When do Neural ODEs Generalize on Complex Networks?

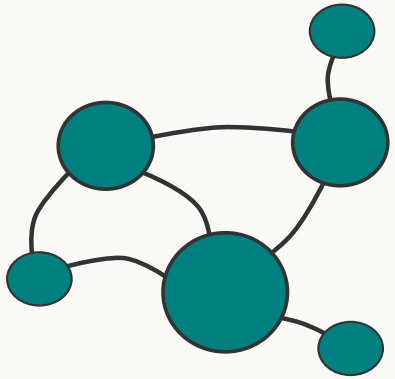
Moritz Laber, Tina Eliassi-Rad, Brennan Klein



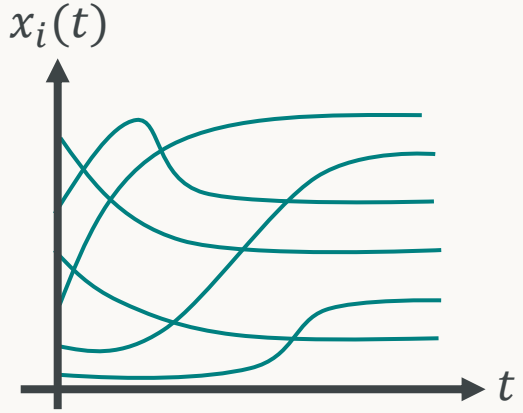
✉ **laber.m@northeastern.edu**

Network Science Student Research Symposium, Boston, US, 2026-02-19

Dynamics from Data



graph A

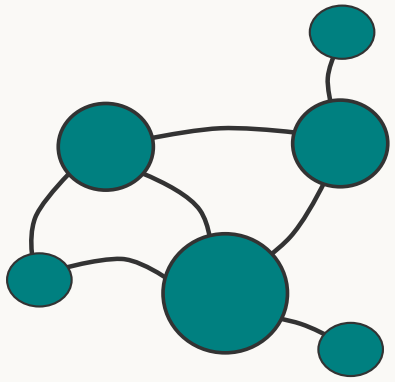


node-wise timeseries $x_i(t)$

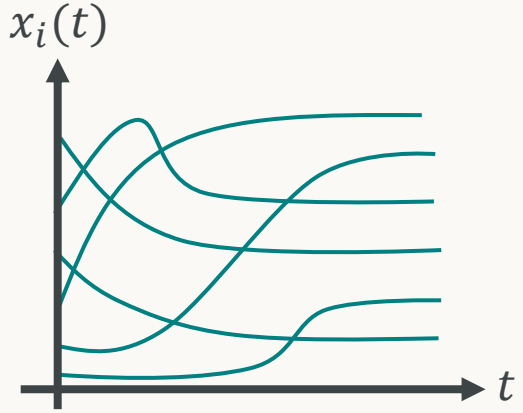
Dynamics from Data

Classical ODEs:
Handcraft differential equations using domain knowledge.

$$\frac{dx_i}{dt} = f(x_1, \dots, x_n)$$



graph A

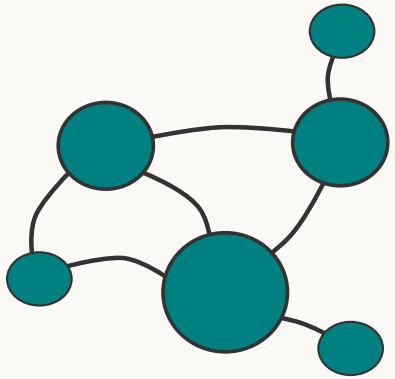


node-wise timeseries $x_i(t)$

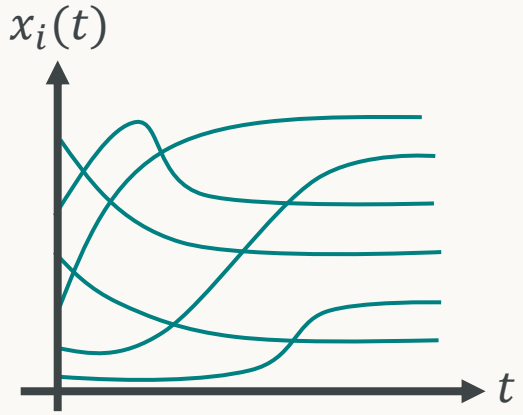
Dynamics from Data

Classical ODEs:
Handcraft differential equations using domain knowledge.

$$\frac{dx_i}{dt} = f(x_1, \dots, x_n)$$



graph A



node-wise timeseries $x_i(t)$

Neural ODEs [1,2,3]**:**
Learn differential equations from timeseries data.

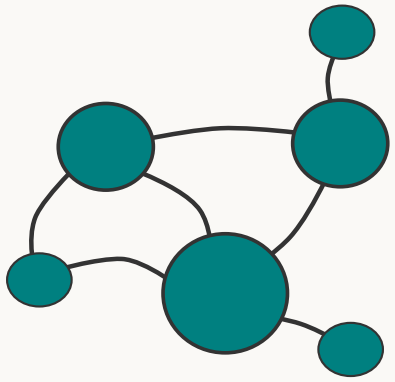
$$\frac{dx_i}{dt} = f_\omega(x_1, \dots, x_n)$$

[1] *Chen et al.* NeurIPS (2018), [2] *Poli et al.* AAAI (2022), [3] *Kidger* PhD Thesis (2022)

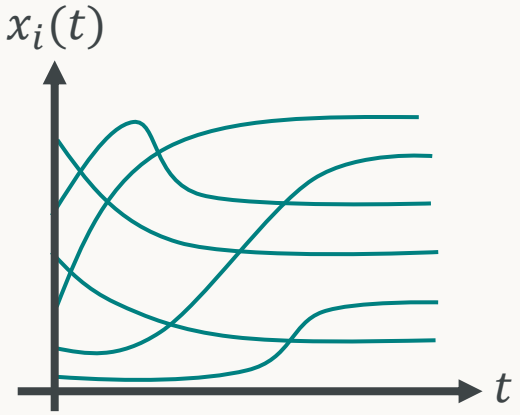
Dynamics from Data

Classical ODEs:
Handcraft differential equations using domain knowledge.

$$\frac{dx_i}{dt} = f(x_1, \dots, x_n)$$



graph A



node-wise timeseries $x_i(t)$

Neural ODEs [1,2,3]**:**
Learn differential equations from timeseries data.

$$\frac{dx_i}{dt} = f_\omega(x_1, \dots, x_n)$$

Questions: How do the performance and robustness of neural ODEs depend on the network? [4]

[1] *Chen et al.* NeurIPS (2018), [2] *Poli et al.* AAAI (2022), [3] *Kidger* PhD Thesis (2022)

[4] *Vasiliauskaite & Antulov-Fantulin* Comms Phys. 7 1 (2024)

Contributions

We train neural ODEs on **different dynamical systems** and **graphs from the $\mathbb{S}^1$ model** with different structural characteristics

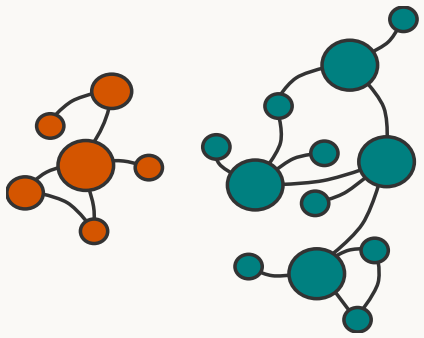
Contributions

We train neural ODEs on **different dynamical systems** and **graphs from the $\mathbb{S}^1$ model** with different structural characteristics

We evaluate their **performance and robustness** in terms of four criteria:

Generalization:

Graph Size



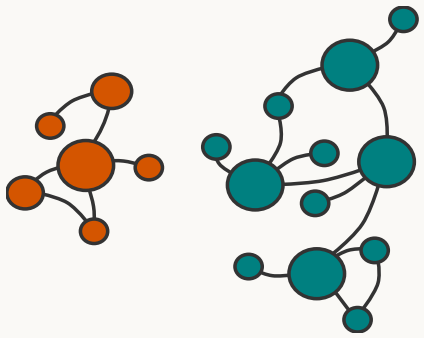
$n^{\text{train}} \ll n^{\text{test}}$

Contributions

We train neural ODEs on **different dynamical systems** and **graphs from the $\mathbb{S}^1$ model** with different structural characteristics

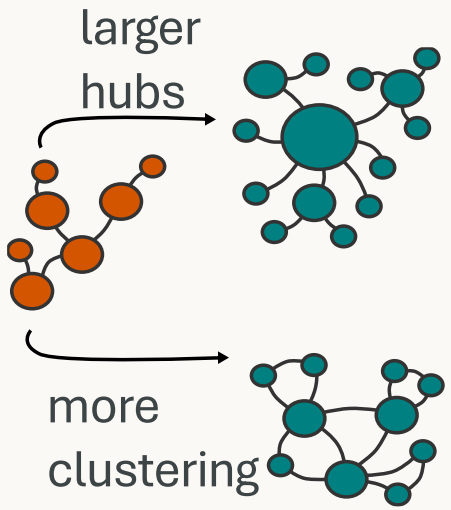
We evaluate their **performance and robustness** in terms of four criteria:

Generalization: Graph Size



$n^{\text{train}} \ll n^{\text{test}}$

Generalization: Graph Properties

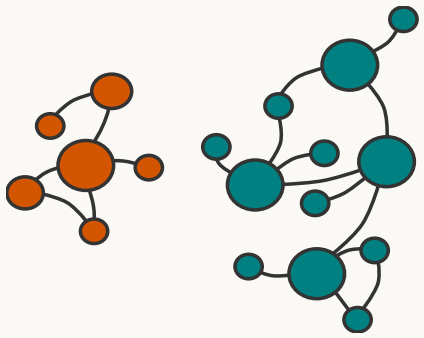


Contributions

We train neural ODEs on **different dynamical systems** and **graphs from the $\mathbb{S}^1$ model** with different structural characteristics

