

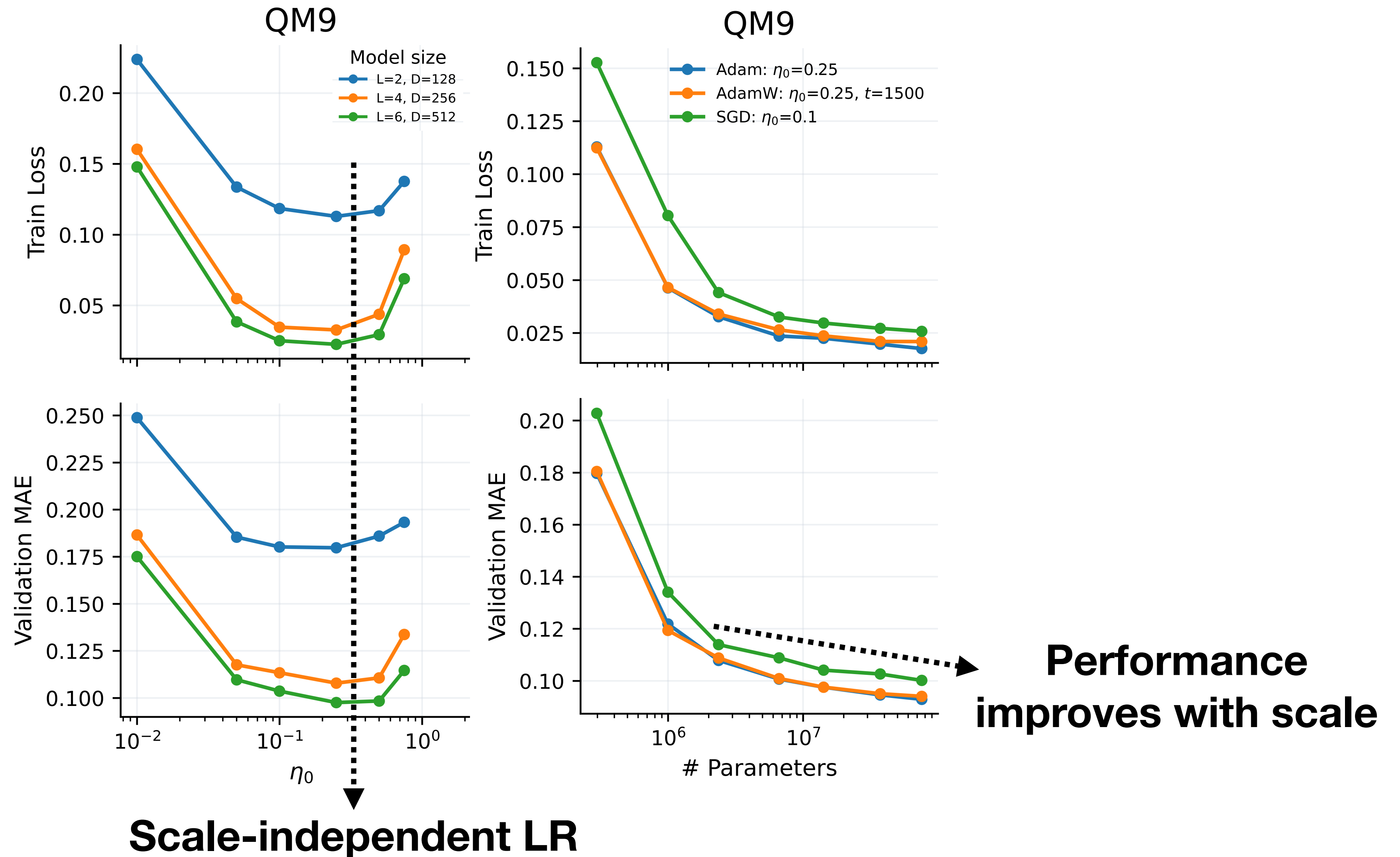
AION-1: 200M observations and 300M-3.1B params

GNNs are **well-suited to scientific data** (relational inductive biases, sparse representations)... but *difficult to scale*

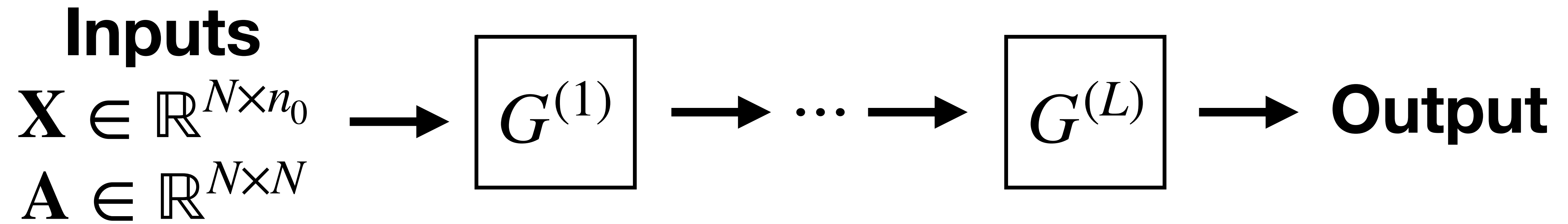


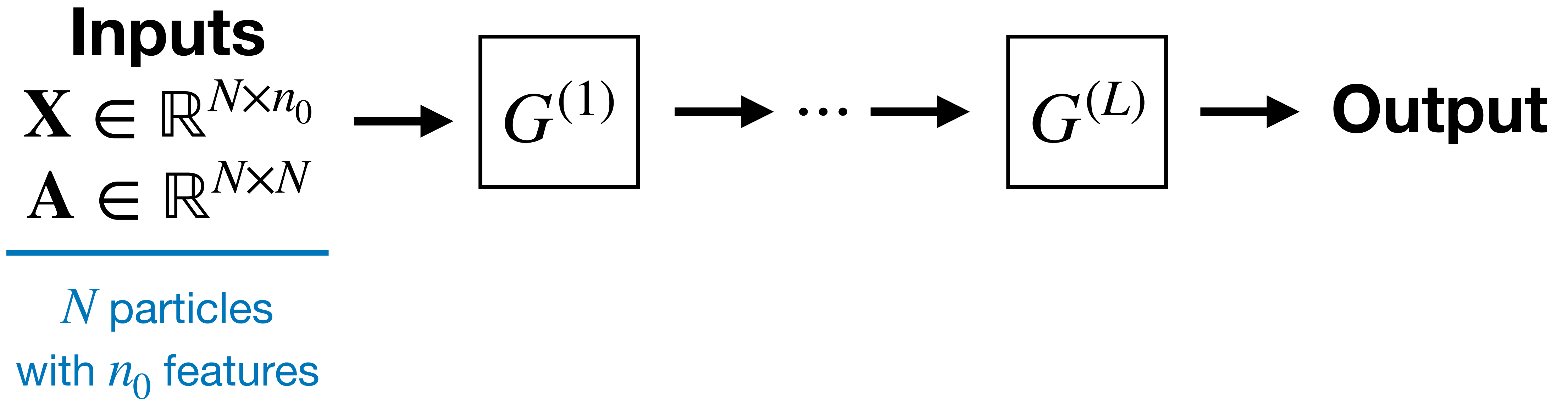
<https://www.nature.com/articles/s42254-023-00569-0>

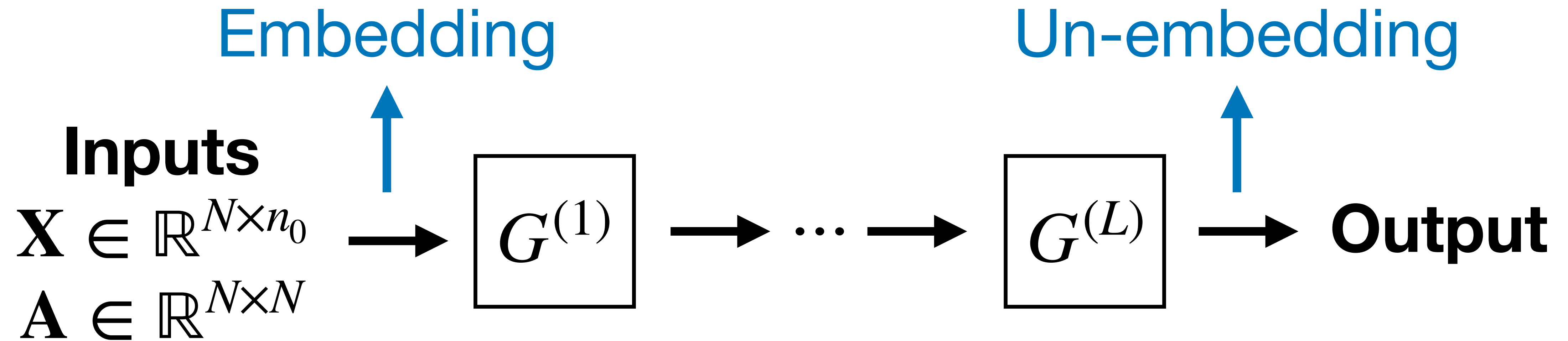
Hyperparameter Transfer for GNNs

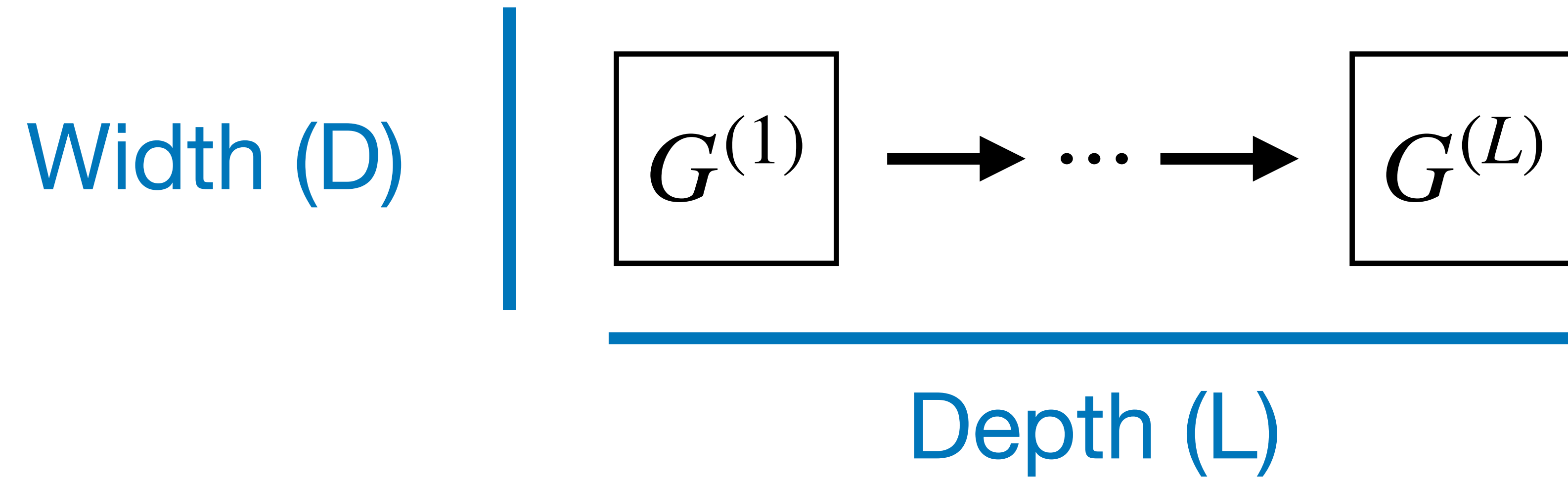


The goal is to identify a ***parameterization*** — a prescription for how to modify initialization schemes and hyperparameters across model scales



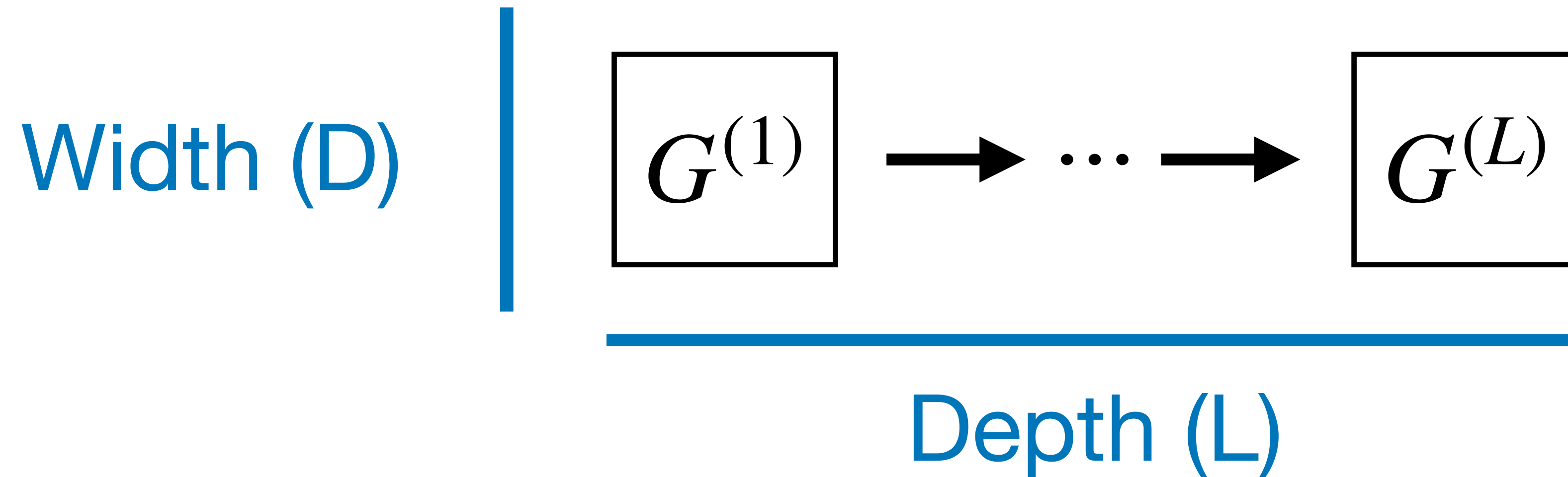






Feature Learning Limit:

Initialize so that training dynamics are stable with respect to scale,
activations $g_{u,i}^{(l)} \sim O(1)$ and **updates** $\Delta g_{u,i}^{(l)} \sim O(1)$