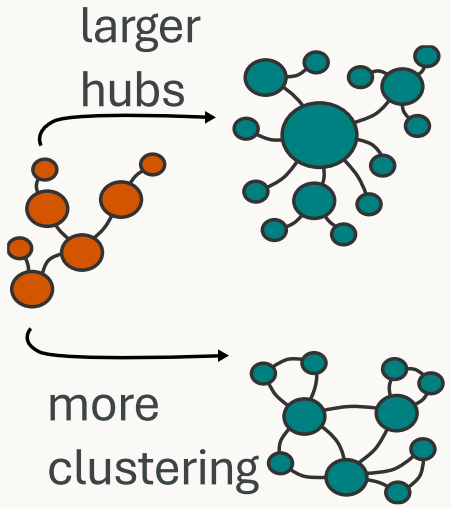
We evaluate their **performance and robustness** in terms of four criteria:

Generalization: Graph Size

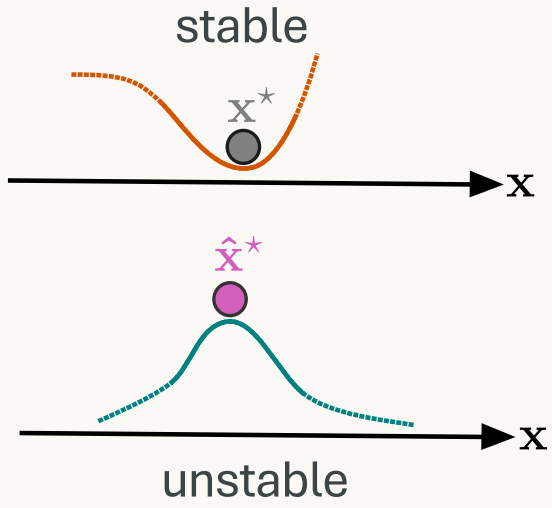


$n^{\text{train}} \ll n^{\text{test}}$

Generalization: Graph Properties



Robustness: Local Stability

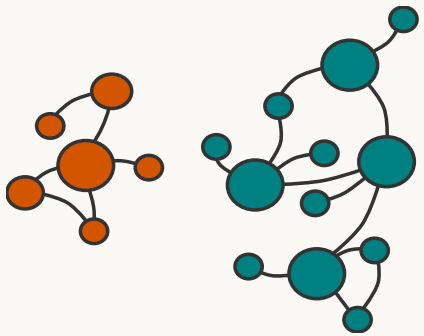


Contributions

We train neural ODEs on **different dynamical systems** and **graphs from the $\mathbb{S}^1$ model** with different structural characteristics

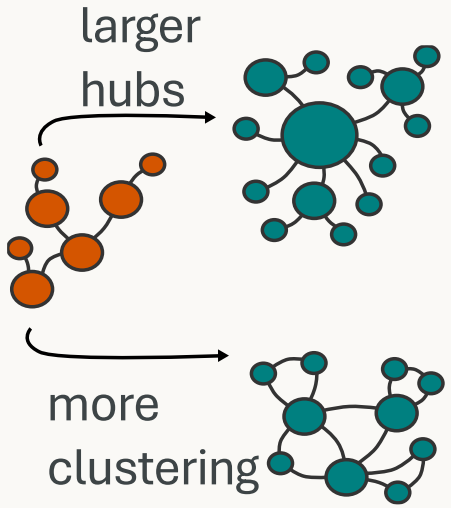
We evaluate their **performance and robustness** in terms of four criteria:

Generalization: Graph Size

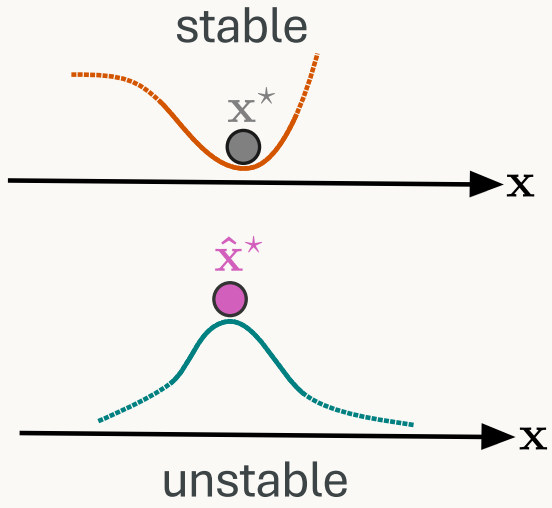


$n^{\text{train}} \ll n^{\text{test}}$

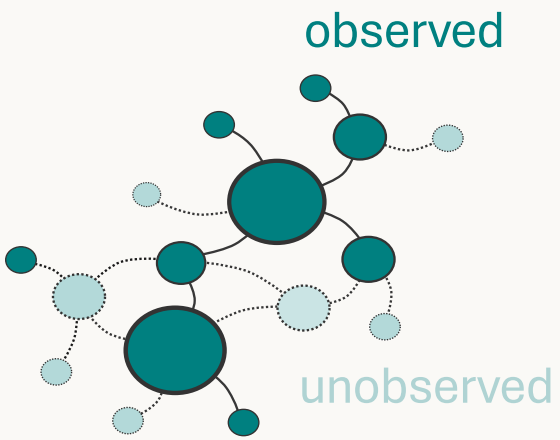
Generalization: Graph Properties



Robustness: Local Stability



Robustness: Missing Nodes

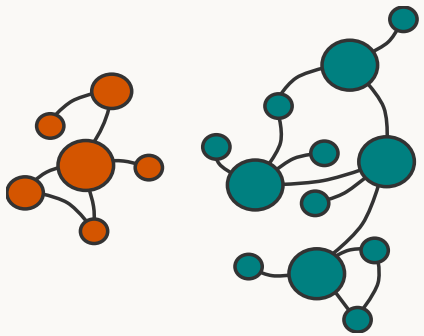


Contributions

We train neural ODEs on **different dynamical systems** and **graphs from the $\mathbb{S}^1$ model** with different structural characteristics

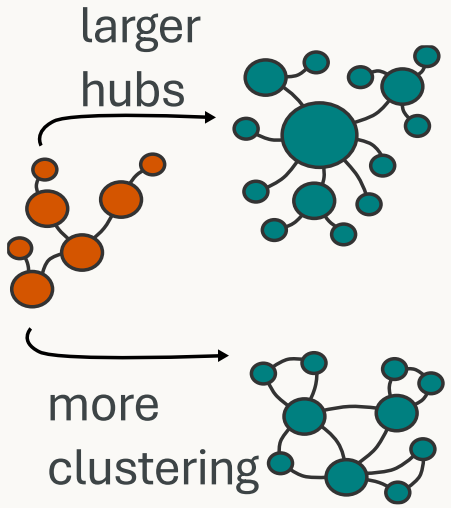
We evaluate their **performance and robustness** in terms of four criteria:

Generalization: Graph Size

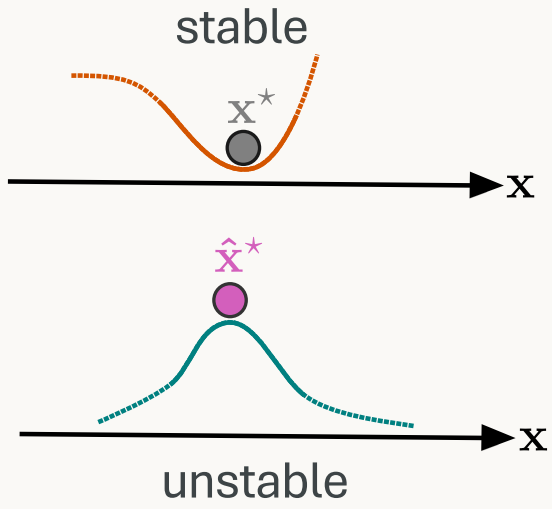


$n^{\text{train}} \ll n^{\text{test}}$

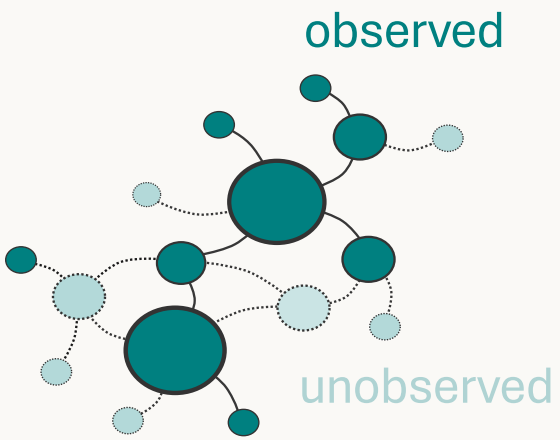
Generalization: Graph Properties



Robustness: Local Stability



Robustness: Missing Nodes



Background: Neural ODEs

Neural ODEs are ordinary differential equations in which the vector field is parameterized by a neural network.

$$\frac{dx_i}{dt} = f_\omega(t, x_1(t), \dots, x_n(t)) \qquad x_i(t=0) = \tilde{x}_i$$

Background: Neural ODEs

Neural ODEs are ordinary differential equations in which the vector field is parameterized by a neural network.

$$\frac{dx_i}{dt} = f_\omega(t, x_1(t), \dots, x_n(t)) \qquad x_i(t=0) = \tilde{x}_i$$

Applications:

- Continuum limit of architectures with residual connections [1].

[1] *Chen et al.* NeurIPS (2018)

Background: Neural ODEs

Neural ODEs are ordinary differential equations in which the vector field is parameterized by a neural network.

$$\frac{dx_i}{dt} = f_\omega(t, x_1(t), \dots, x_n(t)) \qquad x_i(t=0) = \tilde{x}_i$$

Applications:

- Continuum limit of architectures with residual connections [1].
- Part of generative models, i.e., continuous normalizing flows [1,2].

[1] *Chen et al.* NeurIPS (2018) [2] *Grathwohl et al.* ICLR (2019)

Background: Neural ODEs

Neural ODEs are ordinary differential equations in which the vector field is parameterized by a neural network.

$$\frac{dx_i}{dt} = f_\omega(t, x_1(t), \ldots, x_n(t)) \qquad x_i(t=0) = \tilde{x}_i$$

Applications:

- Continuum limit of architectures with residual connections [1].
- Part of generative models, i.e., continuous normalizing flows [1,2].
- Modeling of dynamical systems from data [3,4].

[1] *Chen et al.* NeurIPS (2018) [2] *Grathwohl et al.* ICLR (2019) [3] *Rackauckas et al.* arXiv:2001.04385 (2020) [4] *Gao et al.* Nat.Comms. 15 (1) 6029 (2024)

Background: Neural ODEs

Neural ODEs are ordinary differential equations in which the vector field is parameterized by a neural network.

$$\frac{dx_i}{dt} = f_\omega(t, x_1(t), \ldots, x_n(t)) \qquad x_i(t=0) = \tilde{x}_i$$

Applications:

- Continuum limit of architectures with residual connections [1].
- Part of generative models, i.e., continuous normalizing flows [1,2].
- Modeling of dynamical systems from data [3,4].

[1] *Chen et al.* NeurIPS (2018) [2] *Grathwohl et al.* ICLR (2019) [3] *Rackauckas et al.* arXiv:2001.04385 (2020) [4] *Gao et al.* Nat.Comms. 15 (1) 6029 (2024)

Background: Training Neural ODEs

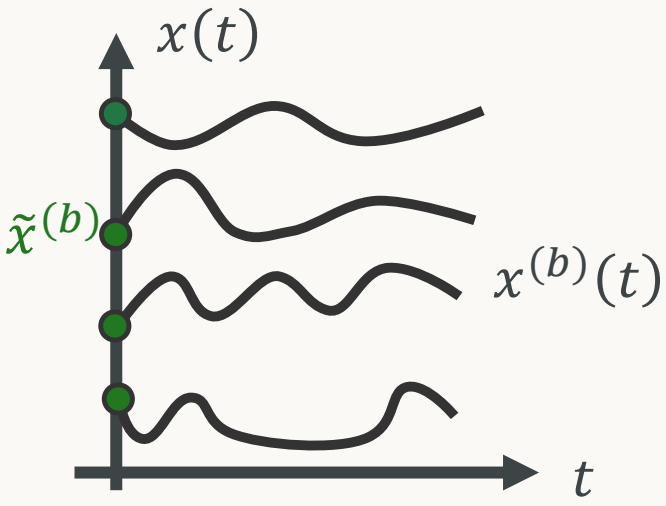
Example: 1-dimensional system, autonomous system $\frac{dx}{dt} = f_\omega(x)$

Background: Training Neural ODEs

Example: 1-dimensional system, autonomous system $\frac{dx}{dt} = f_\omega(x)$

Data

Time series starting from different initial conditions.

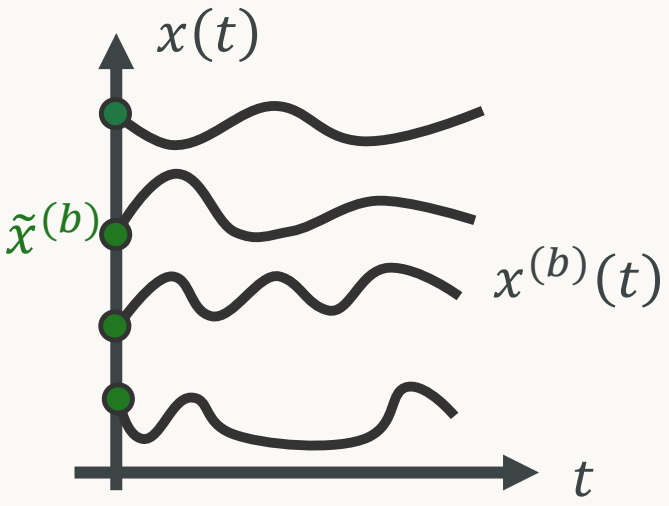


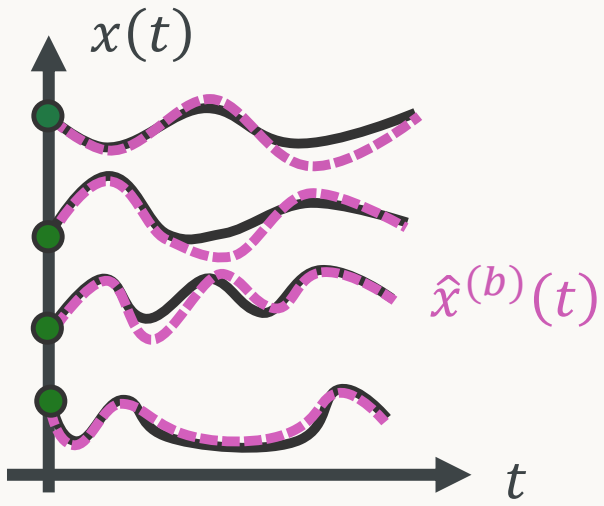
Background: Training Neural ODEs

Example: 1-dimensional system, autonomous system $\frac{dx}{dt} = f_\omega(x)$

Data
Time series starting from different initial conditions.

Prediction
Numerical Integration
e.g., Runge-Kutta or Euler.





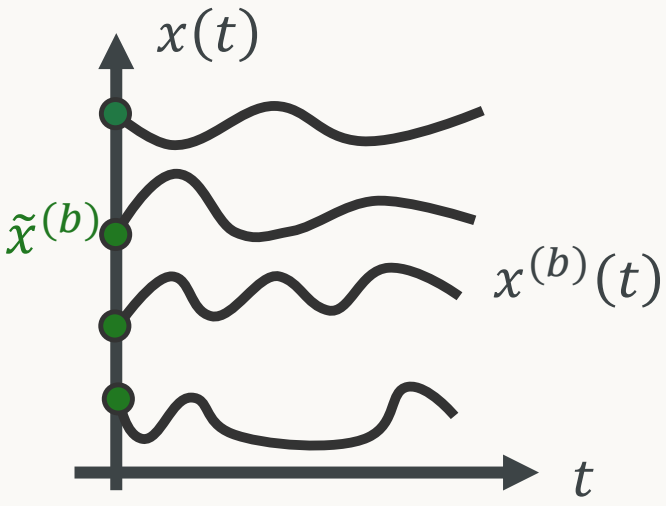
Background: Training Neural ODEs

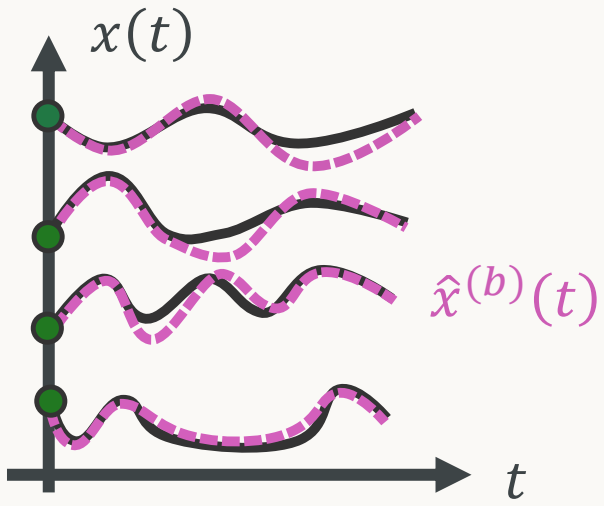
Example: 1-dimensional system, autonomous system $\frac{dx}{dt} = f_\omega(x)$

Data
Time series starting from different initial conditions.

Prediction
Numerical Integration
e.g., Runge-Kutta or Euler.

Optimization
Stochastic Gradient Descent with gradients obtained via backpropagation.



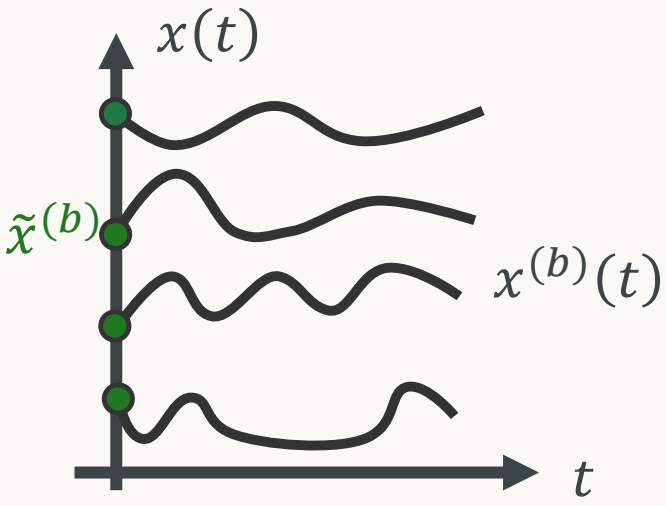


$$\omega \mapsto \omega - \eta \nabla_\omega \mathcal{L}(x, \hat{x})$$

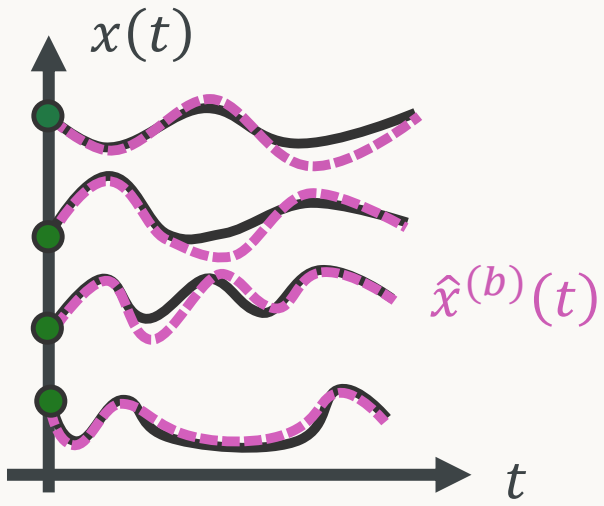
Background: Training Neural ODEs

Example: 1-dimensional system, autonomous system $\frac{dx}{dt} = f_\omega(x)$

Data
Time series starting from different initial conditions.