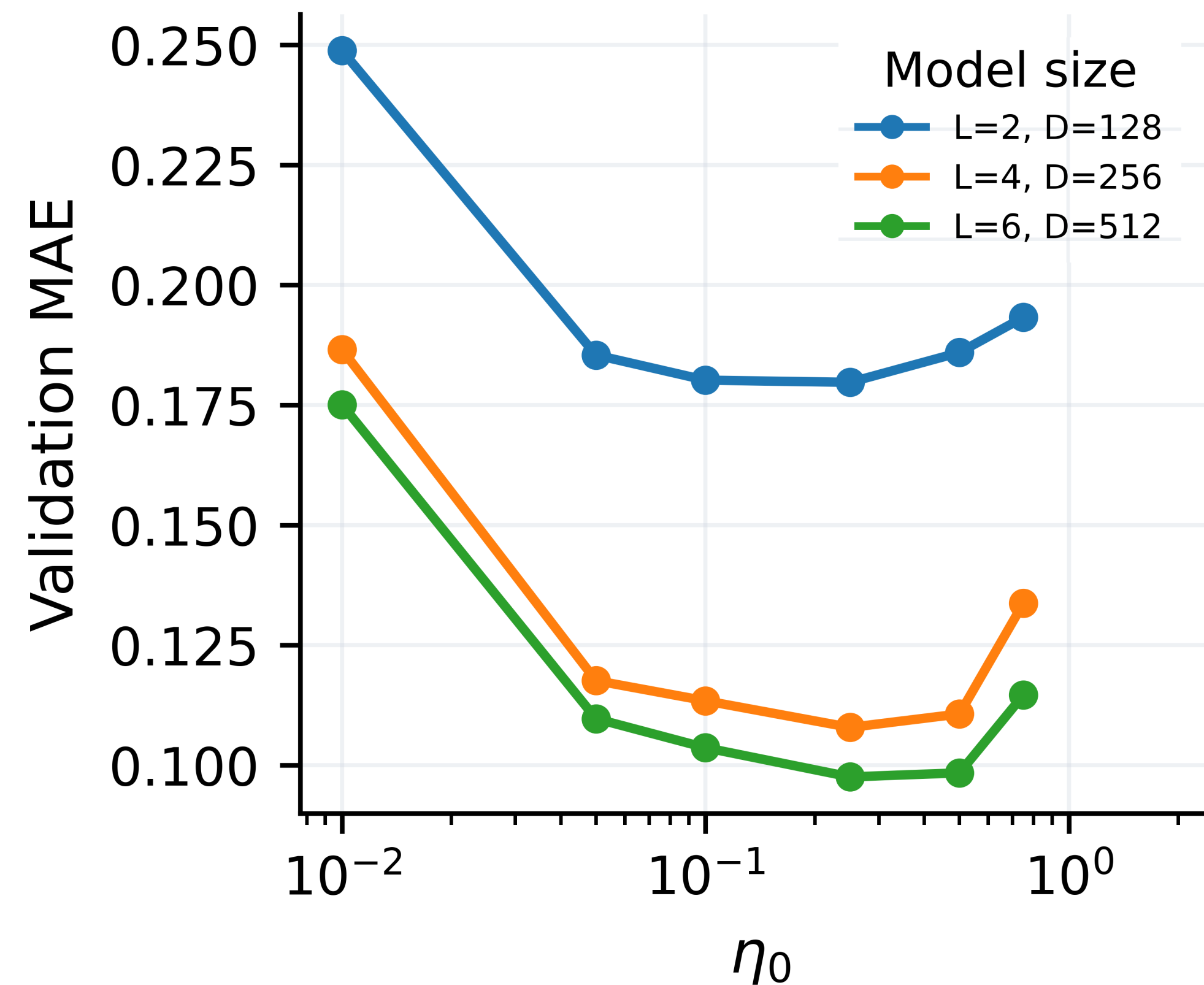
scale-dependent *learning rates*

SGD Learning Rate : $\underline{\eta} = \underline{\eta_0} DL$

Adam Learning Rate : $\underline{\eta} = \underline{\eta_0} / \sqrt{D}$

base learning rate transfers

Base Learning Rate Transfers



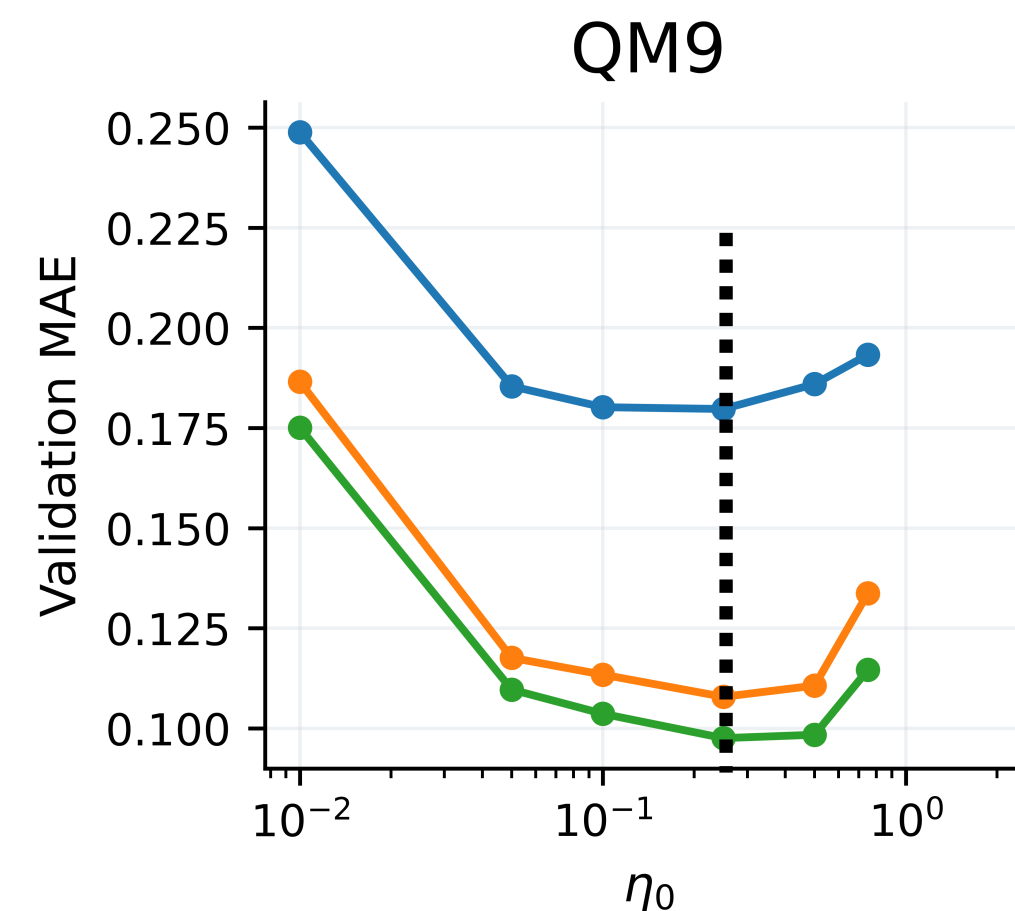
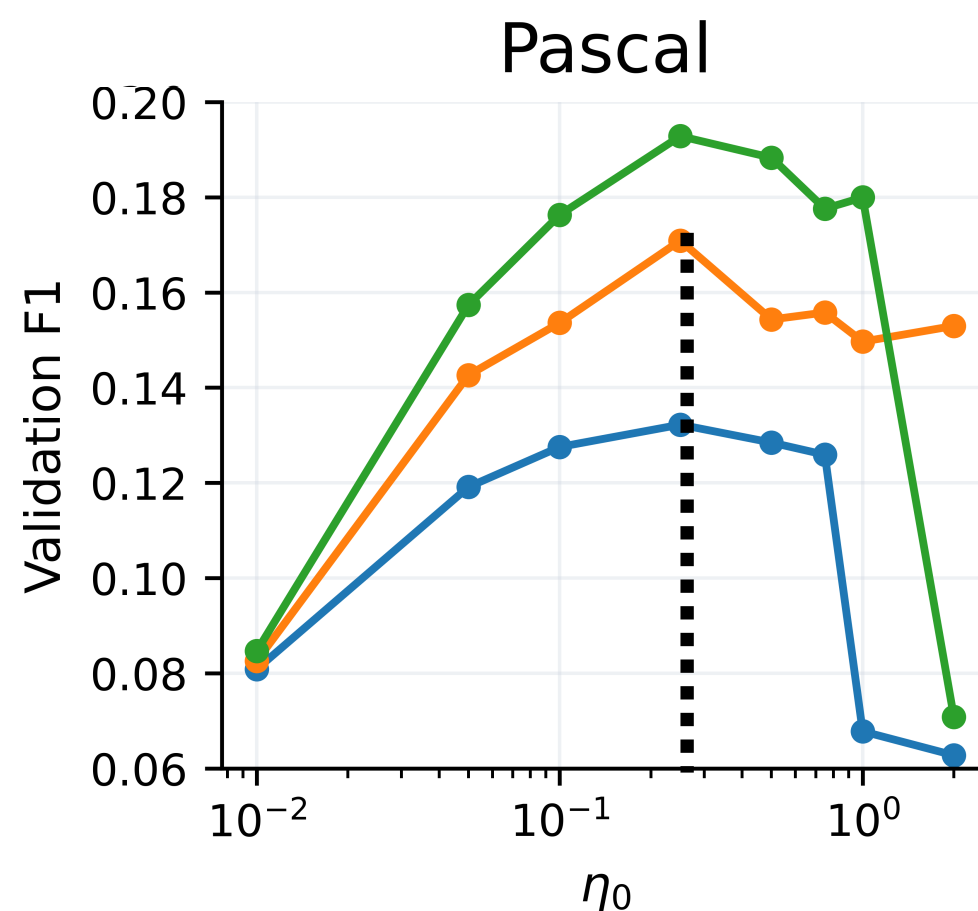
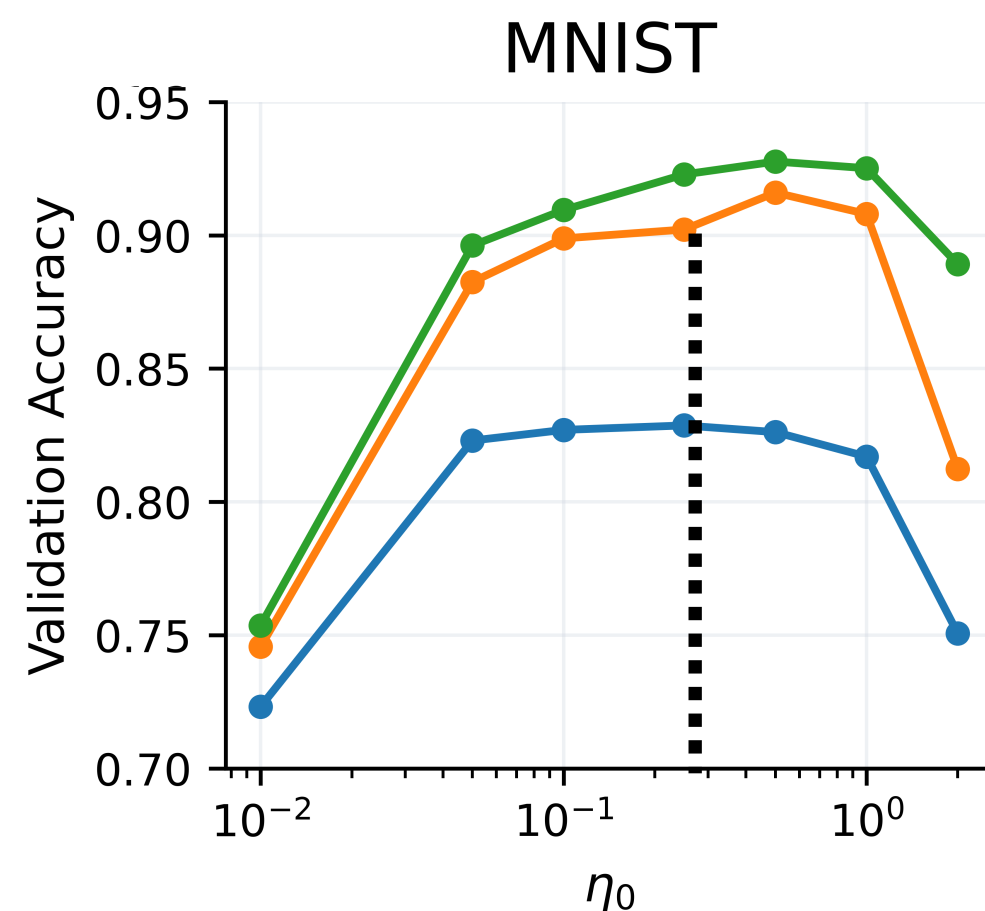
$$\eta = \eta_0 / \sqrt{128}$$

$$\eta = \eta_0 / \sqrt{256}$$

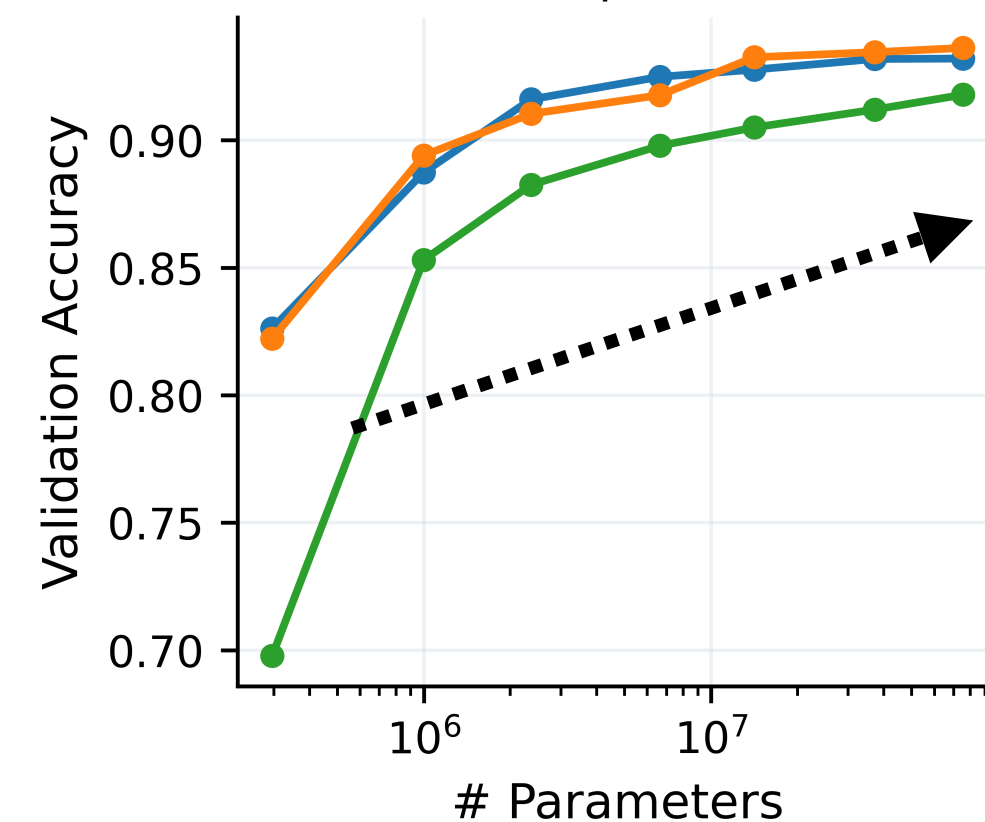
$$\eta = \eta_0 / \sqrt{512}$$

Training via Adam

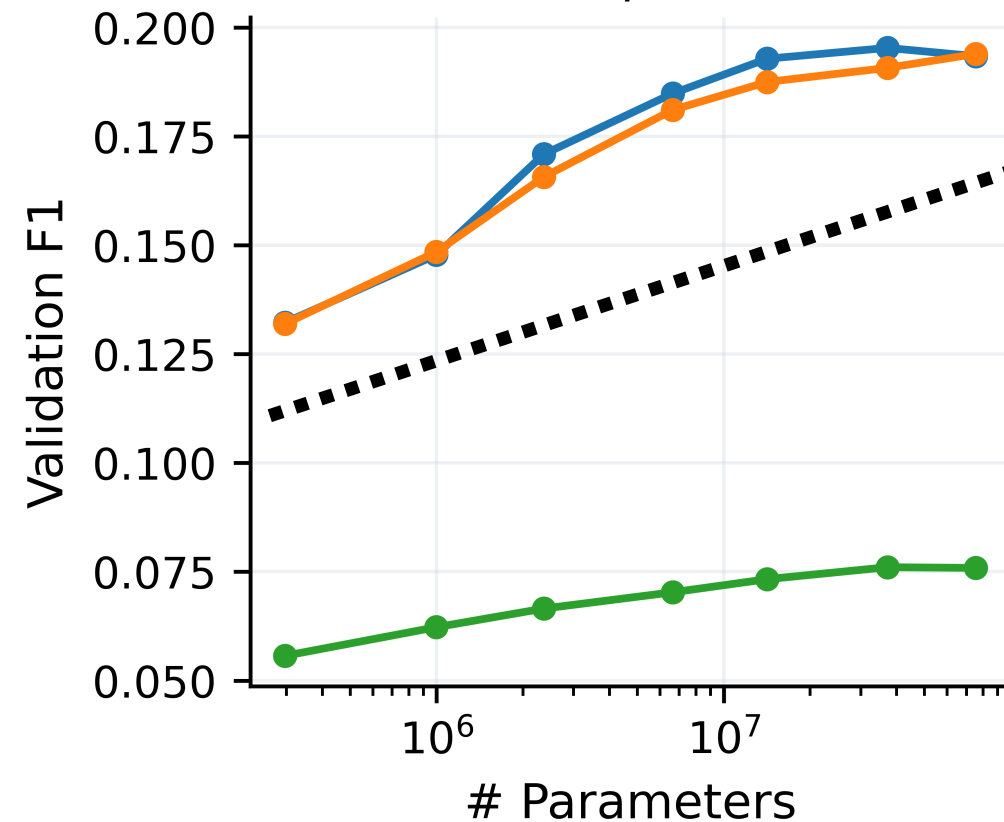
$$\Delta\theta = -\frac{\eta_0}{\sqrt{D}} \frac{\hat{m}_\theta}{\sqrt{\hat{v}_\theta + \epsilon}}$$