Prediction
Numerical Integration
e.g., Runge-Kutta or Euler.



Optimization
Stochastic Gradient Descent with gradients obtained via backpropagation.

$$\omega \mapsto \omega - \eta \nabla_\omega \mathcal{L}(x, \hat{x})$$

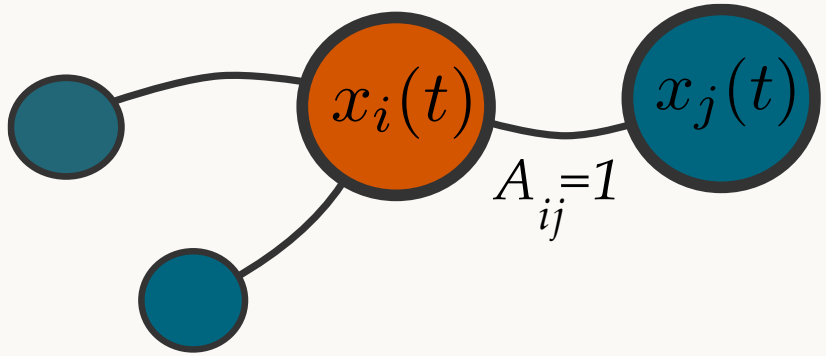
There are several ways to compute this term, some of them still actively researched [1].

[1] *Kidger* PhD Thesis (2022)

Background: ODEs on Graphs

Many dynamical systems on graphs are well described by [1]

$$\frac{dx_i}{dt} = f\left(x_i(t)\right) + \sum_{j \in \mathcal{V}} A_{ij} h^{\text{ego}}\left(x_i(t)\right) h^{\text{alt}}\left(x_j(t)\right)$$

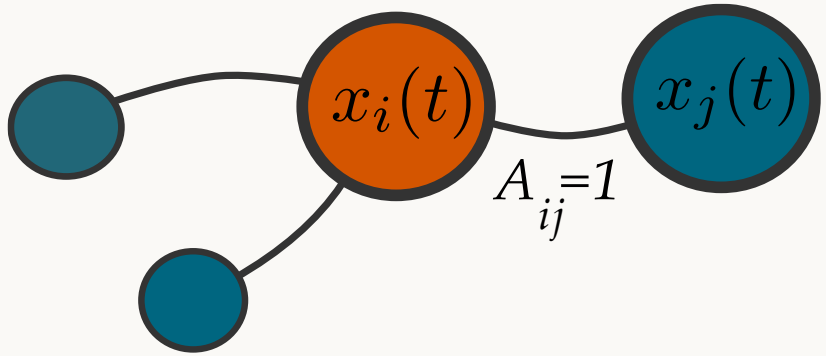


[1] *Barzel & Barabási* Nat. Phys. 9 673-681 (2013)

Background: ODEs on Graphs

Many dynamical systems on graphs are well described by [1]

$$\frac{dx_i}{dt} = f\left(x_i(t)\right) + \sum_{j \in \mathcal{V}} A_{ij} h^{\text{ego}}\left(x_i(t)\right) h^{\text{alt}}\left(x_j(t)\right)$$



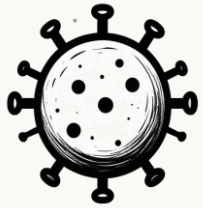
Example: SIS-Model

$$\frac{dx_i}{dt} = -\mu x_i(t) + \sum_{j \in \mathcal{V}} A_{ij}\, \beta\left(1 - x_i(t)\right) x_j(t)$$

[1] *Barzel & Barabási* Nat. Phys. 9 673-681 (2013)

Setup: Dynamics

We use synthetic data from **five dynamical systems** from different domains [1,2].



Susceptible-Infected-Susceptible (SIS)

Mass-Action Kinetics (MAK)

Michaelis-Menten Model (MM)

Birth-Death Process (BD)

Neuronal Dynamics (ND)

[1] *Meena* et al. Nat. Phys. 19 1033 (2023) [2] *Hens et al.* Nat. Phys. 15 403-412 (2019)

Neural ODEs on Graphs

Replace f, h^{ego}, h^{alt} by separate neural networks f_ω, h_ω^{ego}, h_ω^{alt} to obtain a Neural ODE:

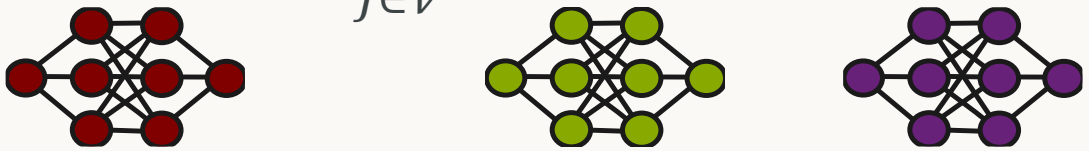
$$\frac{dx_i}{dt} = f\big(x_i(t)\big) + \sum_{j \in \mathcal{V}} A_{ij} h^{\text{ego}}\big(x_i(t)\big) h^{\text{alt}}\Big(x_j(t)\Big)$$

Neural ODEs on Graphs

Replace f, h^{ego}, h^{alt} by separate neural networks f_ω, h_ω^{ego}, h_ω^{alt} to obtain a Neural ODE[1,2,3]:

$$\frac{dx_i}{dt} = f\left(x_i(t)\right) + \sum_{j \in \mathcal{V}} A_{ij} h^{\text{ego}}\left(x_i(t)\right) h^{\text{alt}}\left(x_j(t)\right)$$

$$\frac{dx_i}{dt} = f_\omega\left(x_i(t)\right) + \sum_{j \in \mathcal{V}} A_{ij}\, h_\omega^{\text{ego}}\left(x_i(t)\right) h_\omega^{\text{alt}}\left(x_j(t)\right)$$



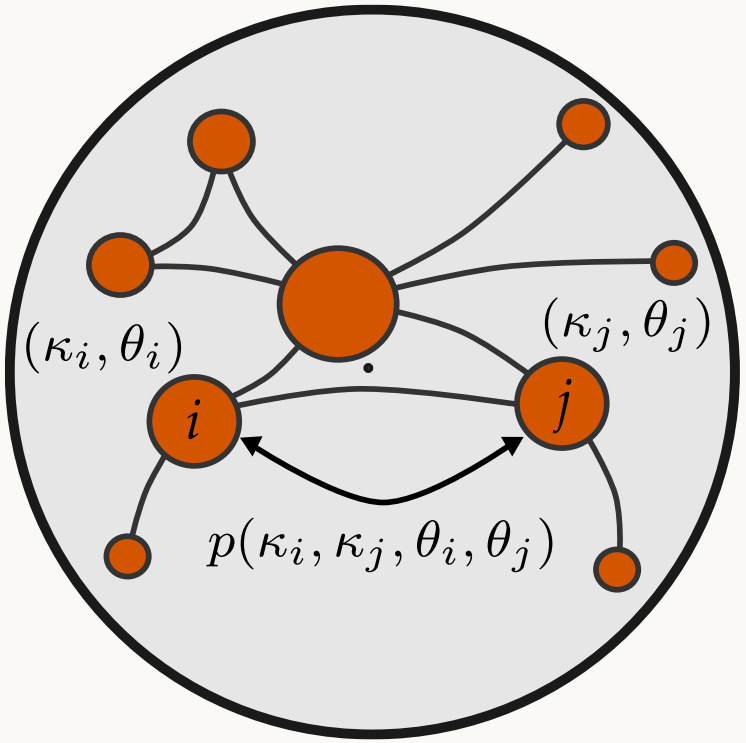
[1] *Poli et al.* AAAI (2022) [2] *Gao et al.* Nat.Comms. 15 (1) 6029 (2024) [3] *Vasiliauskaite & Antulov-Fantulin* Comms Phys. 7 1 (2024)

Background: $\mathbb{S}^1$ model

We use the $\mathbb{S}^1$ model [1,2] to generate graphs with different properties.

The model has **four parameters**:

- the *number of nodes* n
- the *average degree* $\bar{k}$
- the *exponent* γ of the degree distribution
  → controls degree heterogeneity
- The *inverse temperature* β
  → controls the clustering coefficient



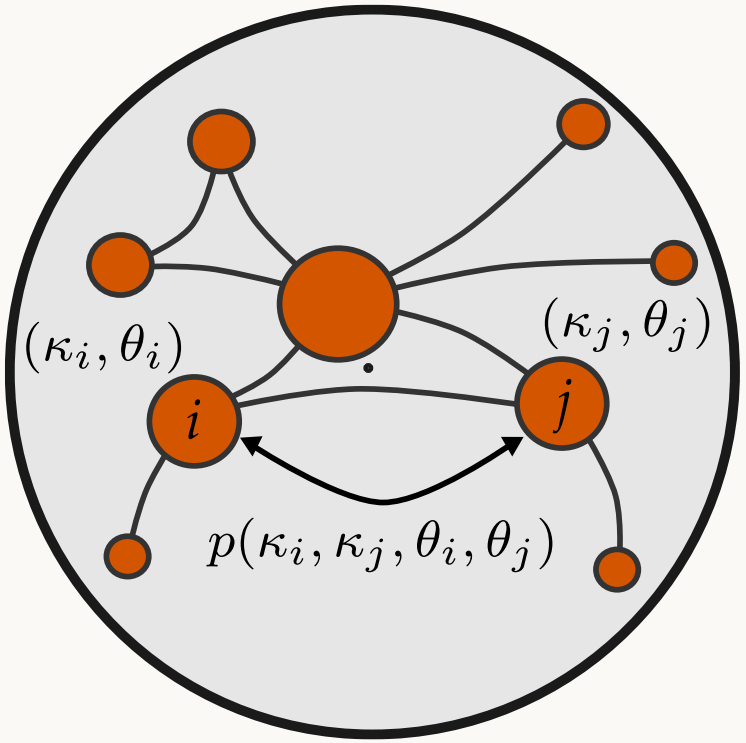
[1] *Serrano* et al. PRL 100, 078701 (2008) [2] *Krioukov* et al. PRE 82 036106 (2010)

Background: $\mathbb{S}^1$ model

We use the $\mathbb{S}^1$ model [1,2] to generate graphs with different properties.

The model has **four parameters**:

- the *number of nodes* n
- the *average degree* $\bar{k}$
- the *exponent* γ of the degree distribution
  → controls degree heterogeneity: **small γ → large hubs**
- The *inverse temperature* β
  → controls the clustering coefficient: **large β → strong clustering**



[1] *Serrano* et al. PRL 100, 078701 (2008) [2] *Krioukov* et al. PRE 82 036106 (2010)

Setup

Graphs

- number of nodes: $n \in \{64, \dots, 8192\}$
- average degree $\bar{k} = 10$
- exponent $\gamma \in [2.1, \dots, 3.9]$
- inverse temperature $\beta \in [0.1, \dots, 4.1]$

Dynamics

- SIS-Model (SIS),
- Mass-Action Kinetics (MAK)
- Birth-Death Process (BD)
- Neural Dynamics (ND)
- Michaelis-Menten (MM) model

Model

- 3 MLPs with layers 3 with 64 neurons.
- Dormand-Prince 4(5) ODE solver

Evaluation

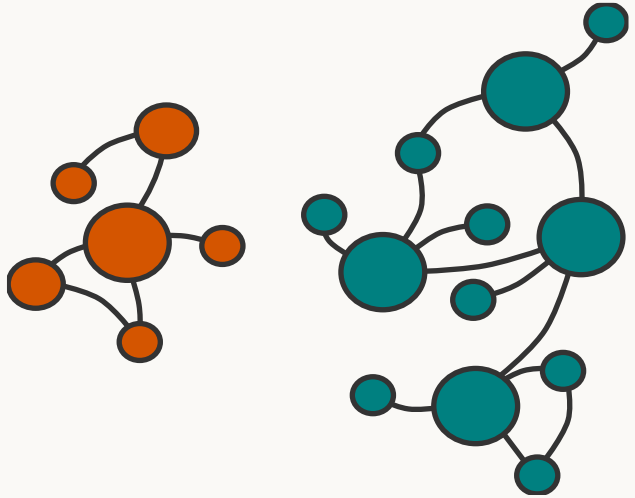
- mean absolute error $\mathcal{L}_{\text{mae}}$
- 100 test graphs

Generalization: Larger Graphs

Question: Can Neural ODEs, trained on small graphs, generalize to larger ones?

Motivation

- Size generalization is highly desirable for efficiency.
- Theoretical and empirical evidence for size generalization in (graph) machine learning is mixed [1,2].

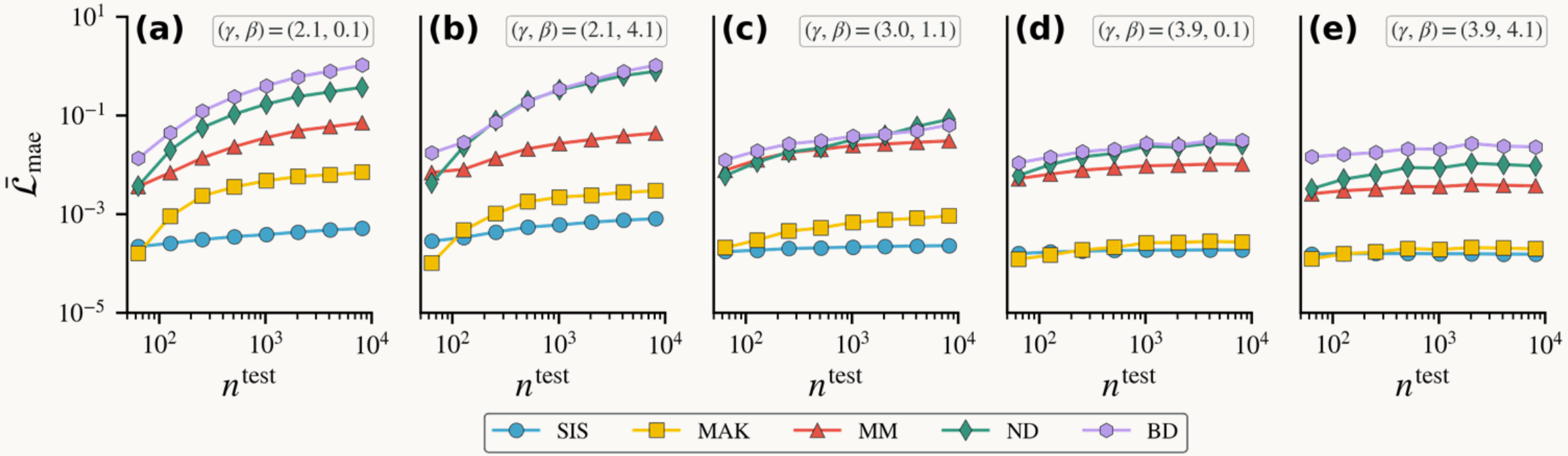


$n^{\text{train}} \ll n^{\text{test}}$

[1] *Yehudai et al.* ICML (2021) [2] *Anil et al.* NeurIPS (2022)

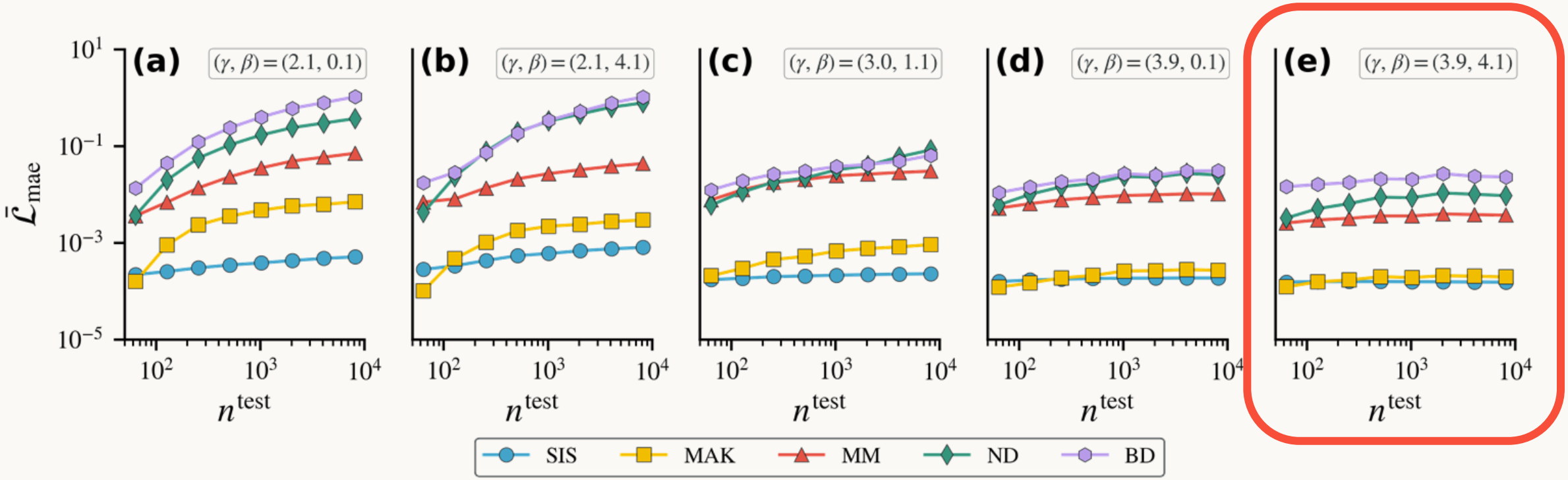
Generalization: Larger Graphs

Generalization to larger graphs is **possible for low degree heterogeneity.**
Clustering has limited influence on size generalization.