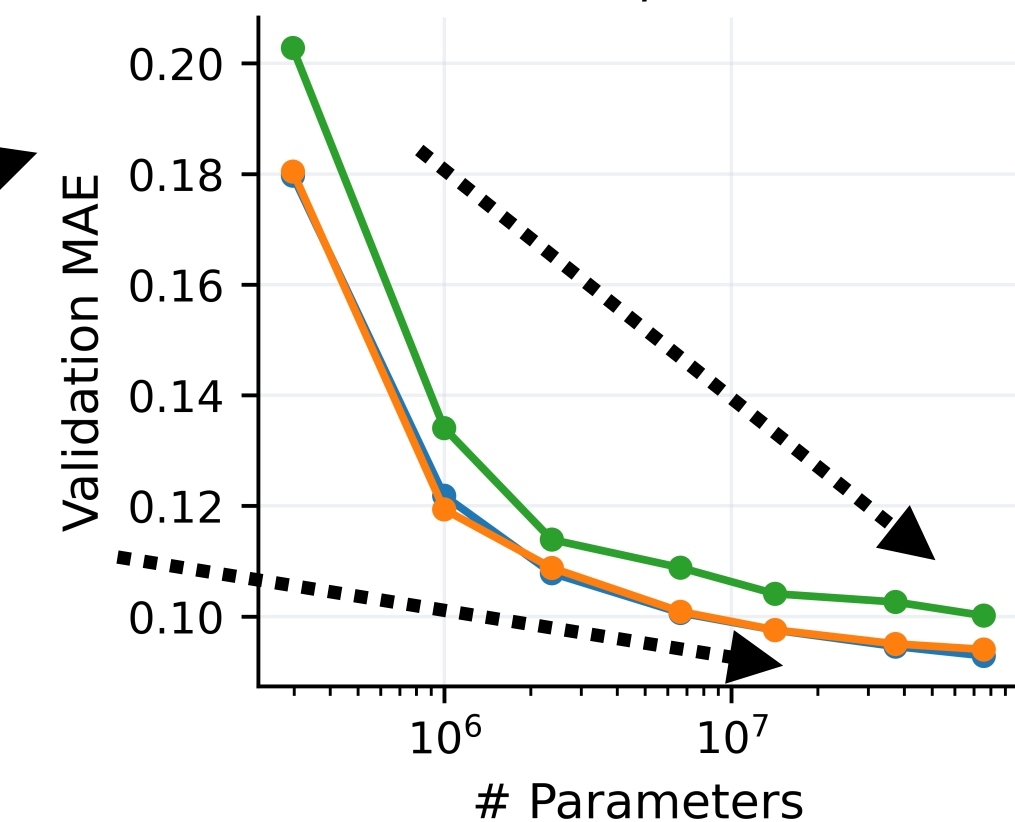
Learning rates transfer



Graph Classification



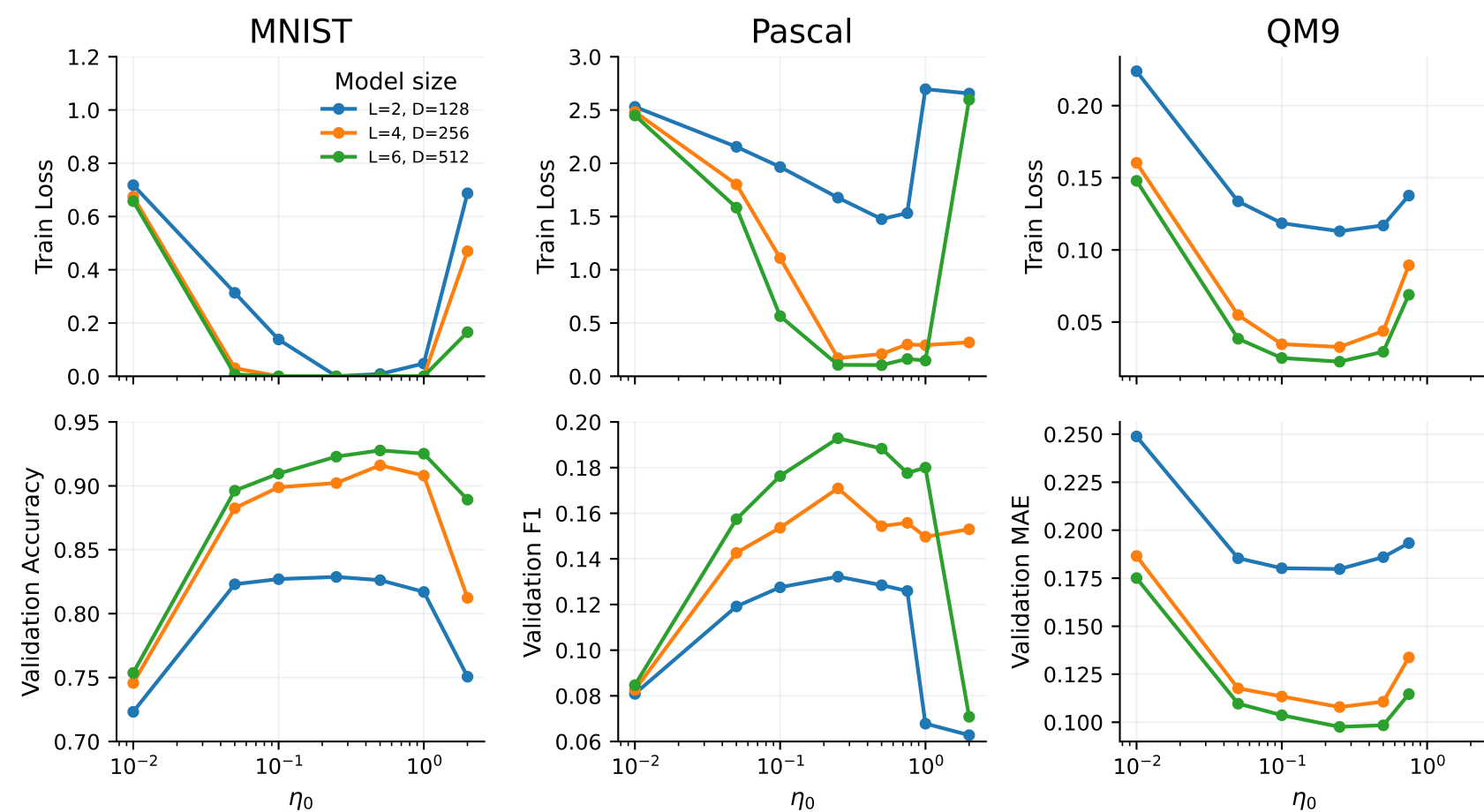
Node Classification



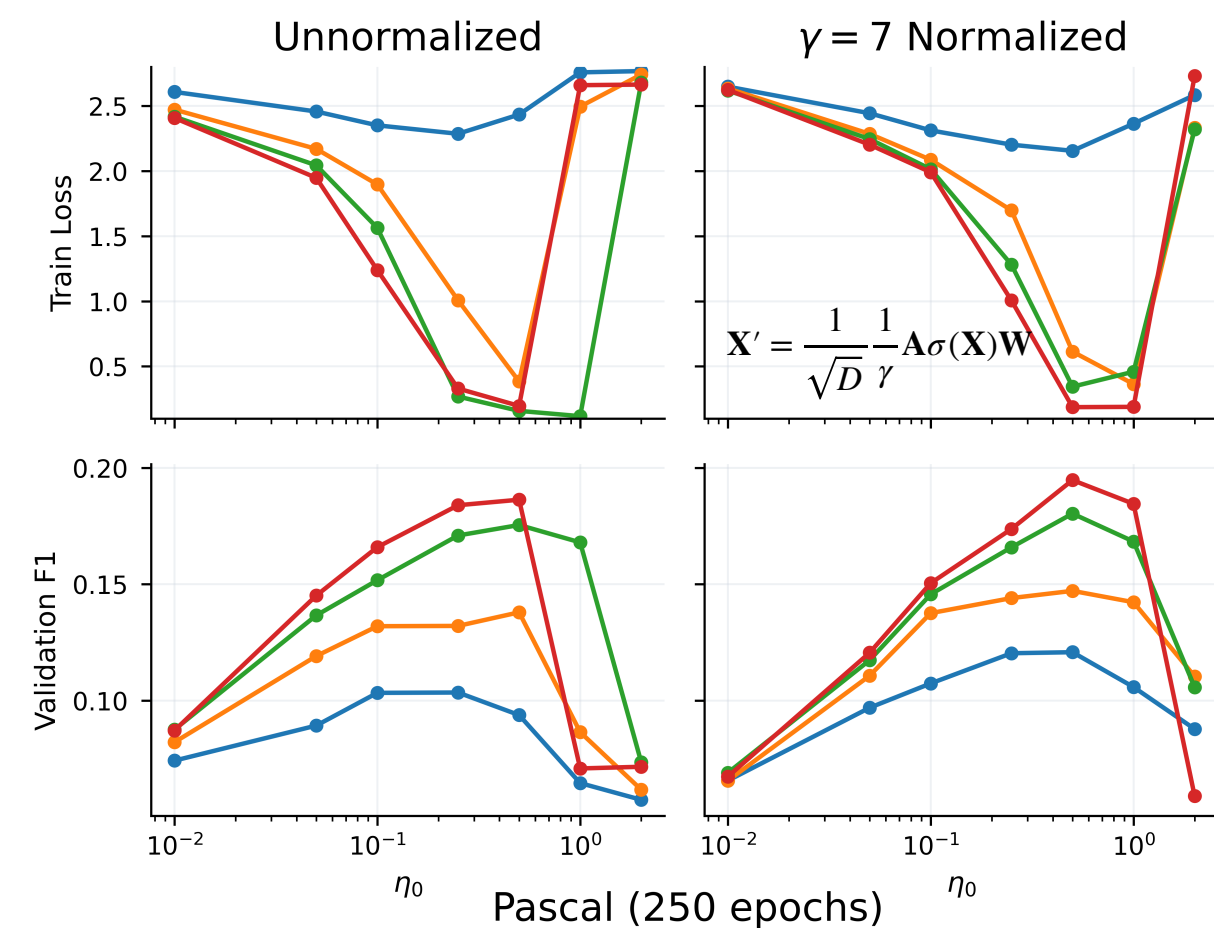
Graph Regression

Performance improves with scale

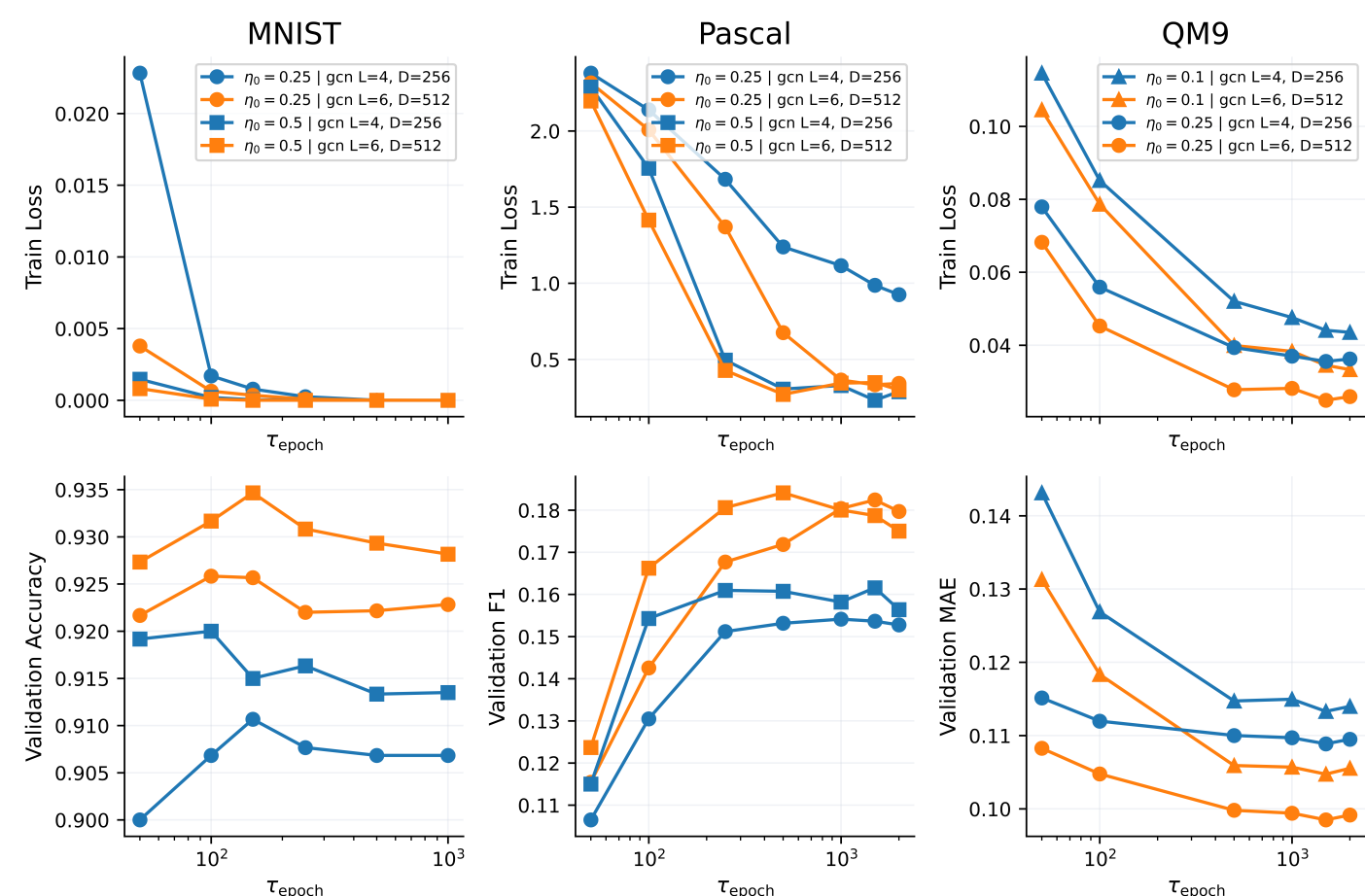
Transfer for SGD, Adam, AdamW



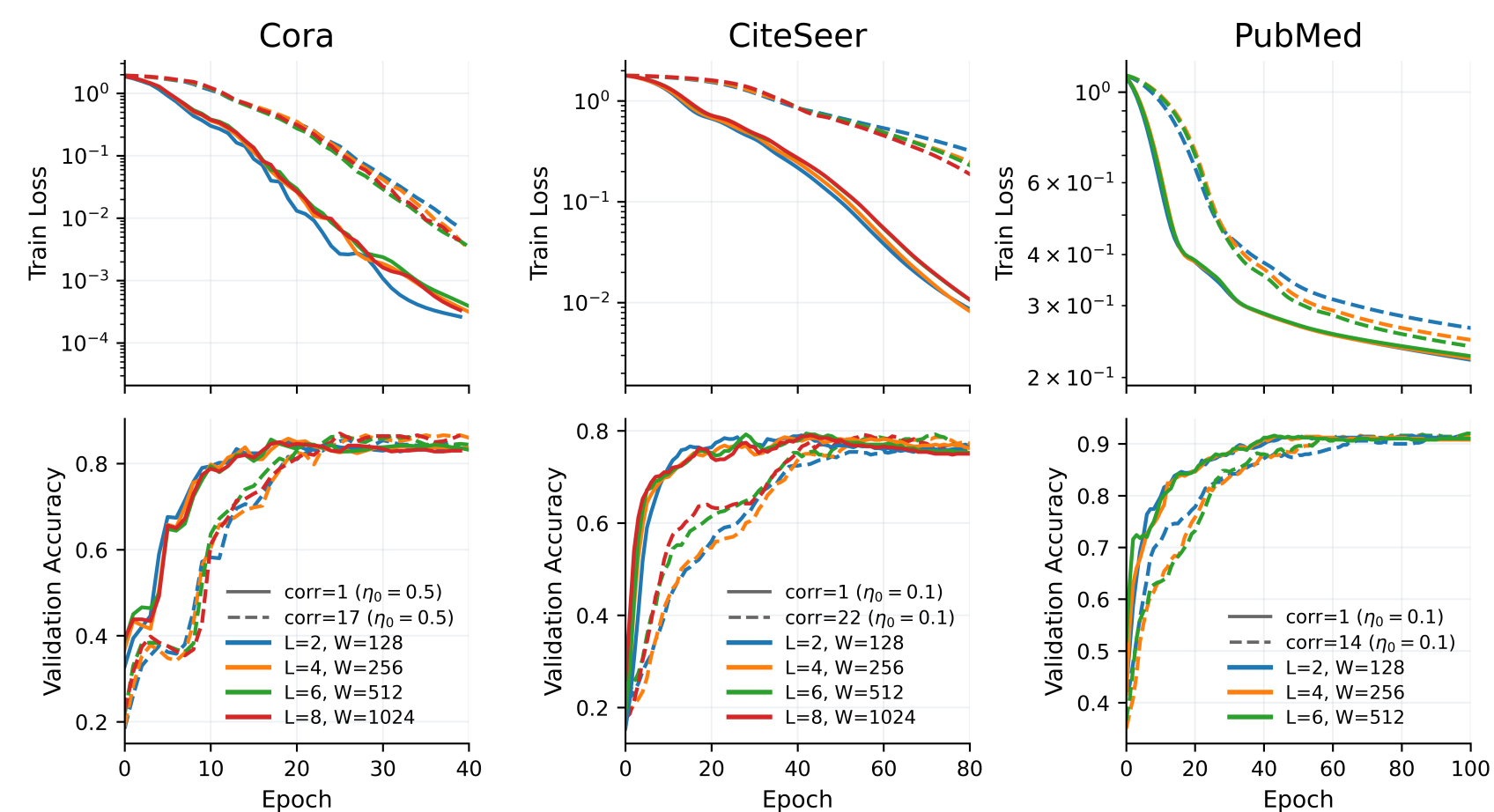
Message passing normalization



Weight decay transfer for AdamW



Input statistics can impact transfer

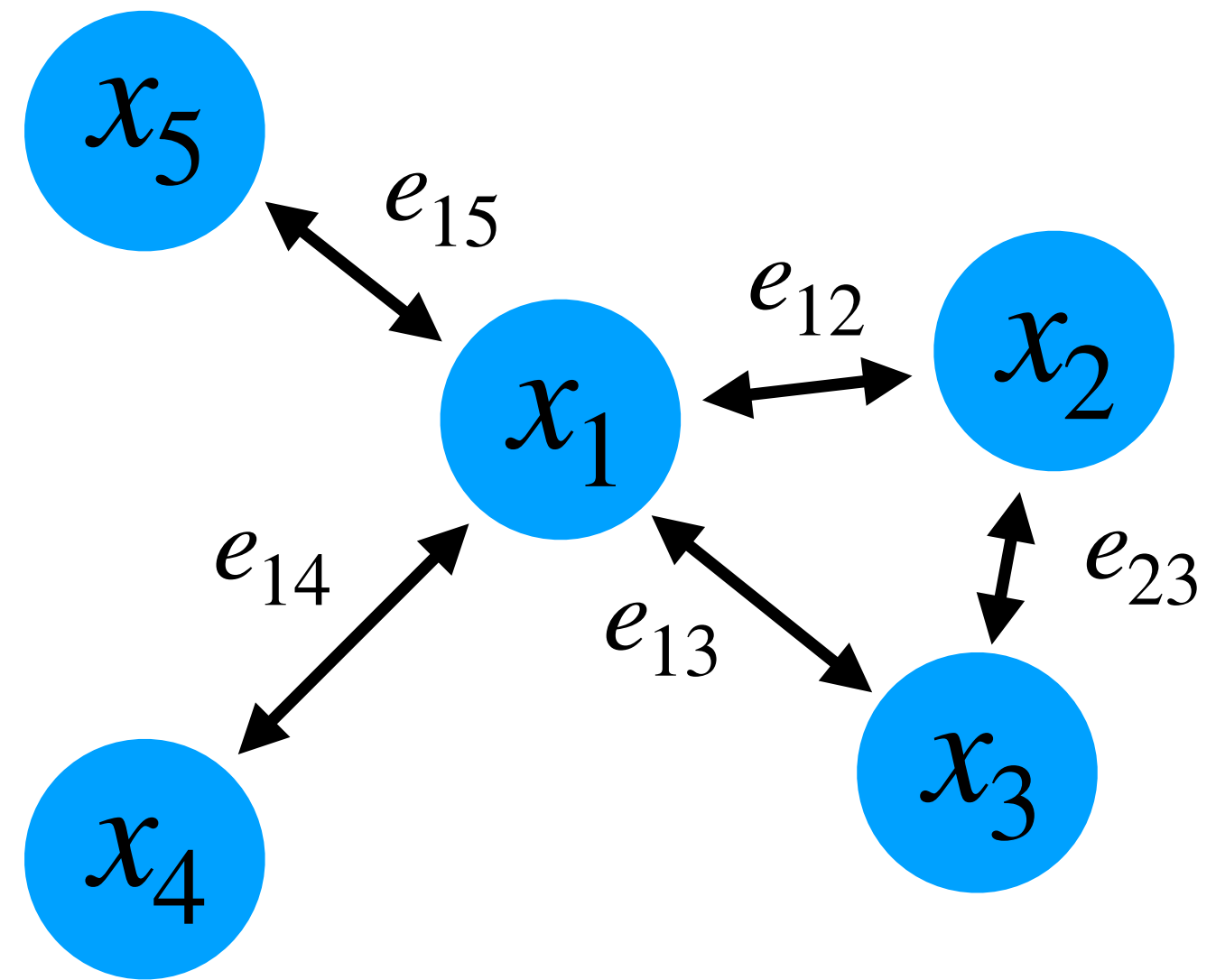


Hyperparameter Transfer in GNNs

Gage DeZoort and Boris Hanin

<https://arxiv.org/abs/2607.05017>

Coupled Particle-Edge Networks



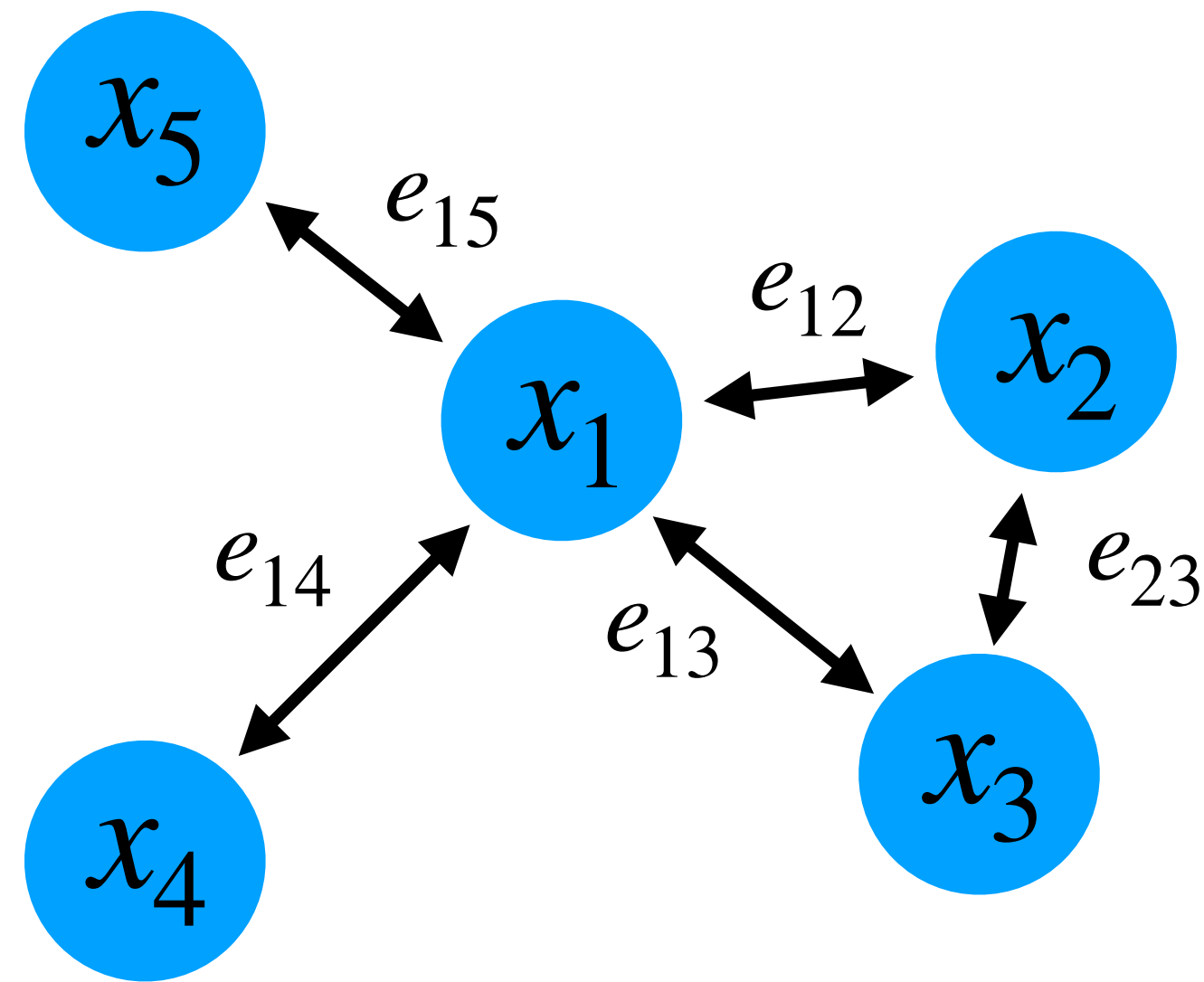
Particle Features

$$\mathbf{X} \in \mathbb{R}^{N \times D}$$

Edge Features

$$\mathbf{E} \in \mathbb{R}^{M \times D}$$

Coupled Particle-Edge Networks



Particle Features

$$\mathbf{X}' \leftarrow f_p(\mathbf{X}) + \underline{f_{e \rightarrow p}(\mathbf{E})}$$

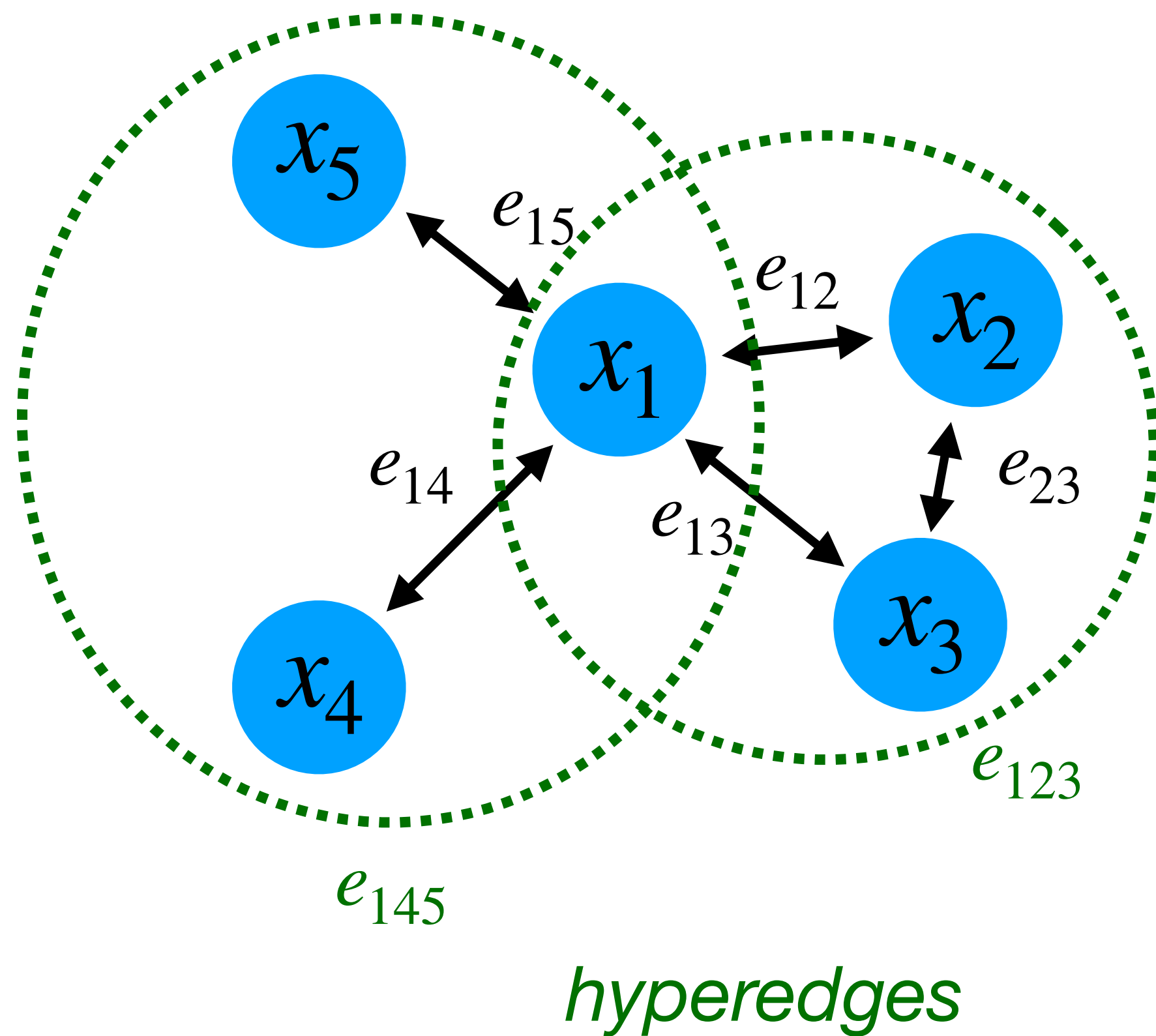
“lowering” operation

Edge Features

$$\mathbf{E}' \leftarrow f_e(\mathbf{E}) + \underline{f_{p \rightarrow e}(\mathbf{X})}$$

“lifting” operation

Coupled Particle-Edge Networks



Particle Features

$$\mathbf{X}' \leftarrow f_p(\mathbf{X}) + \underline{f_{e \rightarrow p}(\mathbf{E})}$$

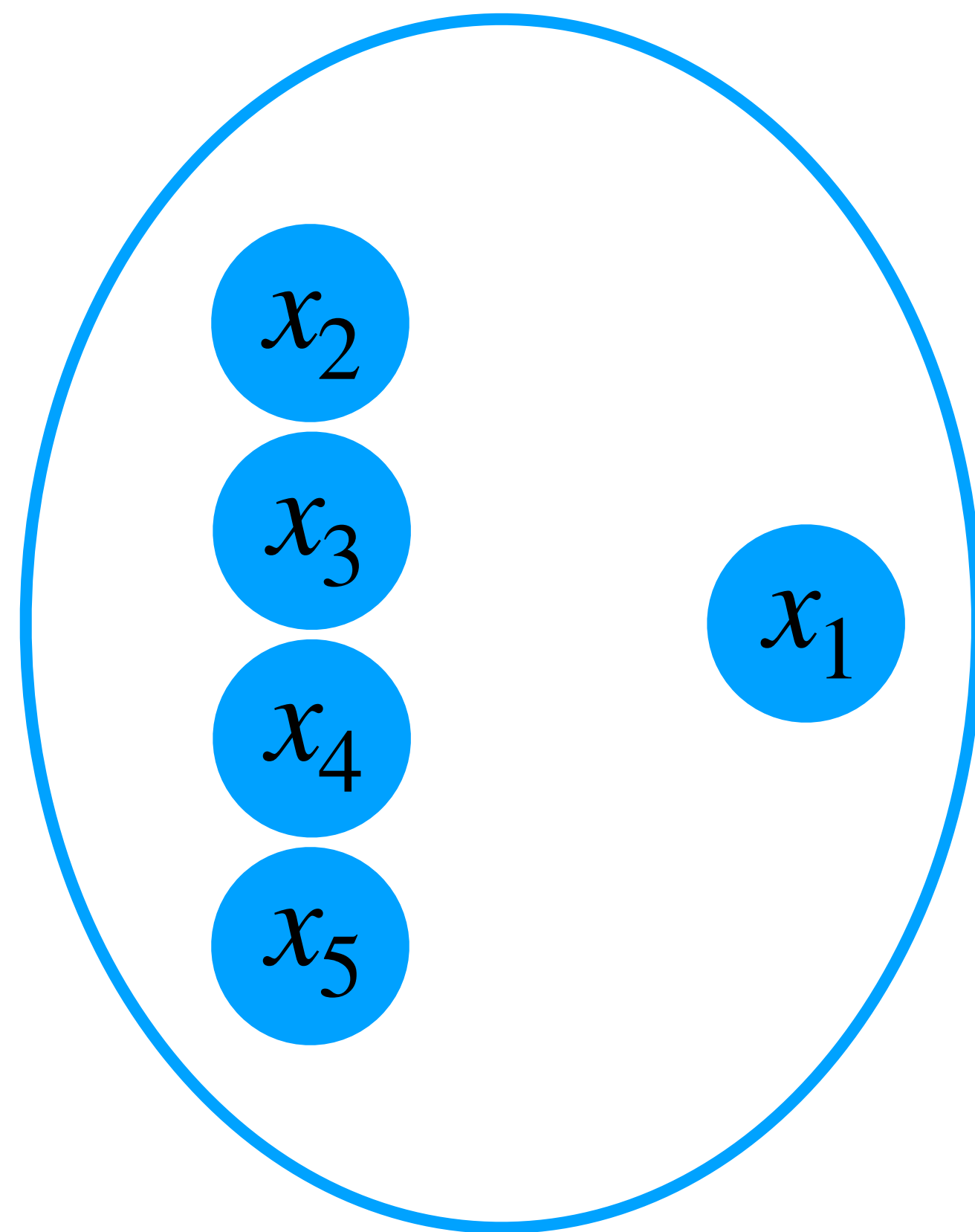
“lowering” operation

Edge Features

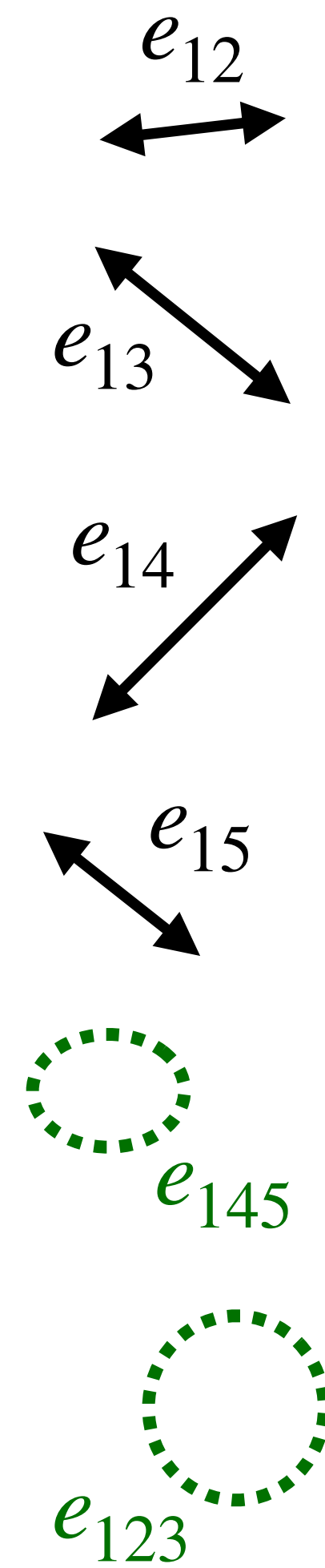
$$\mathbf{E}' \leftarrow f_e(\mathbf{E}) + \underline{f_{p \rightarrow e}(\mathbf{X})}$$

“lifting” operation

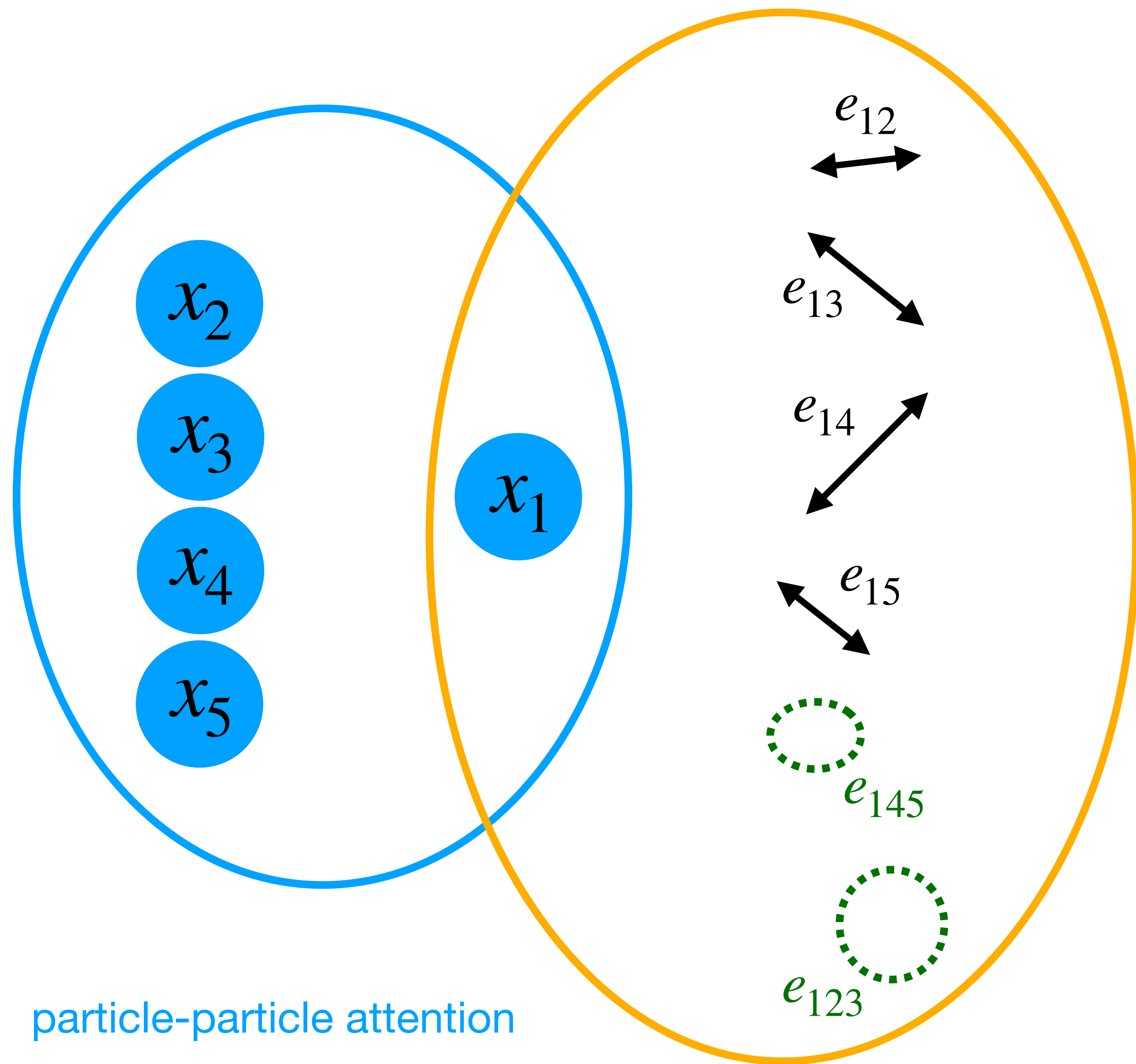
$$\mathbf{A}_p \in \mathbb{R}^{N \times N}$$



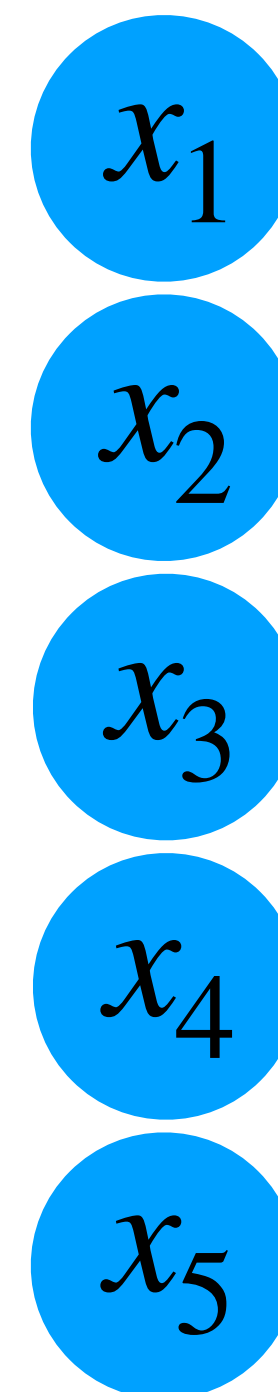
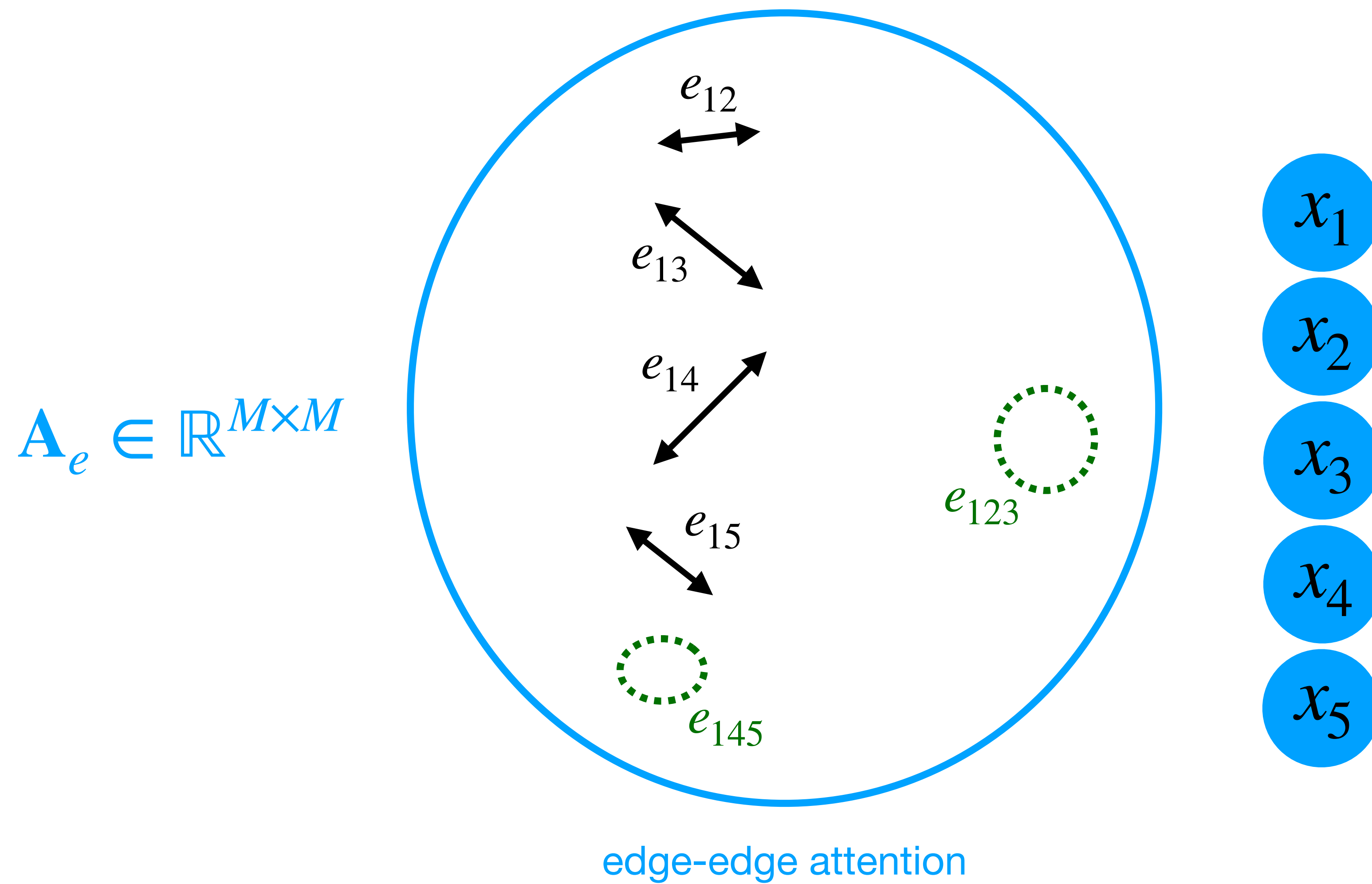
particle-particle attention