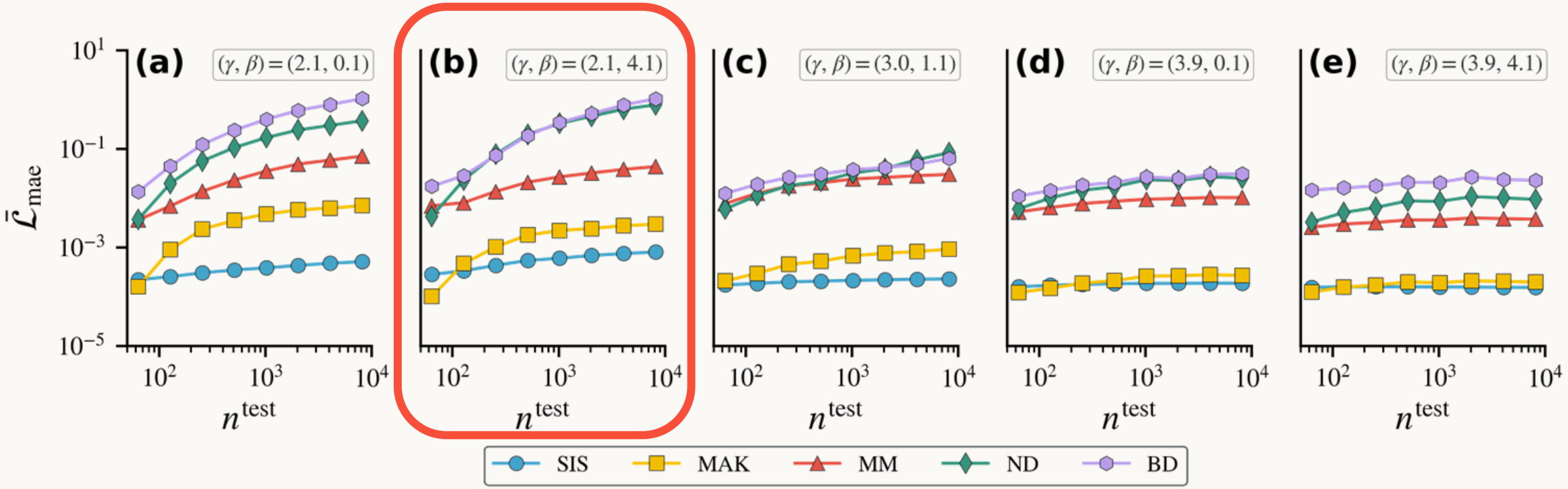
Generalization: Larger Graphs

Generalization to larger graphs is **possible for low degree heterogeneity.**
Clustering has limited influence on size generalization.



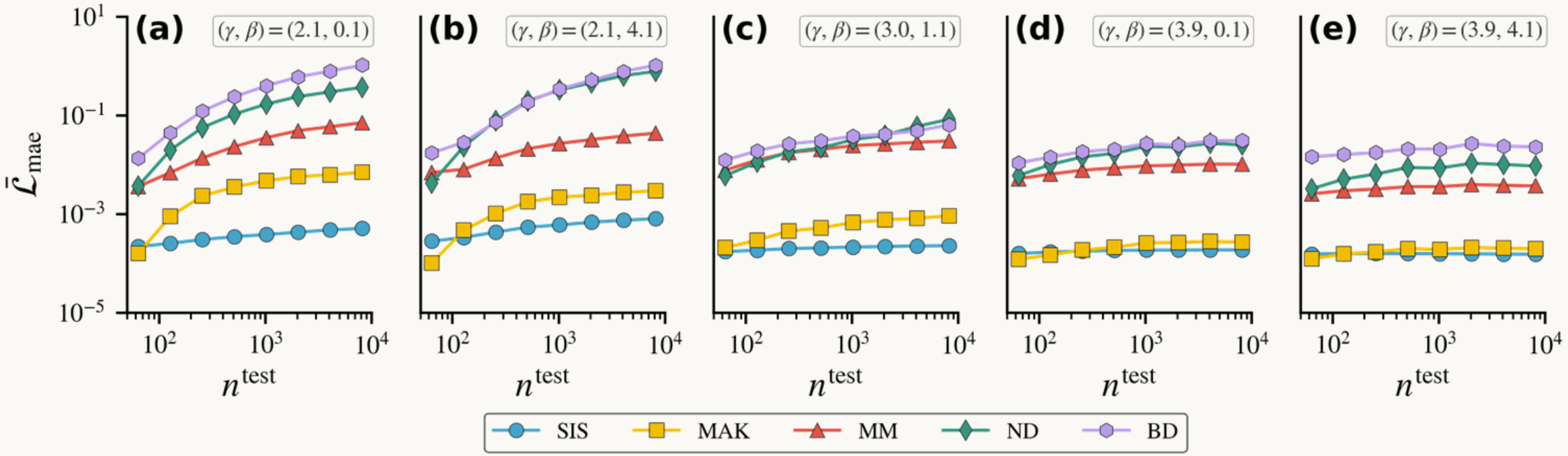
Generalization: Larger Graphs

Generalization to larger graphs is **possible for low degree heterogeneity.**
Clustering has limited influence on size generalization.



Generalization: Larger Graphs

Generalization to larger graphs is **possible for low degree heterogeneity.**
Clustering has limited influence on size generalization.



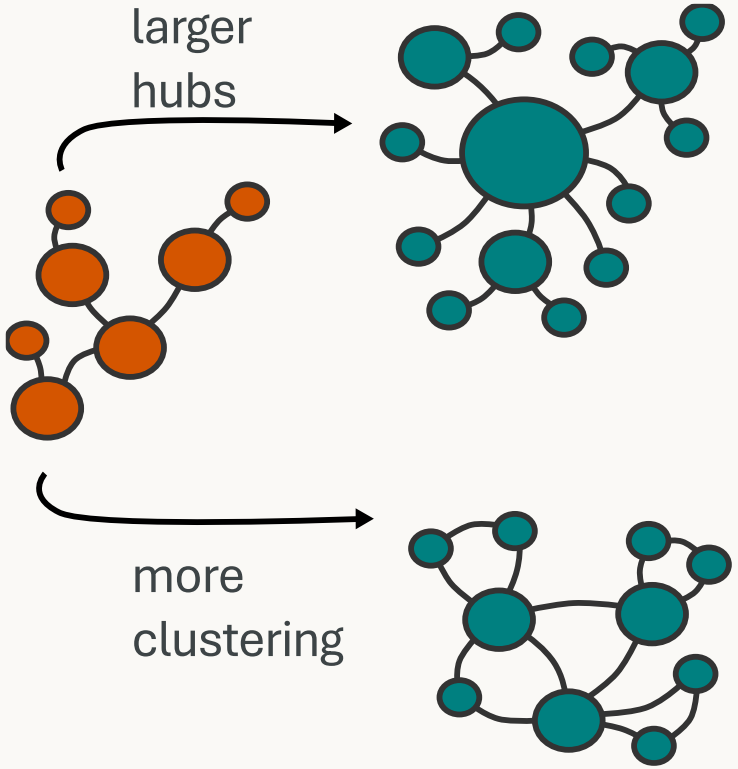
Generalization: Structurally Different Graphs

Question: Can Neural ODEs generalize to networks with different properties?

Motivation

- Many complex systems constantly evolve, causing potential mismatch between graph properties at training and test time.

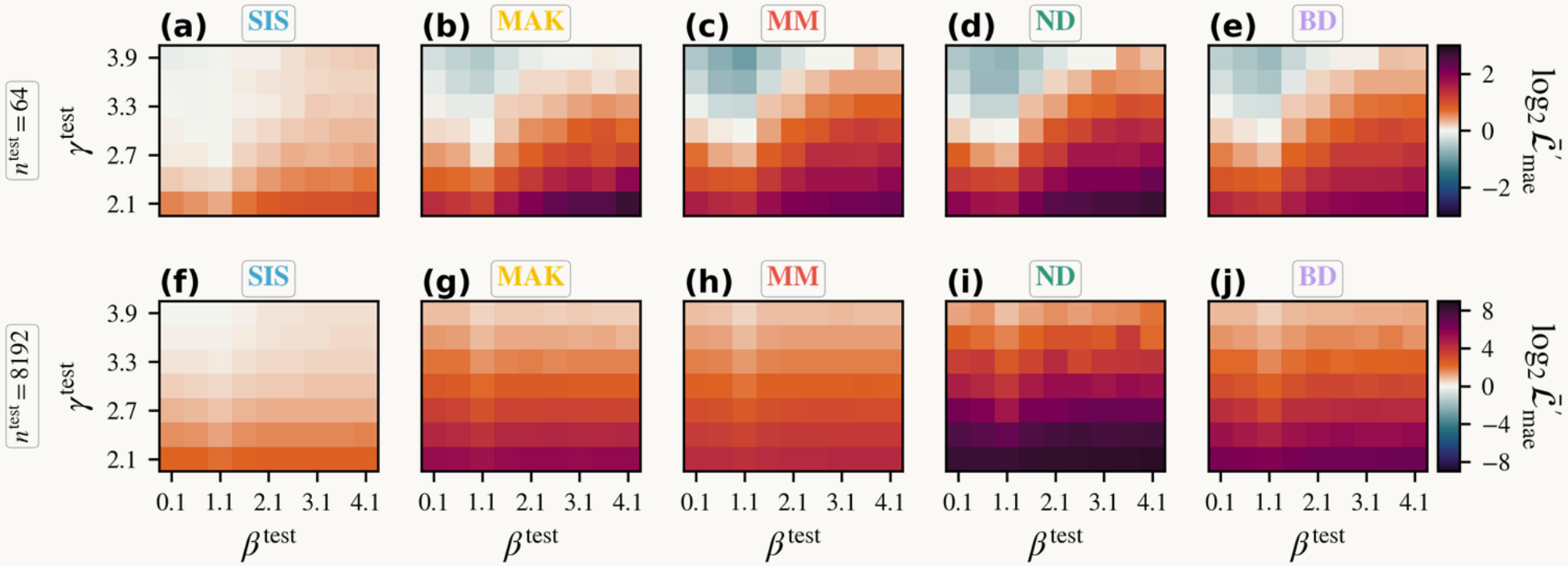
- Graph machine learning models are usually evaluated on a few benchmark datasets. Only recently, have random graph ensembles been recognized as a possible testbed [1,2,3].