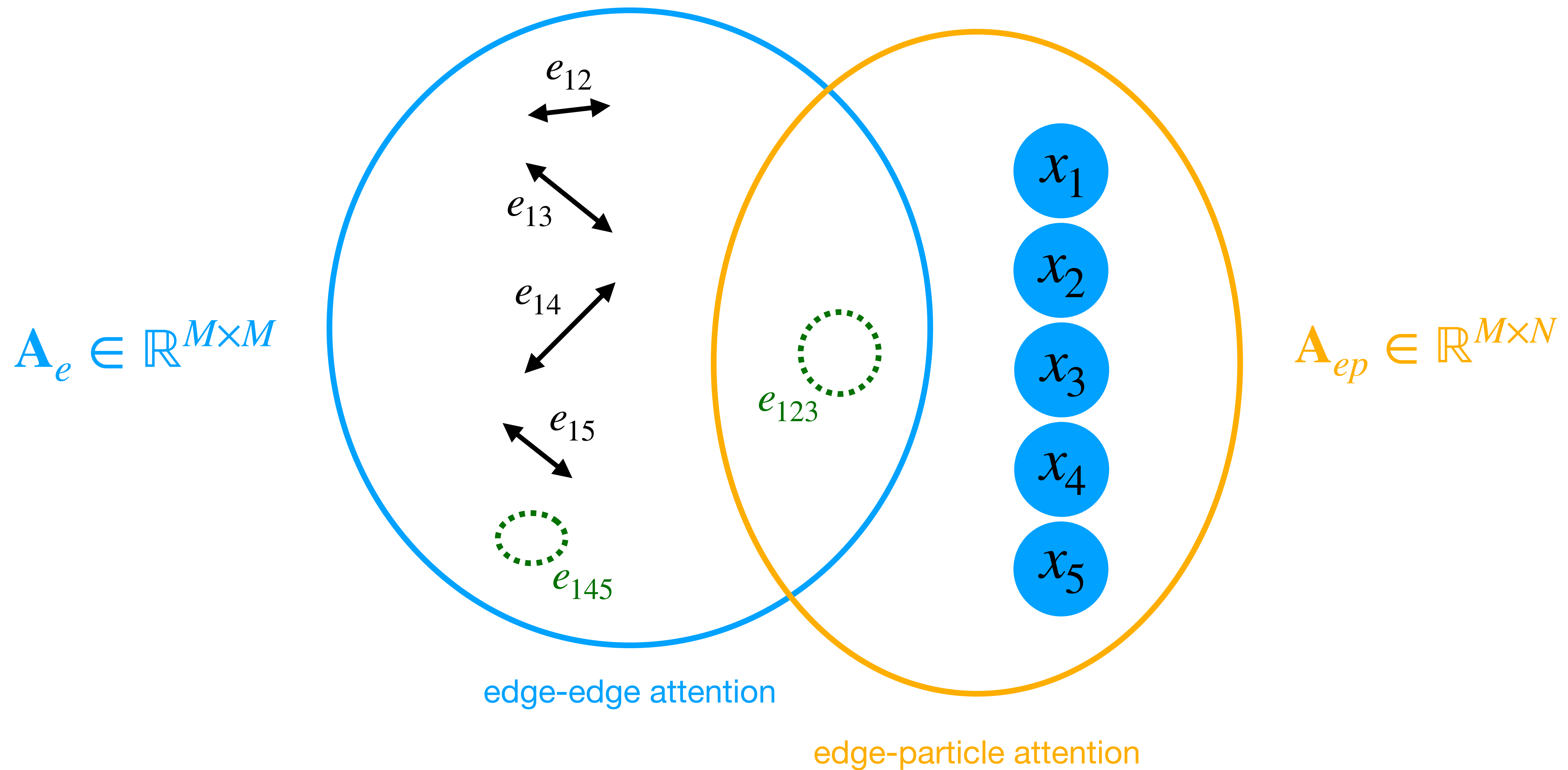


$$\mathbf{A}_p \in \mathbb{R}^{N \times N}$$

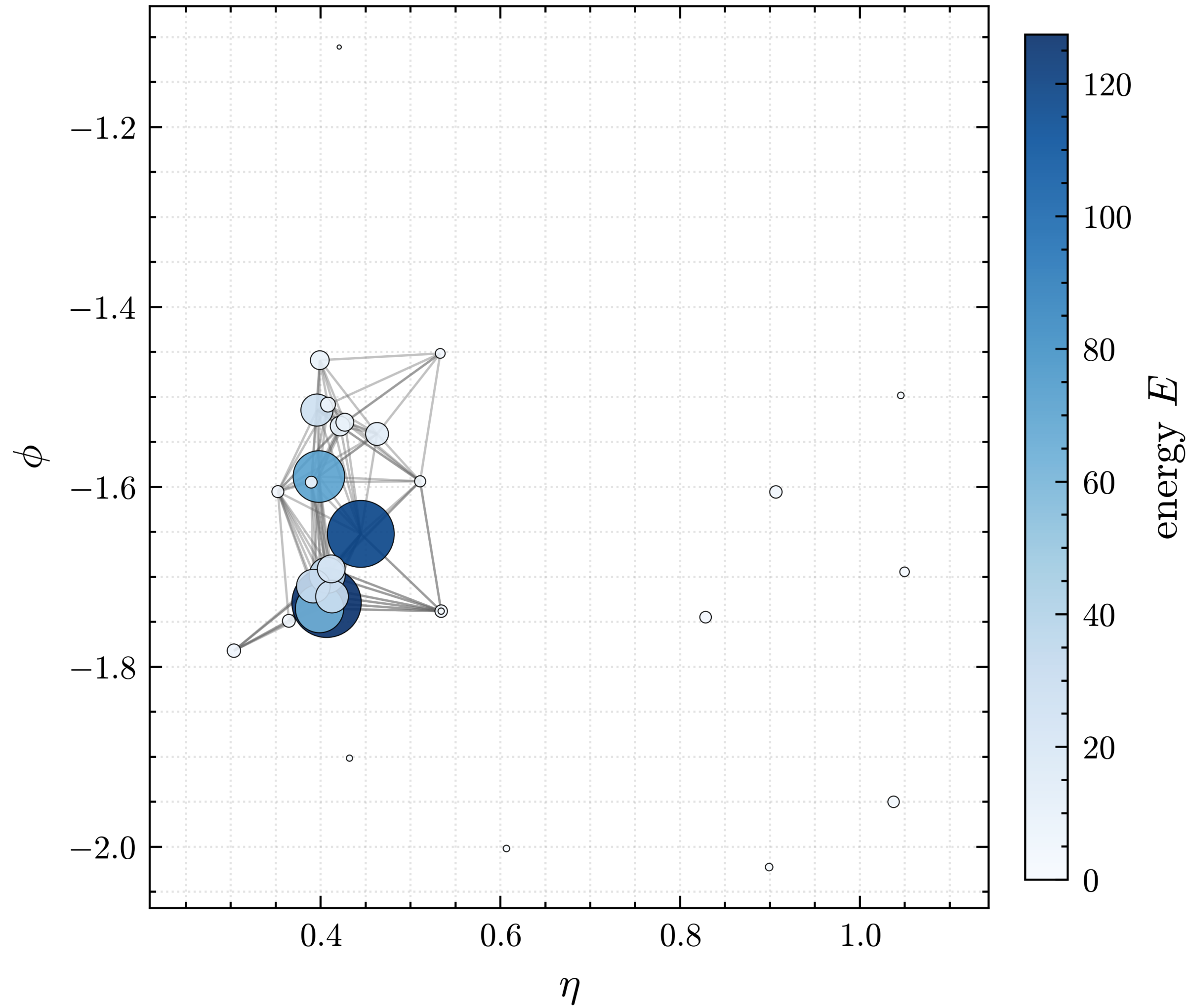


$$\mathbf{A}_{pe} \in \mathbb{R}^{N \times M}$$

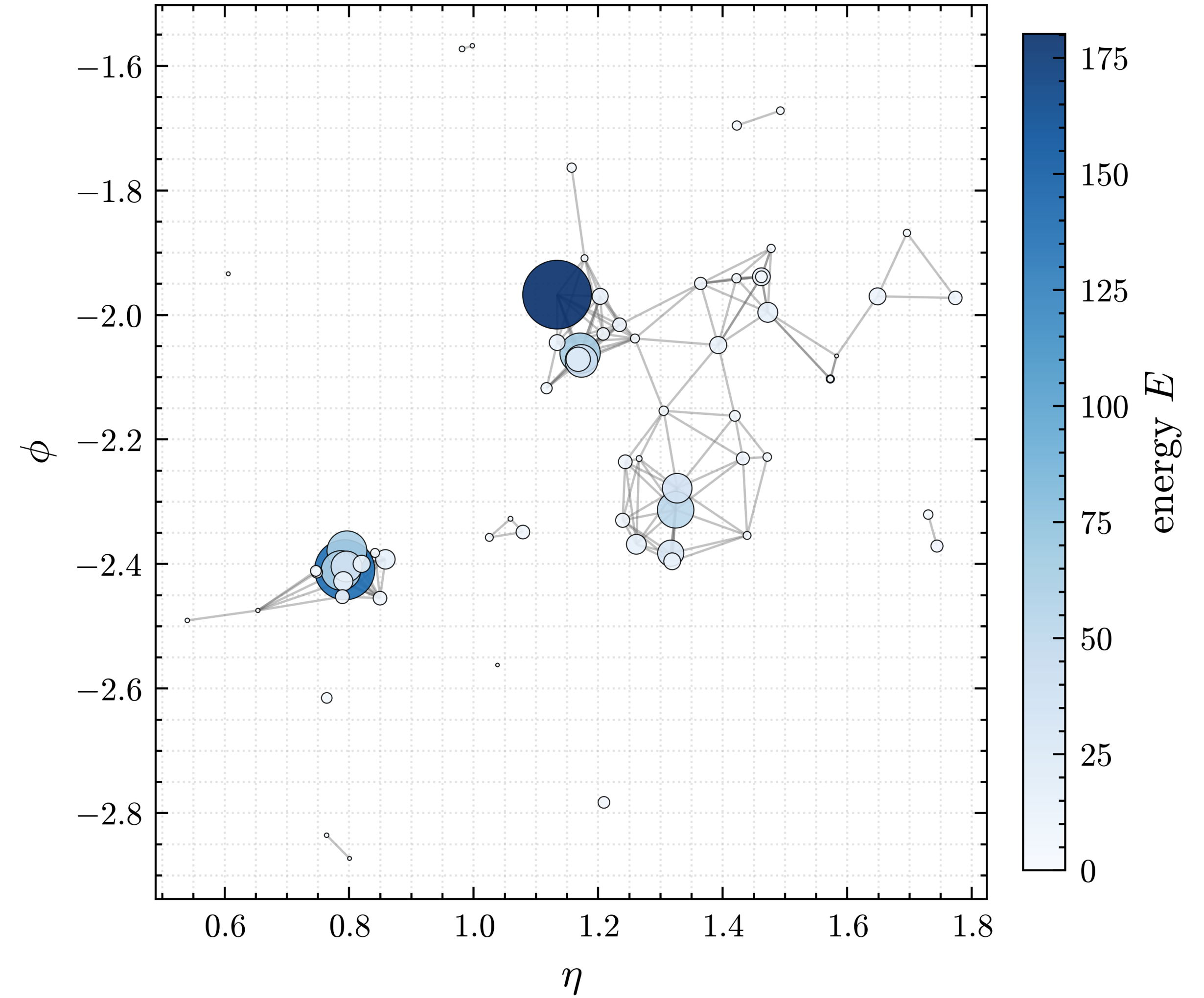




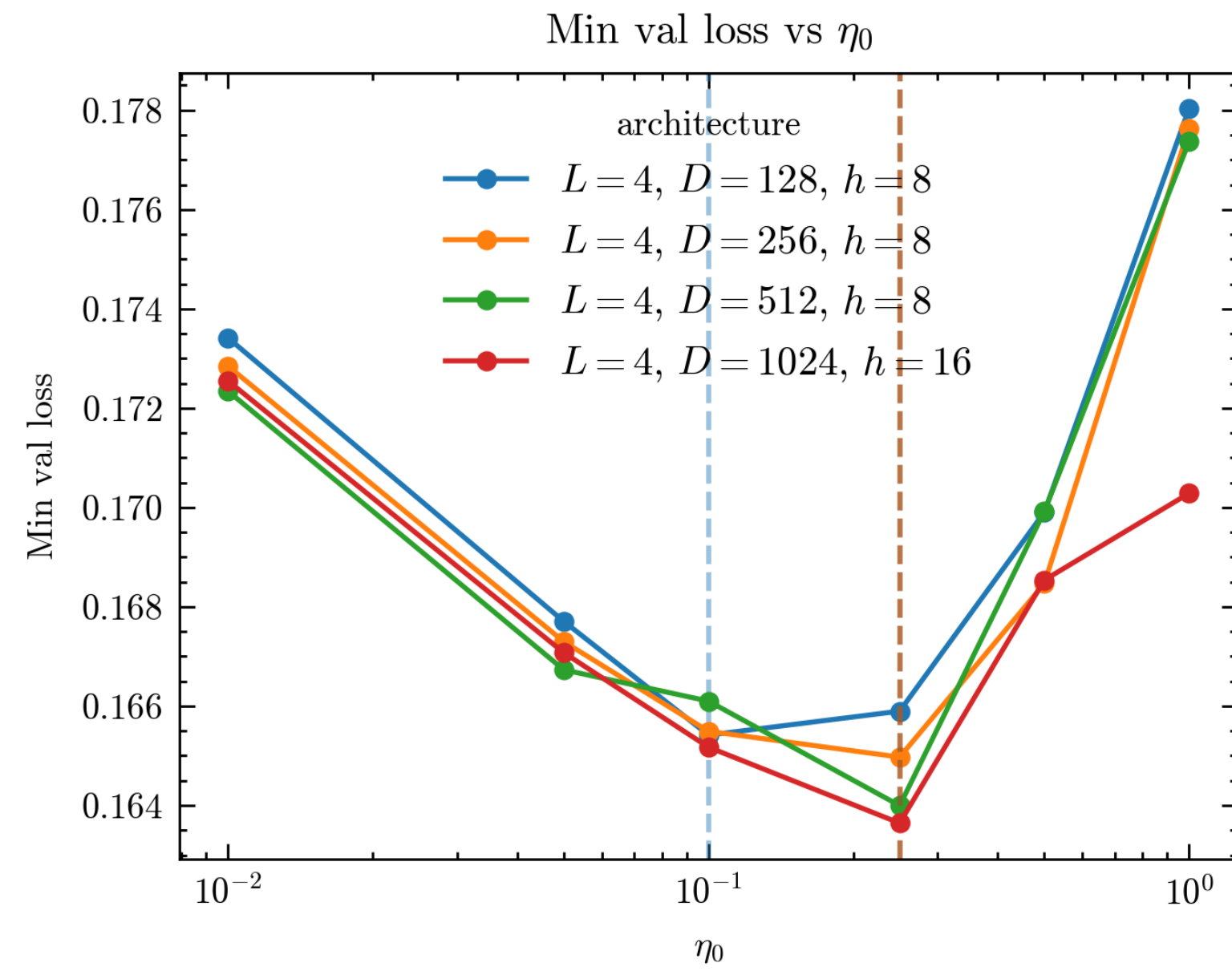
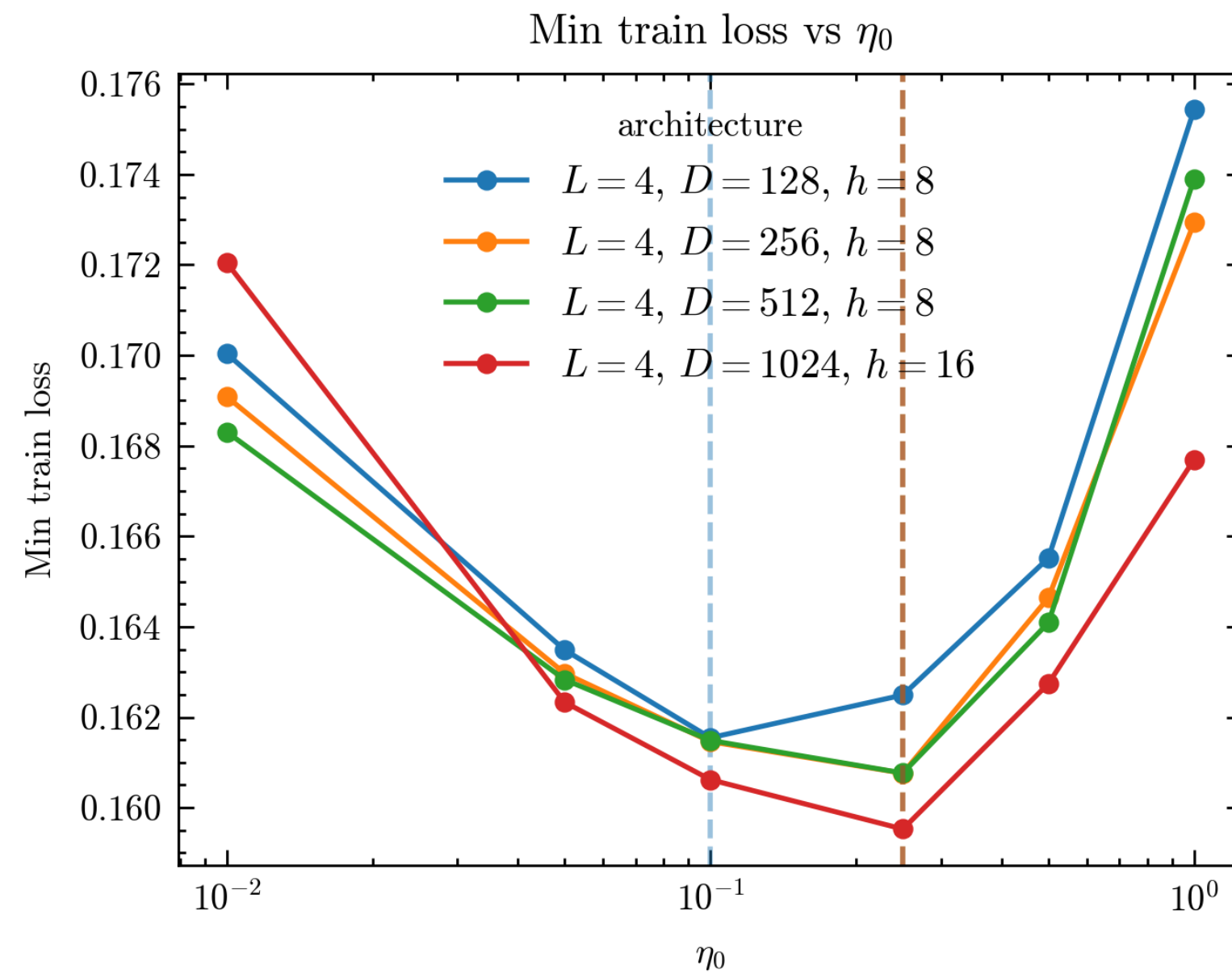
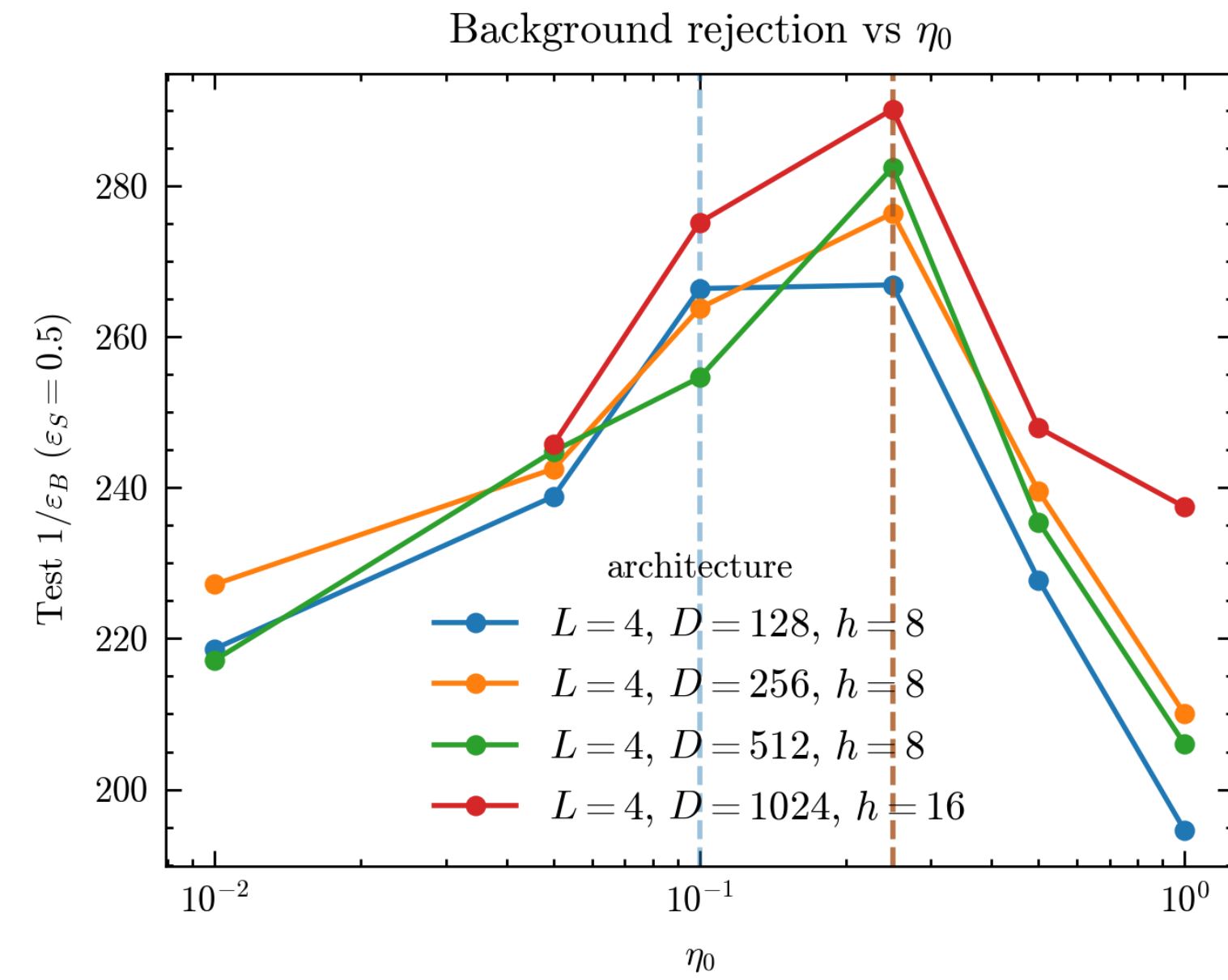
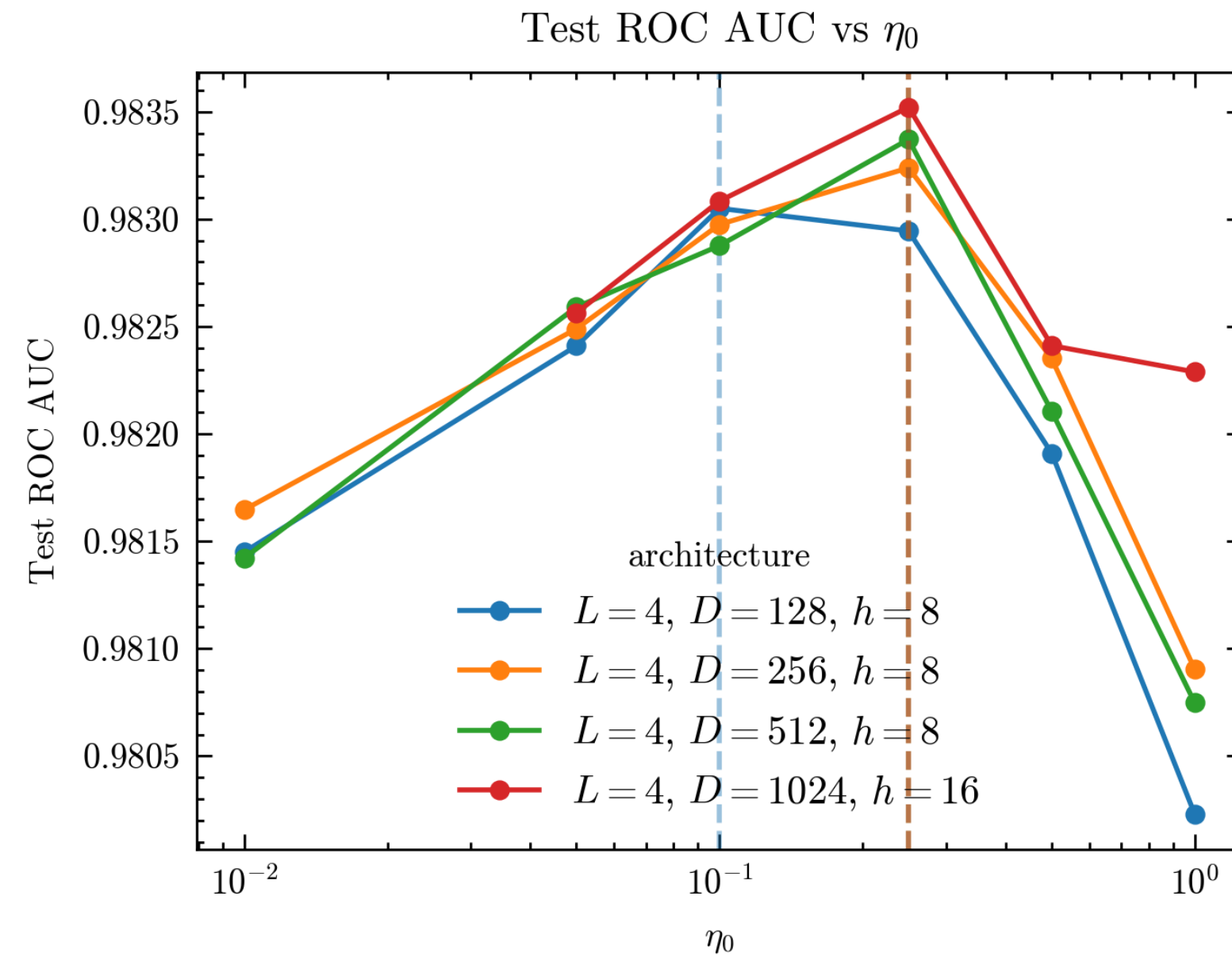
QCD jet 91179

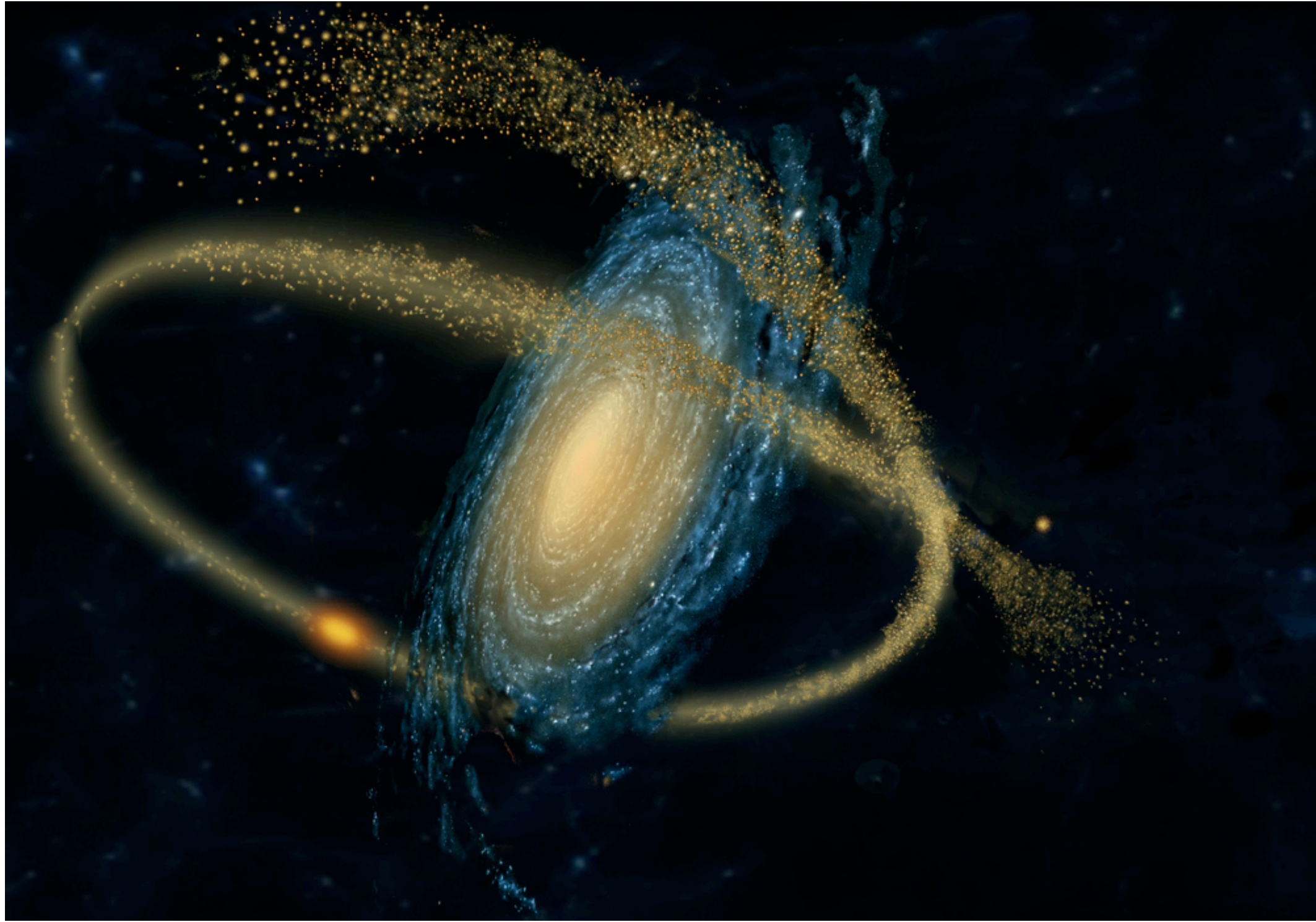


top jet 619179

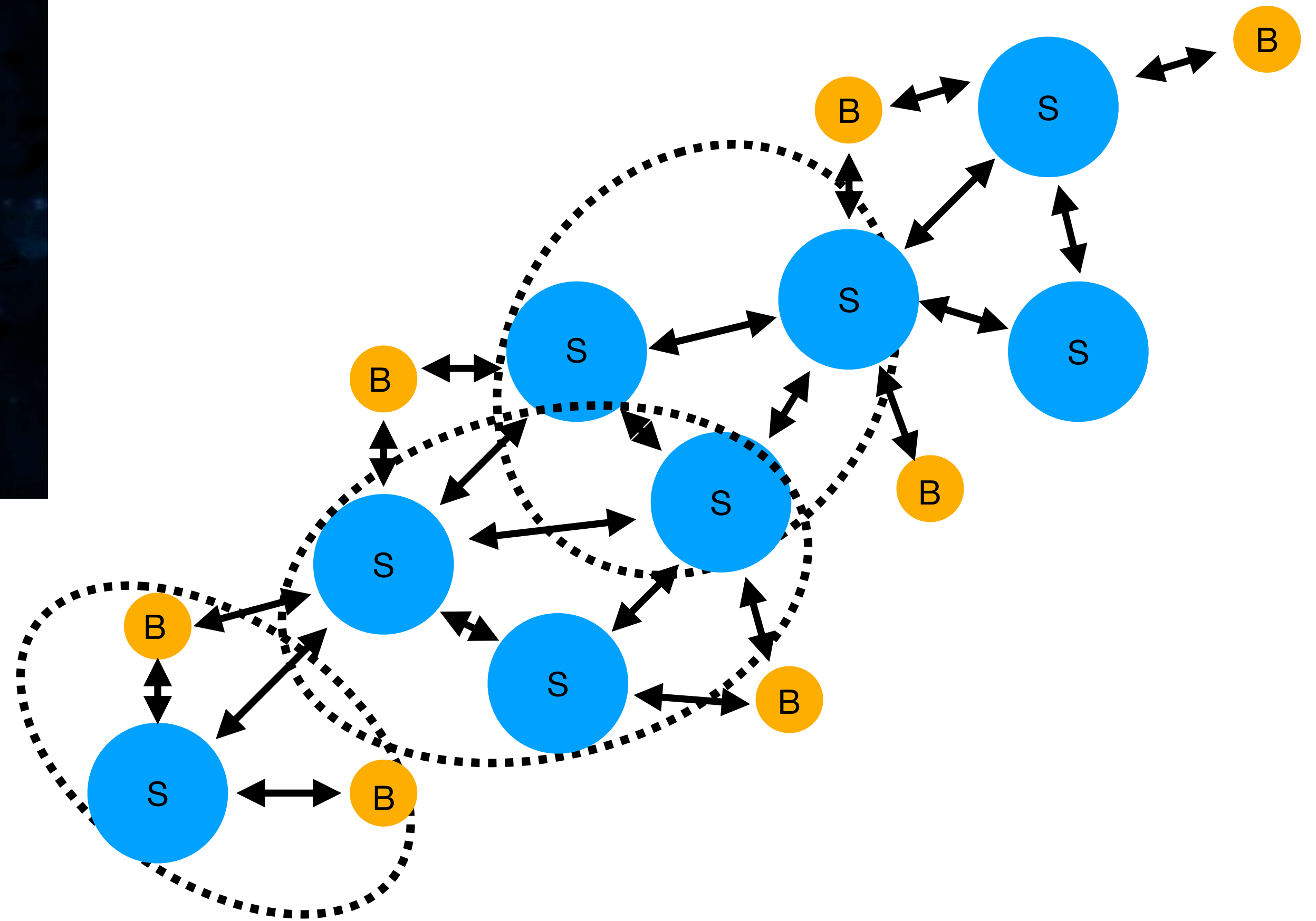


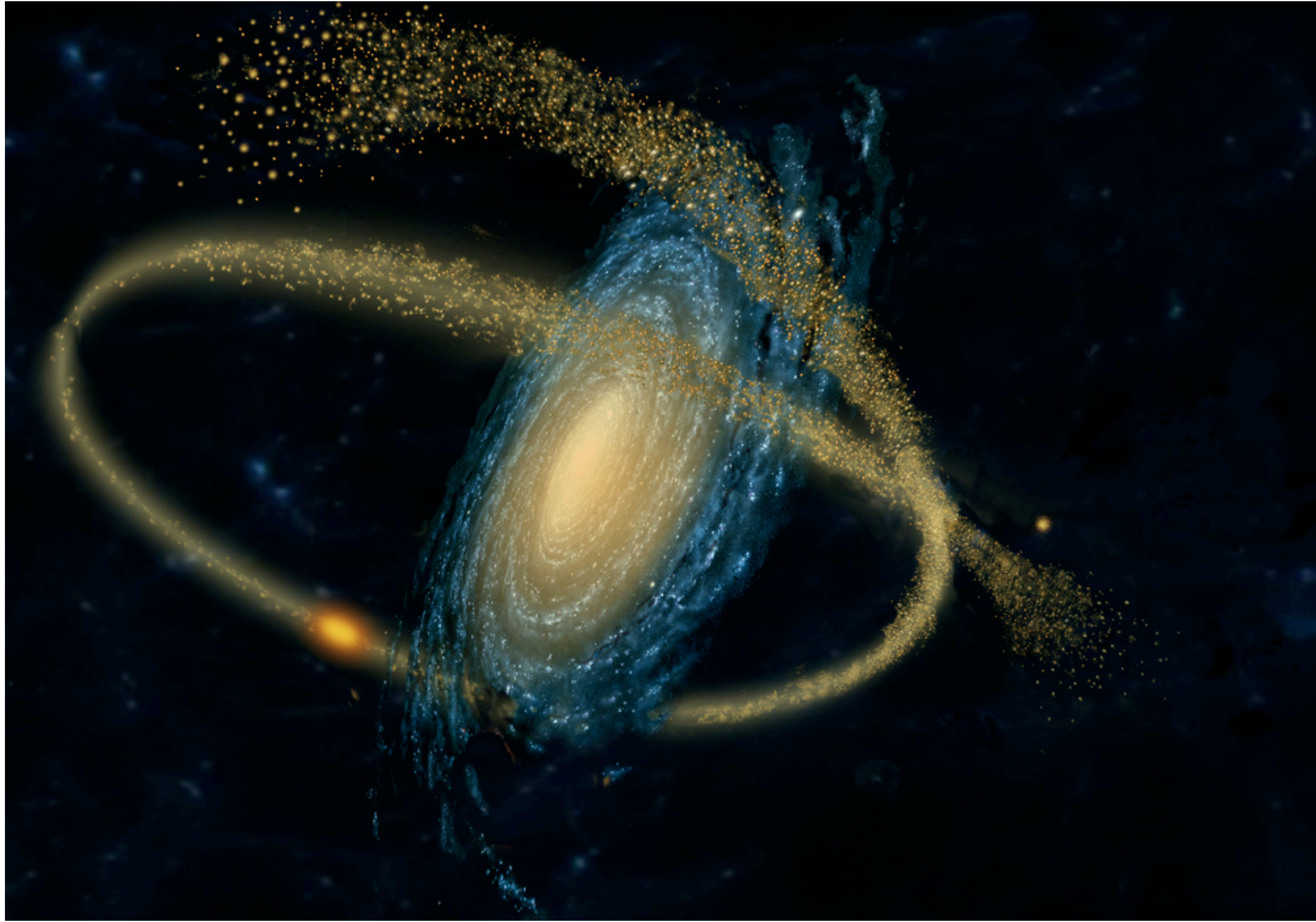
“Star graph” construction: 1 hyper-edge per
particle $\rightarrow \Delta R < 0.15$ neighbors