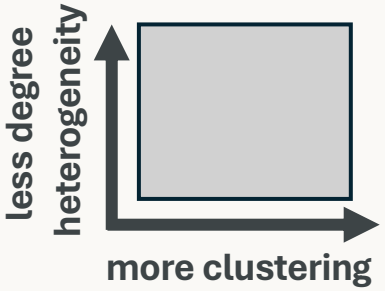


[1] *Palowitch et al.* KDD (2022) [2] *Aliakbarisani et al.* arXiv:2406.02772 (2024) [3] *Vasiliauskaite & Antulov-Fantulin* Comms Phys. 7 1 (2024)

Generalization: Structurally Different Graphs

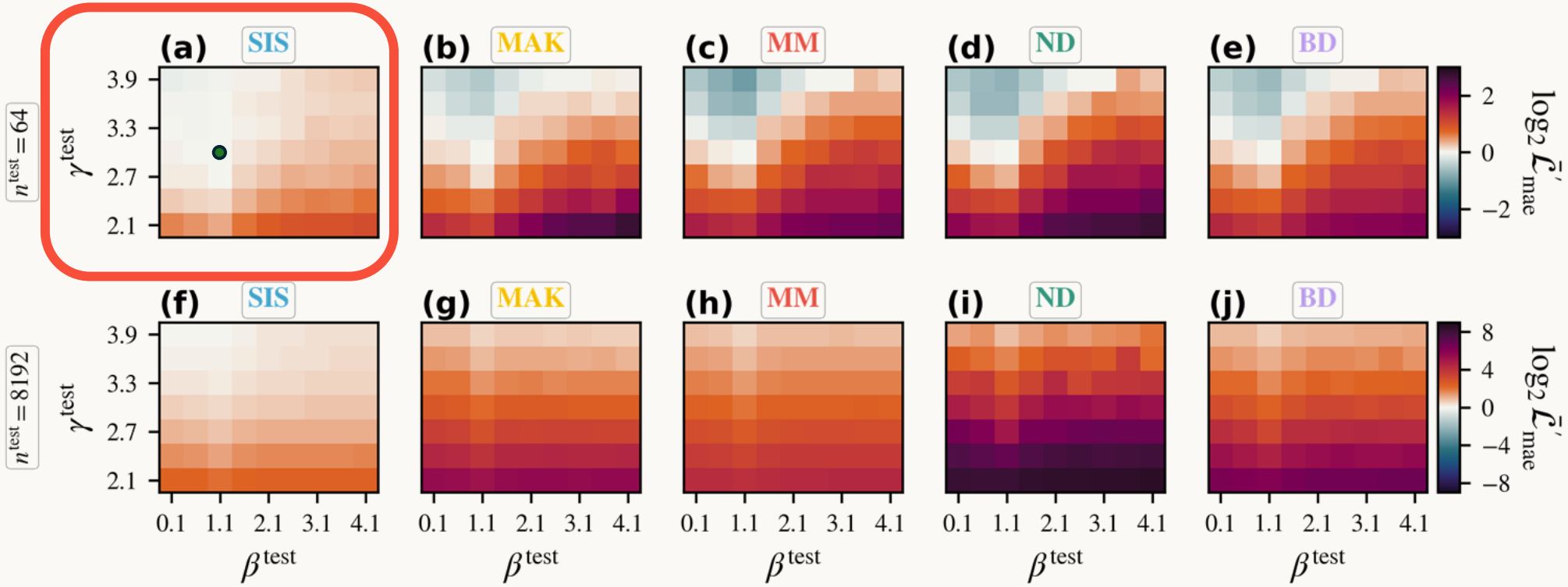
Performance is **good on less clustered**, and **less degree heterogeneous graphs**, but degrades with larger degree heterogeneity.

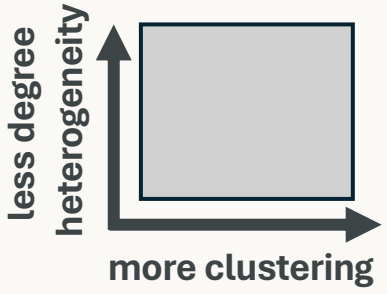




Generalization: Structurally Different Graphs

Performance is **good on less clustered**, and **less degree heterogeneous graphs**, but degrades with larger degree heterogeneity.

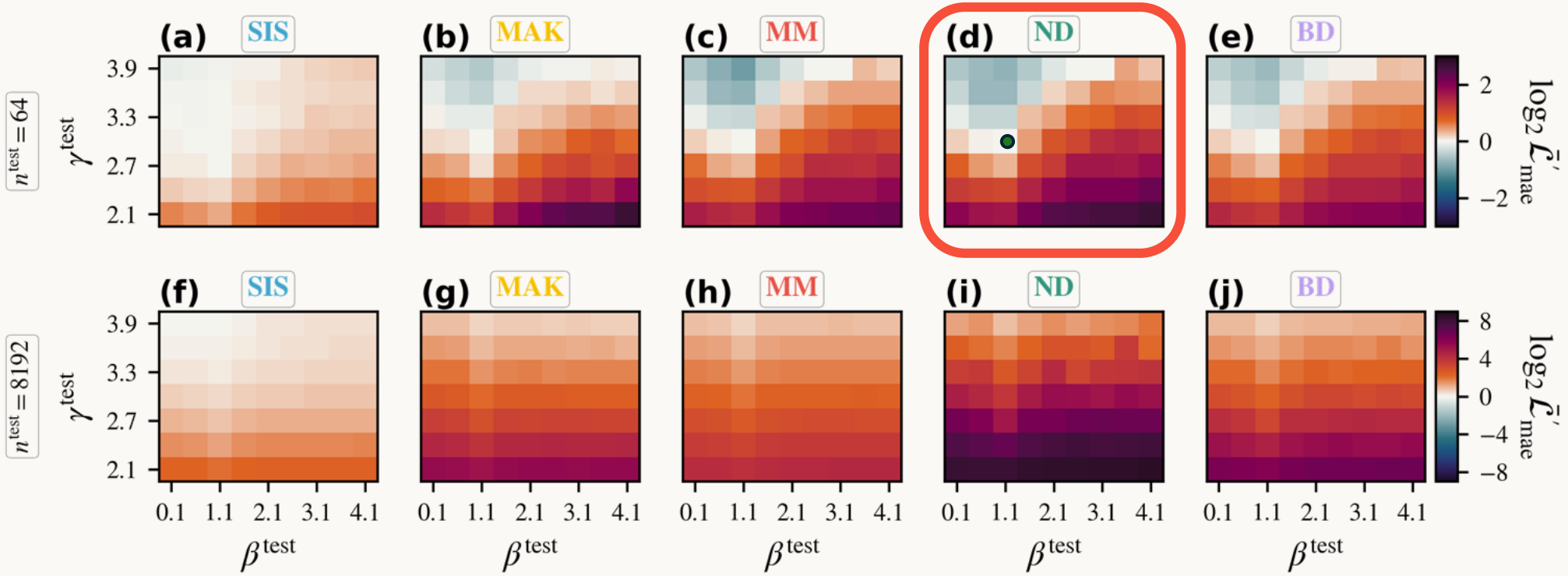


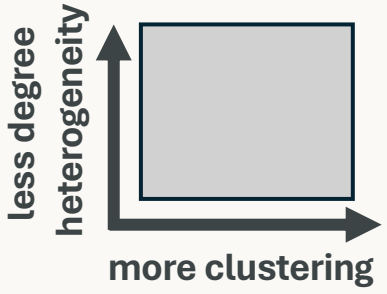


Training Parameters $(\gamma^{\text{train}}, \beta^{\text{train}}) = (3.0, 1.1)$

Generalization: Structurally different graphs

Performance is **good on less clustered**, and **less degree heterogeneous graphs**, but degrades with larger degree heterogeneity.