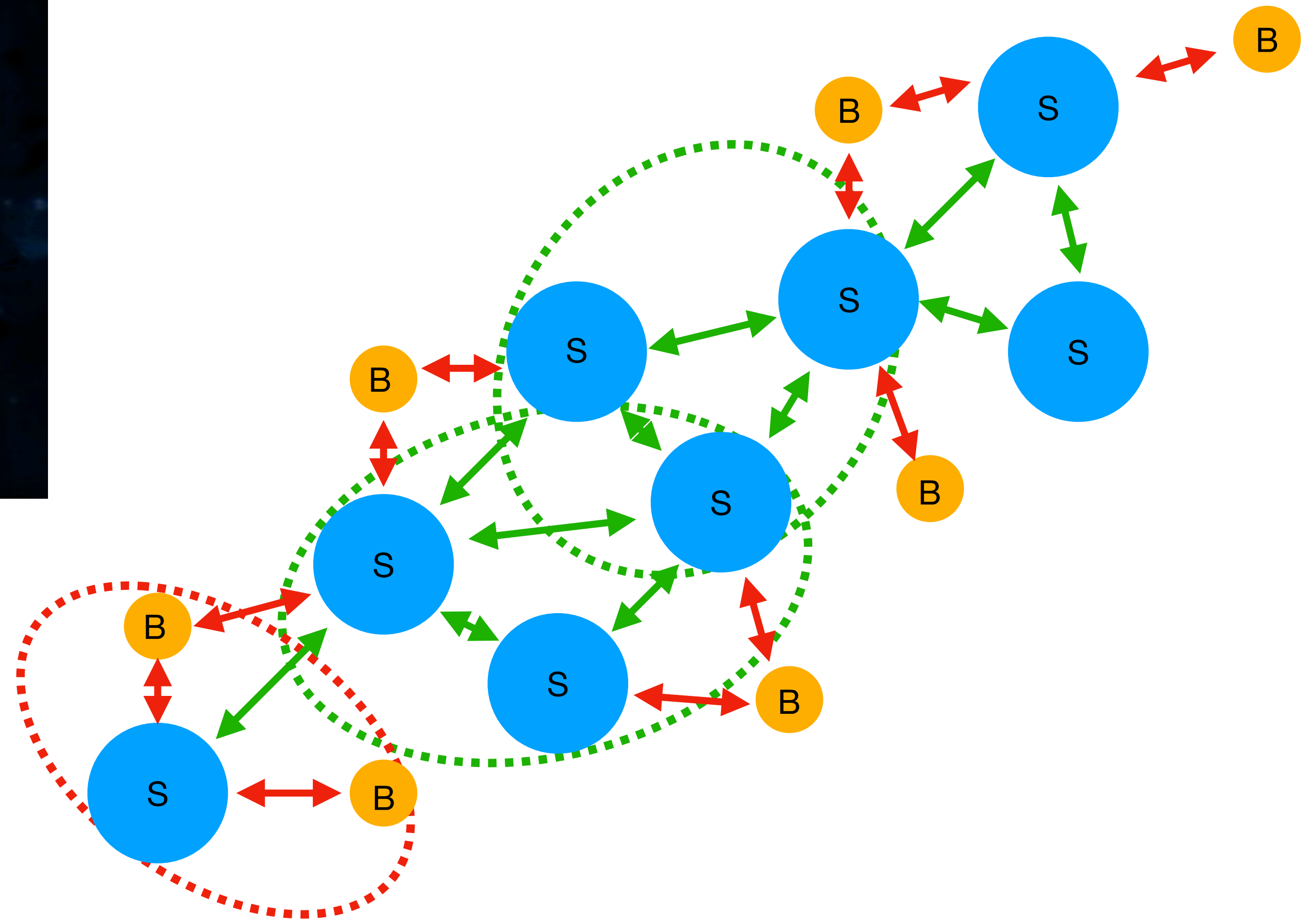


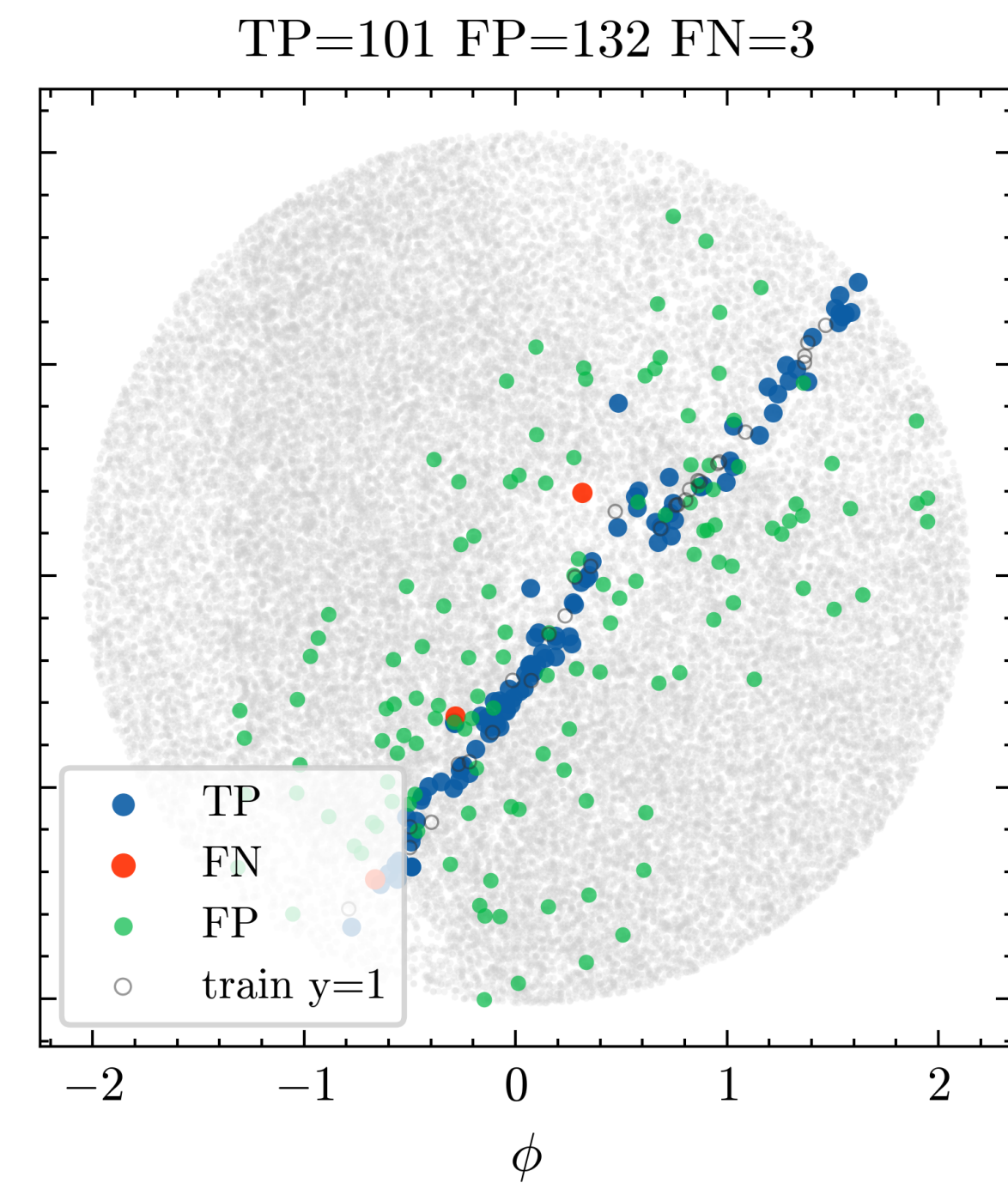
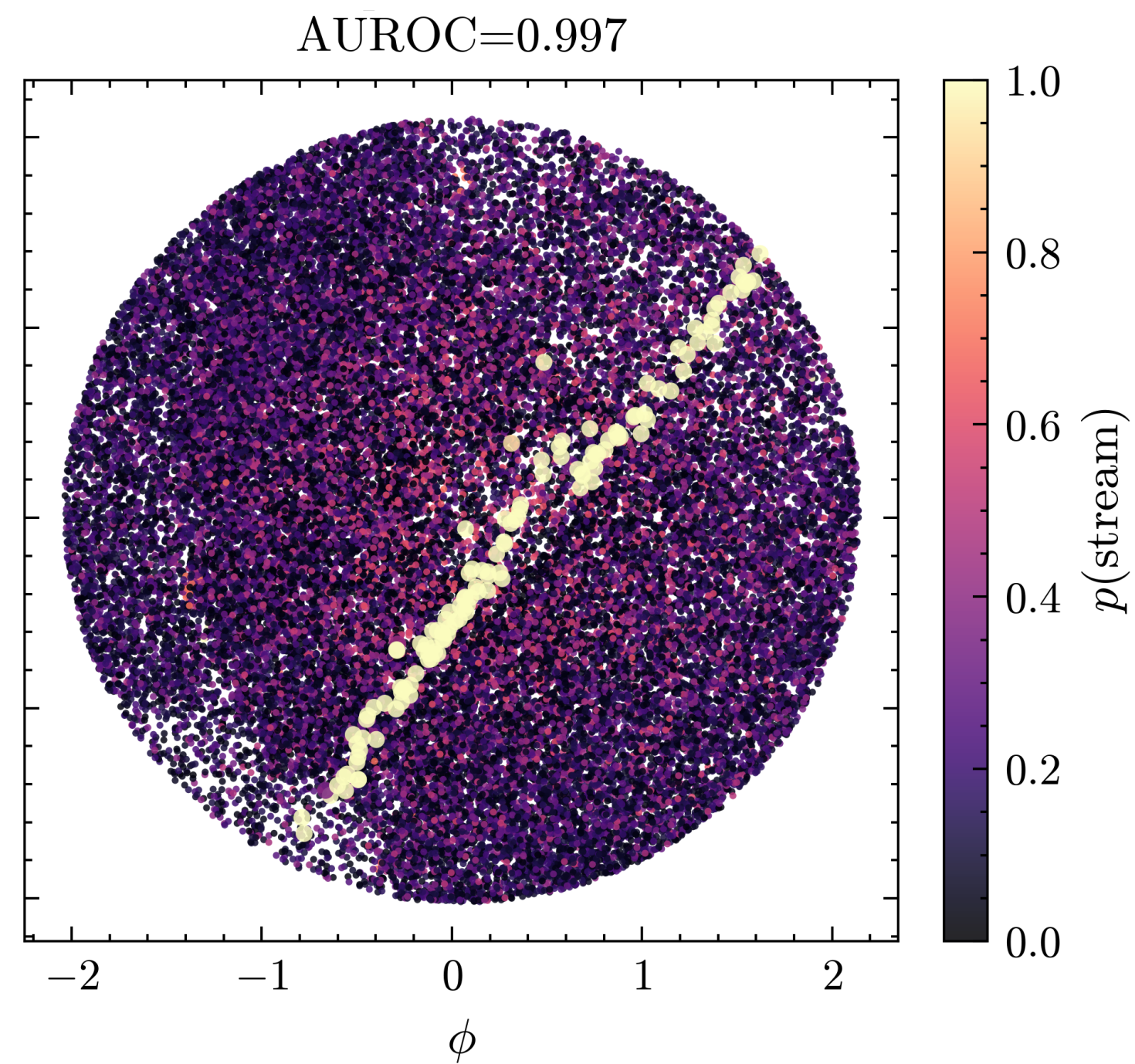
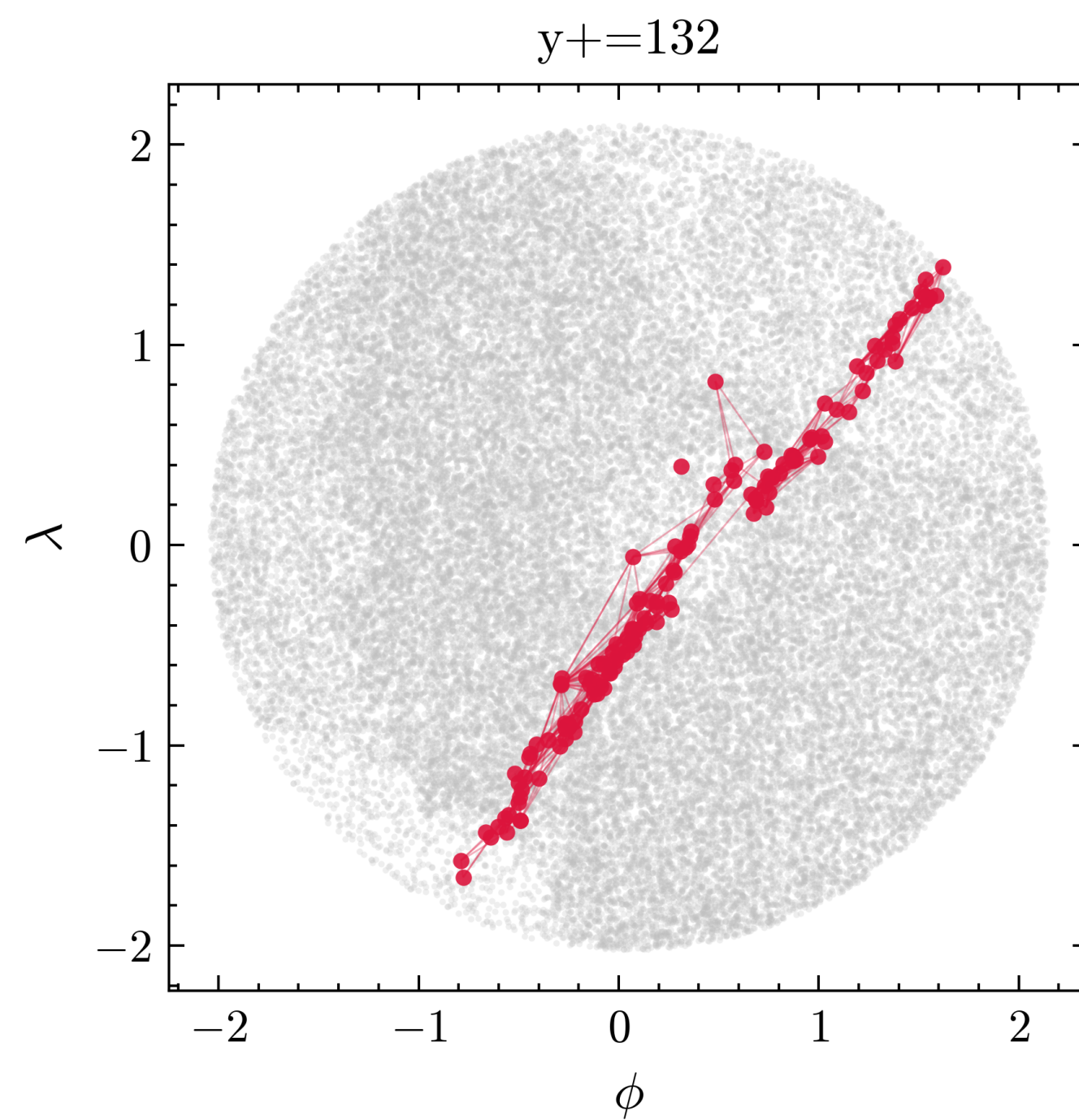
Stellar Streams





Stellar Streams





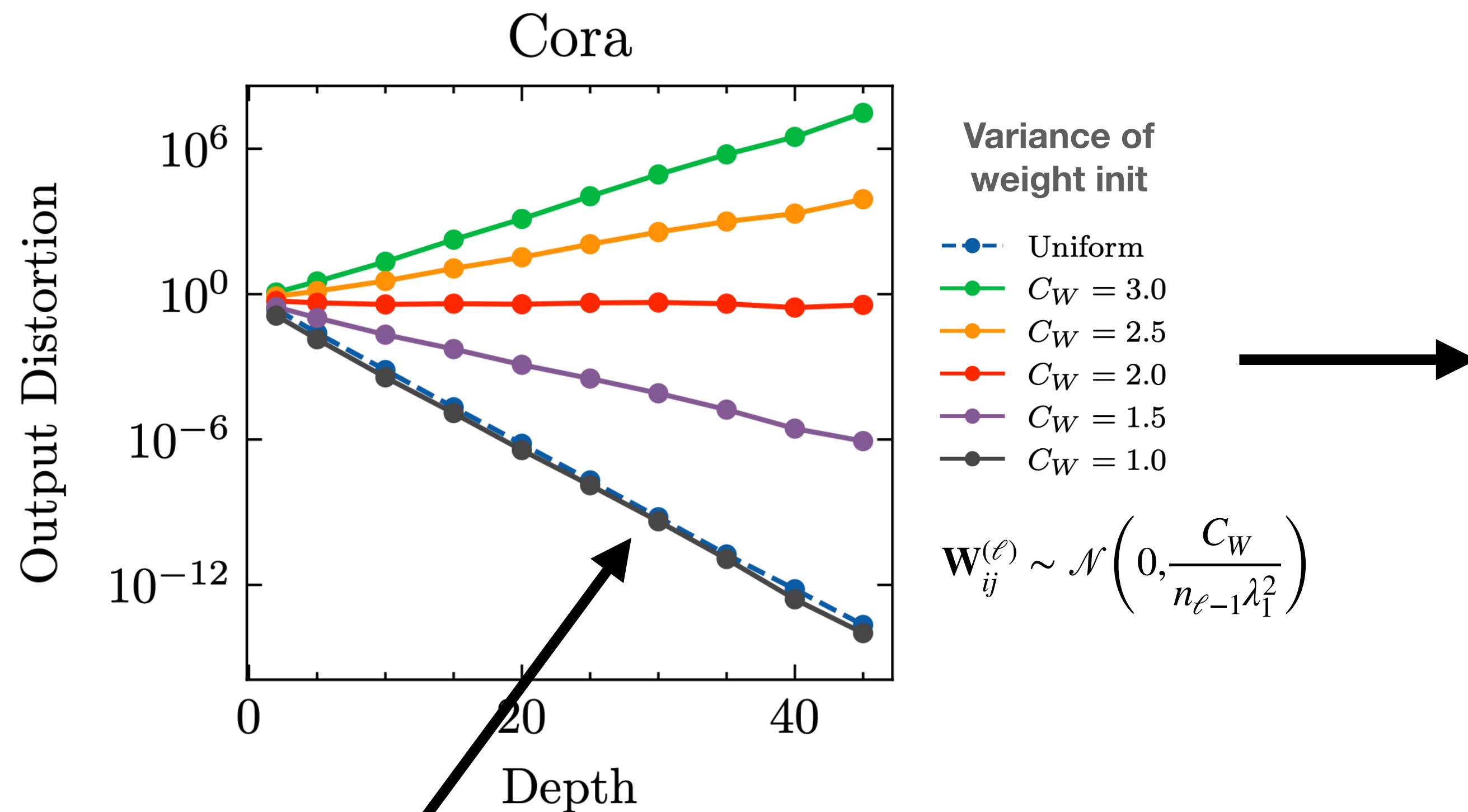
Go see Adri's poster!

Conclusions, Future Directions

- Scaling up GNNs is *highly nontrivial*
 - Many numerical issues, very few scaling laws as a result
 - This framework provides a *stable* and *scalable* approach to learning on graph data
- *Hyperparameter Transfer in Graph Neural Networks* paper
 - Recipe for scaling up GNNs and achieving hyperparameter transfer
- New work on hyperparameter transfer with edge features / attention
 - Application to the JetClass dataset (w/ Prof. Andre Frankenthal, UC Irvine)
 - Application to stellar streams (w/ Adri Dropulic, DARK Fellow @ NBI)

Backup

Numerical Stability in GNNs



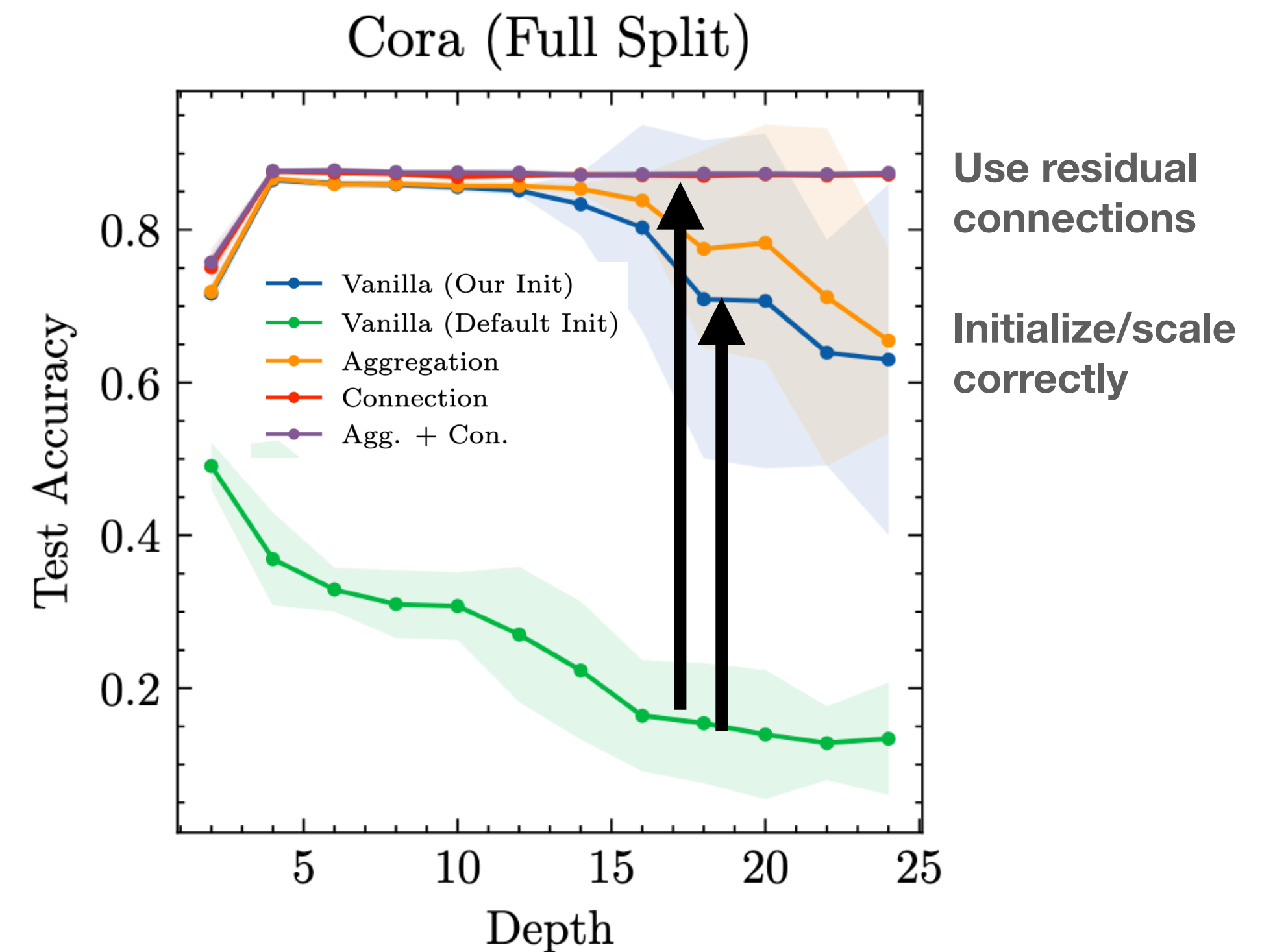
PyTorch Geometric
default init applied
to ReLU GCN

Principles for Initialization and Architecture Selection
in Graph Neural Networks with ReLU Activations

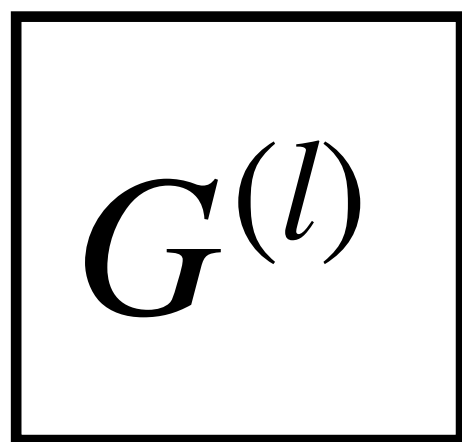
Gage DeZoor and Boris Hanin

<https://arxiv.org/pdf/2306.11668>

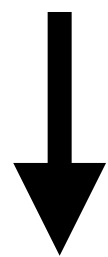
<https://epubs.siam.org/doi/abs/10.1137/23M1600621>



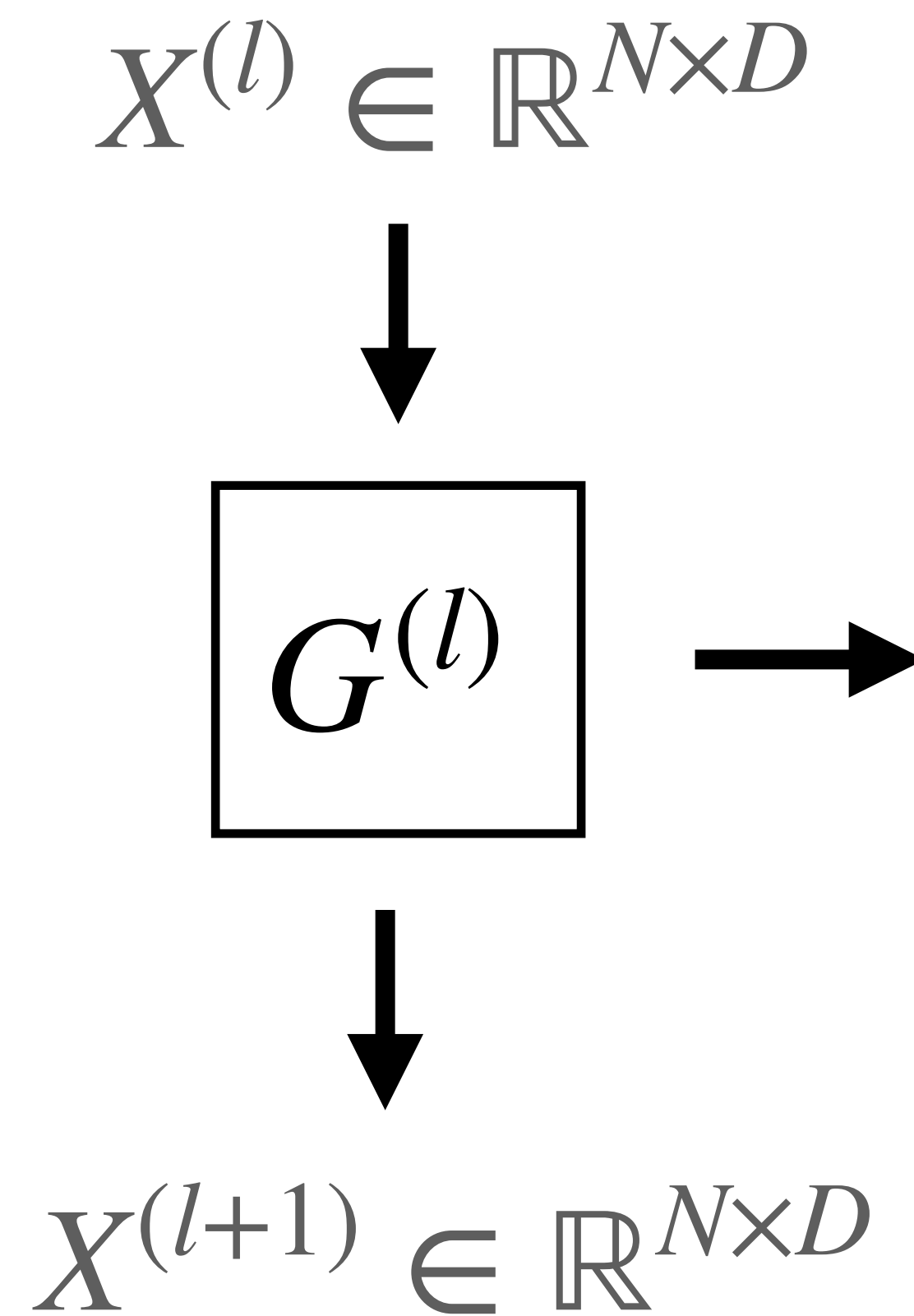
$$X^{(l)} \in \mathbb{R}^{N \times D}$$



Residual GNN Block



$$X^{(l+1)} \in \mathbb{R}^{N \times D}$$



Structural Updates

$$\tilde{X}^{(l)} = X^{(l)} + \frac{1}{L} \text{GNN}^{(l+1)}(X^{(l)}, \mathbf{A})$$

Self Updates

$$X^{(l+1)} = \tilde{X}^{(l)} + \frac{1}{L} \text{MLP}^{(l+1)}(\tilde{X}^{(l)})$$

Desiderata for Hyperparameter Transfer

- 1. Feature Stability:** features $\mathbf{X}_{u,i}^{(l)}$ should remain order-one with respect to scale — **initialize and scale the forward pass carefully**
- 2. Update Stability:** updates $\Delta \mathbf{X}_{u,i}^{(l)}$ induced by optimizer steps should remain stable and nontrivial with respect to scale — **parameterize learning rate with respect to *width* and *depth***
- 3. Complete Learning:** updates to features should not converge to their linearization around any given parameter — **use residual connections scaled by 1/depth**

Desiderata for Hyperparameter Transfer

1. Feature Stability: features $X_{u,i}^{(l)}$ should remain order-one with respect to scale

$$X' = \frac{1}{\sqrt{D}} X \underline{W}$$

Forward pass scaled
with respect to width

Entries of $\underline{W} \in \mathbb{R}^{D \times D}$
initialized from $N(0,1)$

Desiderata for Hyperparameter Transfer

2. Update Stability: updates $\Delta \mathbf{X}_{u,i}^{(l)}$ induced by optimizer steps should remain stable and nontrivial with respect to scale

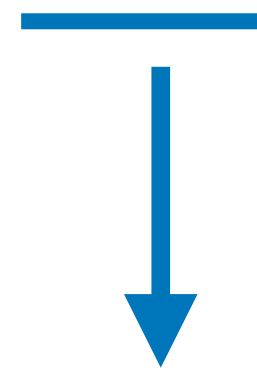
$$\Delta \mathbf{X}' = \frac{1}{\sqrt{D}} \left(\Delta \mathbf{X} \mathbf{W} + \mathbf{X} \frac{\Delta \mathbf{W}}{\quad} \right)$$


Weight updates are proportional to the learning rate;
optimizer-specific learning rate settings

Desiderata for Hyperparameter Transfer

3. Complete Learning: updates to features should not converge to their linearization around any given parameter

$$\mathbf{X}^{(l+1)} = \mathbf{X}^{(l)} + \frac{1}{L} \text{GNN}(\mathbf{X}^{(l)}, \mathbf{A})$$



Scale residual connections by 1/depth

Desiderata for Hyperparameter Transfer

3. Complete Learning: updates to features should not converge to their linearization around any given parameter

$$\mathbf{h}^{\text{lin},\theta}(\boldsymbol{\theta}, \boldsymbol{\theta}_0) = \mathbf{h}(\boldsymbol{\theta}_0) + \langle \nabla_{\boldsymbol{\theta}} \mathbf{h}(\boldsymbol{\theta}) |_{\boldsymbol{\theta}_0}, \boldsymbol{\theta} - \boldsymbol{\theta}_0 \rangle.$$

We say $\mathbf{h}$ is **linear** in $\boldsymbol{\theta}$ if, $\mathbf{h} = \mathbf{h}^{\text{lin},\theta}$.

Definition. We say a layer $\mathbf{h}^\ell$ is **lazy** with respect to a subset of parameters $\theta \subset \{\theta_j\}_{j < \ell}$ if at finite depth and width, $\mathbf{h}^\ell$ is not linear in θ and the change $\Delta_\theta \mathbf{h}^\ell$ at initialization from updating only θ (i.e. replacing $\theta \mapsto \theta + \Delta\theta$) is asymptotically the same as the change to the linearization of $\mathbf{h}$:

$$\frac{|\Delta_\theta \mathbf{h}^\ell - \Delta_\theta \mathbf{h}_\ell^{\text{lin},\theta}|}{|\Delta_\theta \mathbf{h}_\ell^{\text{lin},\theta}|} = o(1), \quad \text{as } N, L \rightarrow \infty. \quad (4)$$

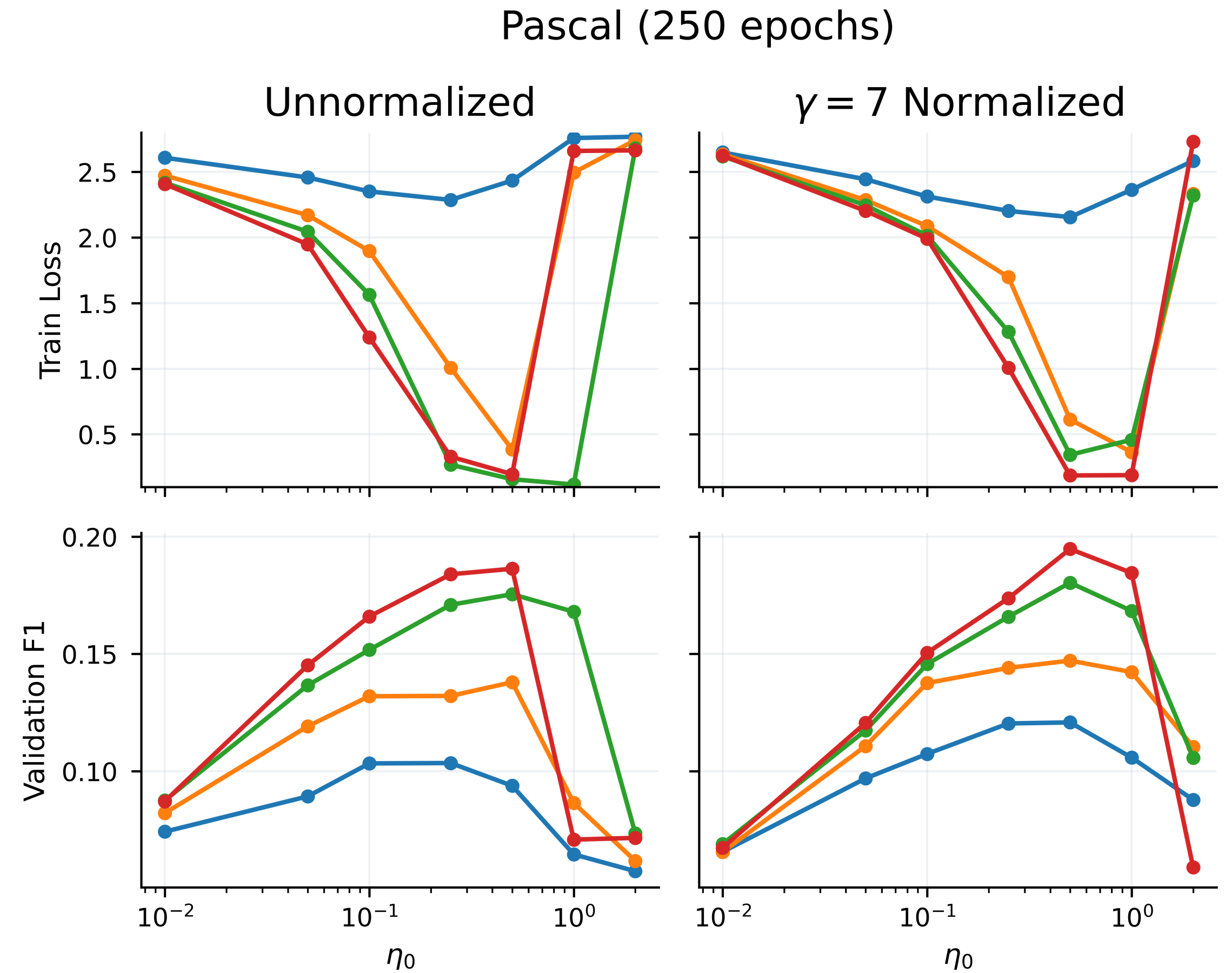
Desideratum 3 (Complete Feature Learning). *The network parameterization satisfies complete feature learning, i.e. neither the hidden layers $\{\mathbf{h}^\ell\}_{\ell \in [L]}$ nor the model output f are lazy with respect to any subset of model parameters.*

$$\mathbf{X}' = \frac{1}{\sqrt{D}} \frac{1}{\underline{\gamma}} \mathbf{A} \sigma(\mathbf{X}) \mathbf{W}$$

$$\mathbf{x}_u = \frac{1}{\sqrt{D}} \frac{1}{\underline{\gamma}} \sum_{v \in \mathcal{N}(u)} \mathbf{W} \sigma(\mathbf{x}_v)$$



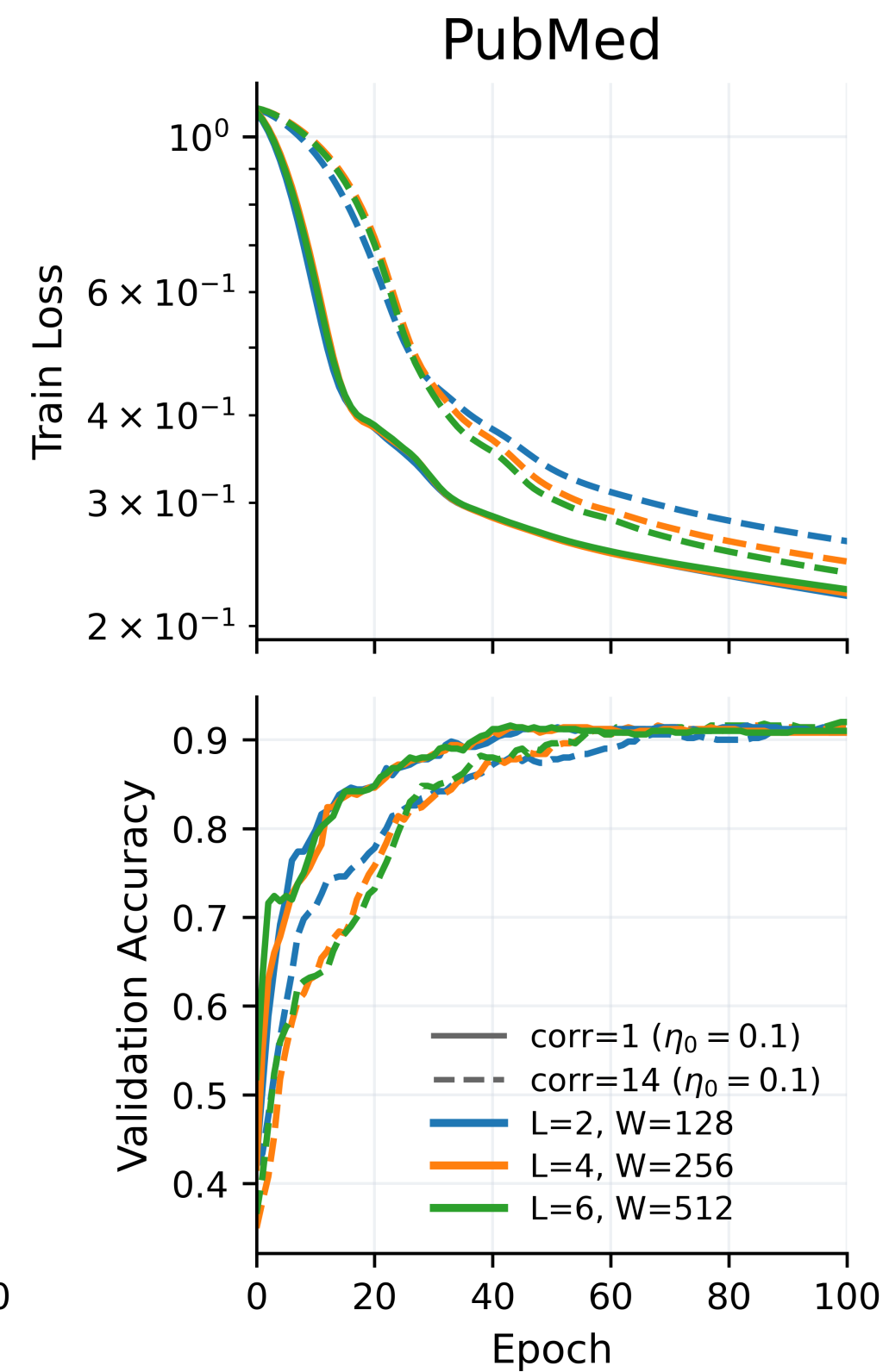
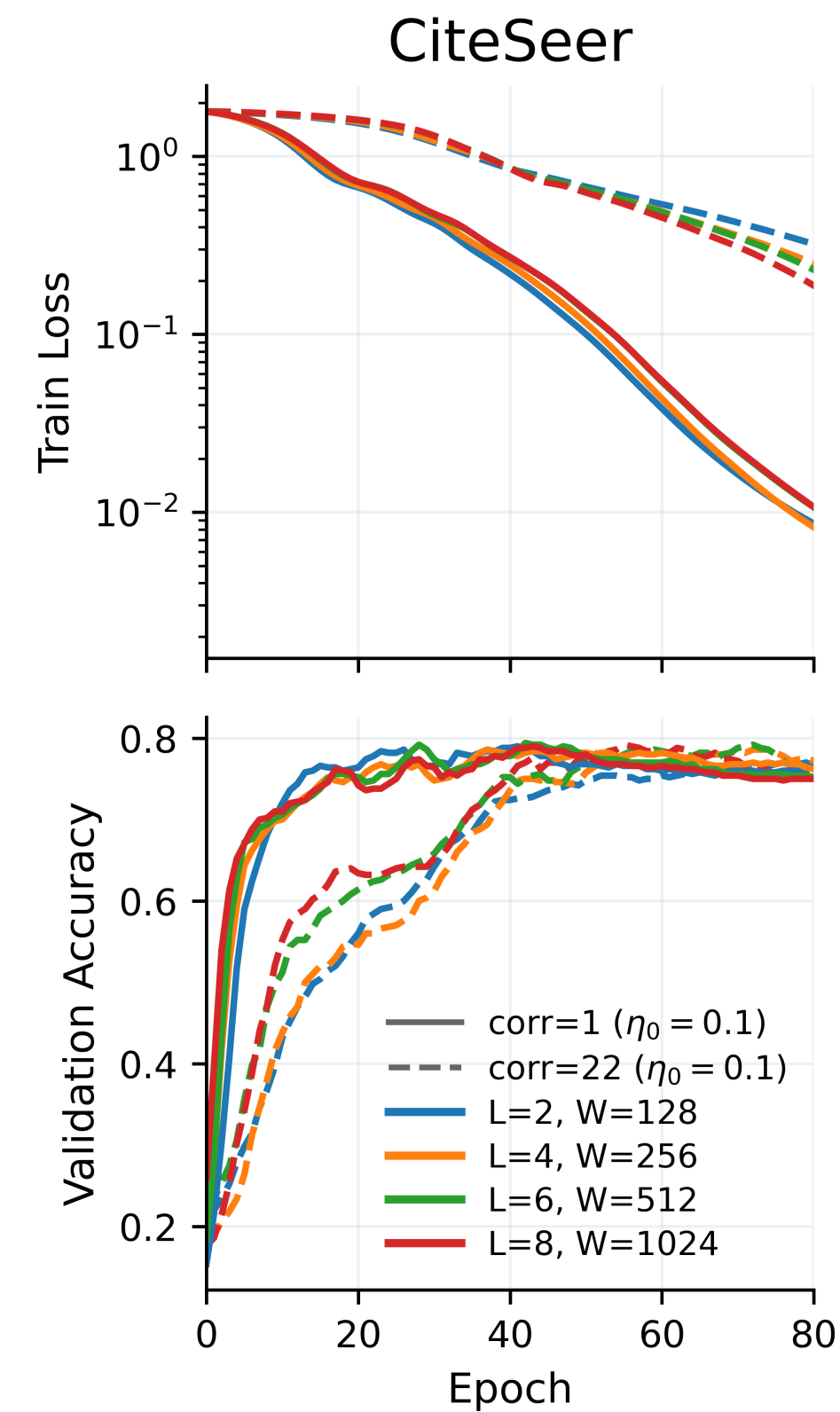
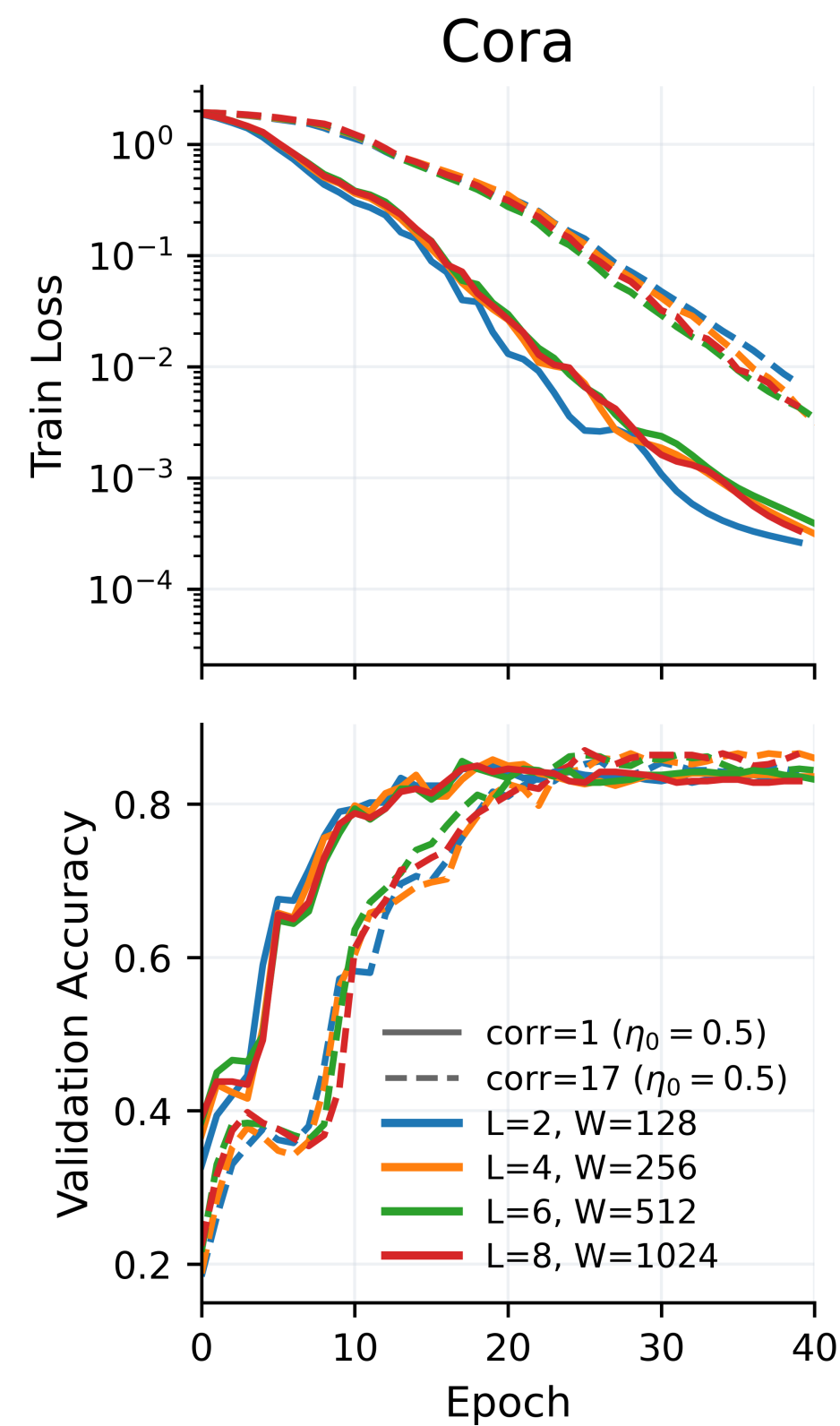
normalize scale of message passing



$$\eta^{(0)} = \eta_0 DL \times C_{ab}$$

$$C_{ab} = \frac{n_0 \sqrt{N_b}}{M_{ab}}$$

$$M_{ab} = ||\bar{\mathbf{x}}_a \mathbf{X}_b^T||_2$$



$$\theta_{t+1} \leftarrow (1 - \lambda\eta)\theta_t - \eta\hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$$

$$\theta_t = \tau^{-1} \sum_{t'=1}^t e^{-(t-t')/\tau} q_{t'} \text{ with } \tau := 1/\eta\lambda$$