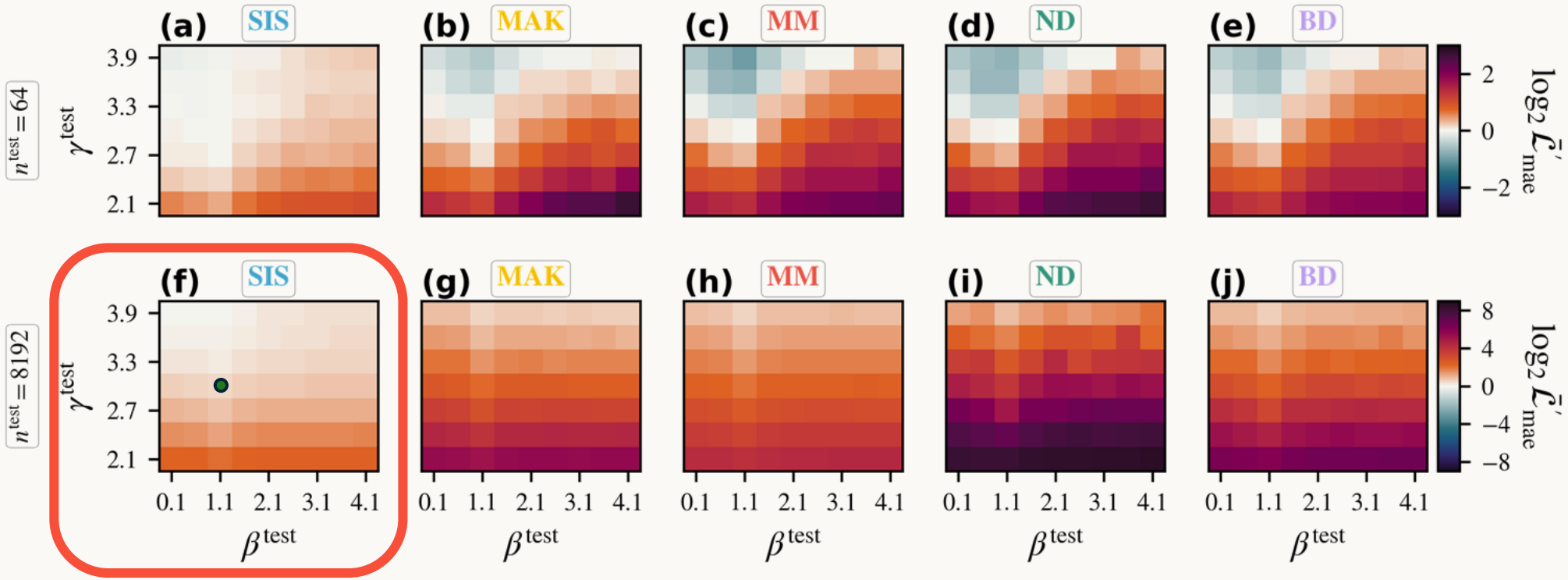


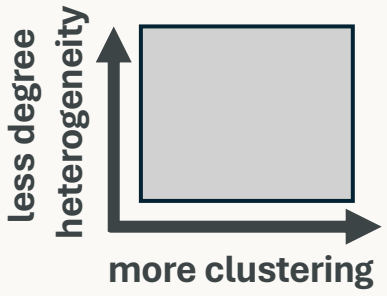


Training Parameters $(\gamma^{\text{train}}, \beta^{\text{train}}) = (3.0, 1.1)$

Generalization: Structurally Different Graphs

Performance is **good on less clustered**, and **less degree heterogeneous graphs**, but degrades with larger degree heterogeneity.

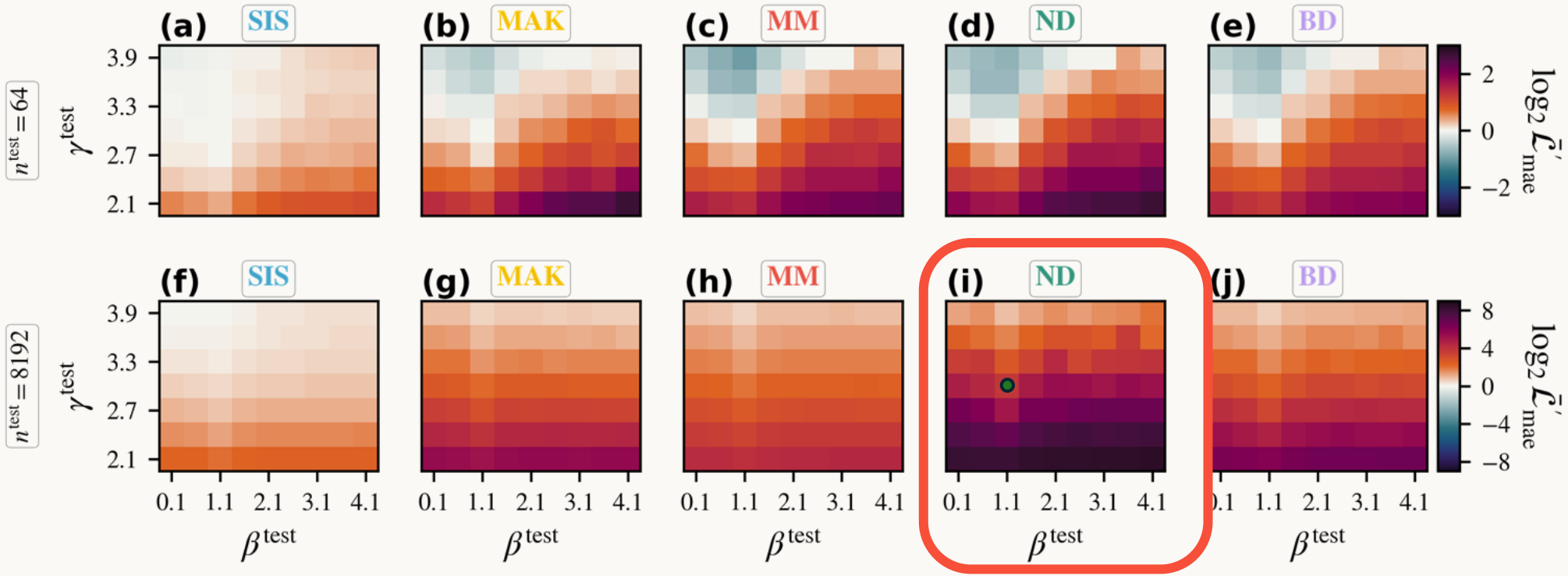


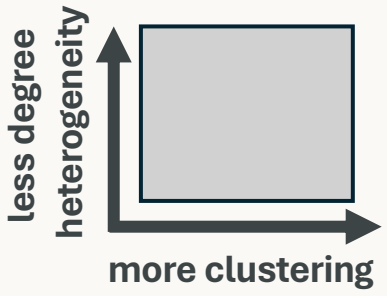


Training Parameters $(\gamma^{\text{train}}, \beta^{\text{train}}) = (3.0, 1.1)$

Generalization: Structurally Different Graphs

Performance is **good on less clustered**, and **less degree heterogeneous graphs**, but degrades with larger degree heterogeneity.

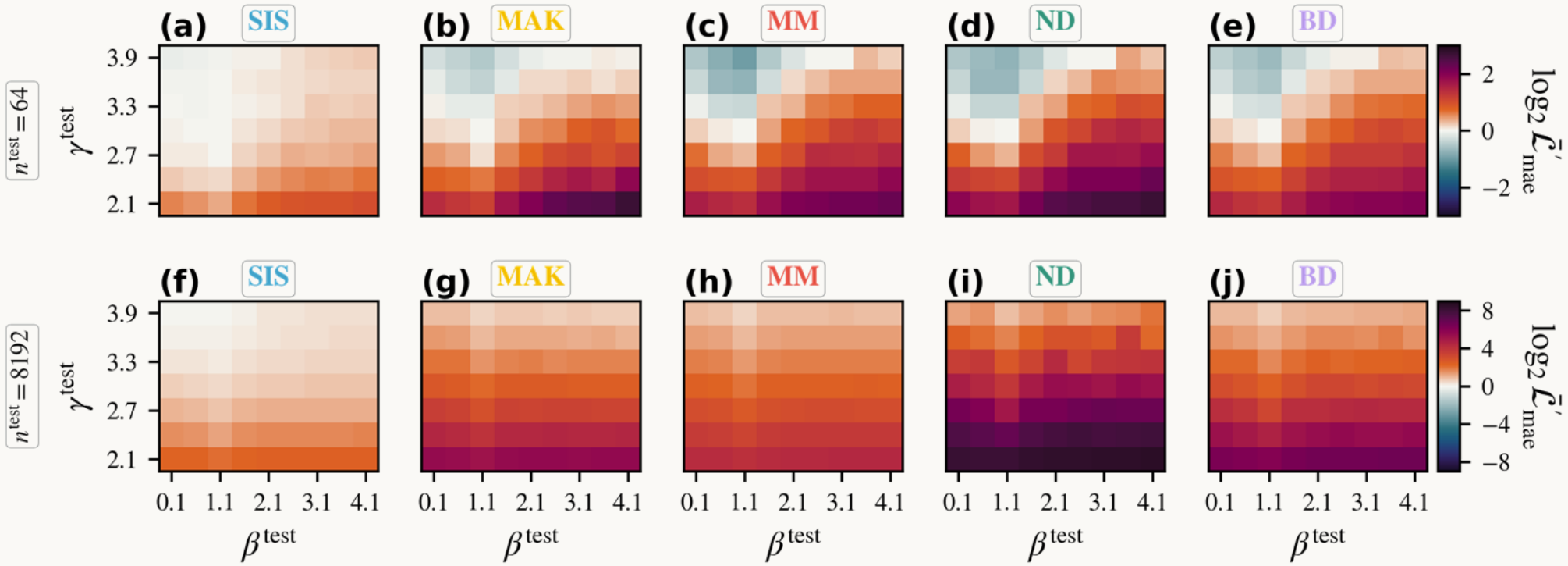


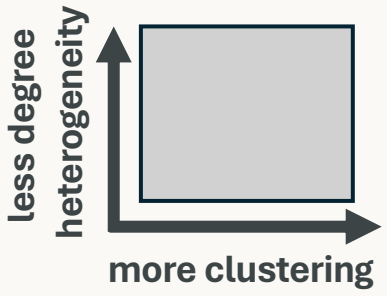


Training Parameters $(\gamma^{\text{train}}, \beta^{\text{train}}) = (3.0, 1.1)$

Generalization: Structurally Different Graphs

Performance is **good on less clustered**, and **less degree heterogeneous graphs**, but degrades with larger degree heterogeneity.





Conclusion

Neural ODEs can exhibit excellent size generalization for some dynamical systems and if graphs are not too degree heterogeneous.

However, they struggle to generalize to graphs that are more degree heterogeneous than the training graph and are brittle to missing nodes at time of deployment.

Preprint!

https://arxiv.org/abs/2602.08980

✉ **laber.m@northeastern.edu**

Copyright

Laber M., Klein B., Eliassi-Rad T., *When do neural ordinary differential equations generalize on complex networks?* arXiv:2602.08980 (2026)