From Discrete Choices to Continuous Spaces: Interpolating Models, Data, and Compute

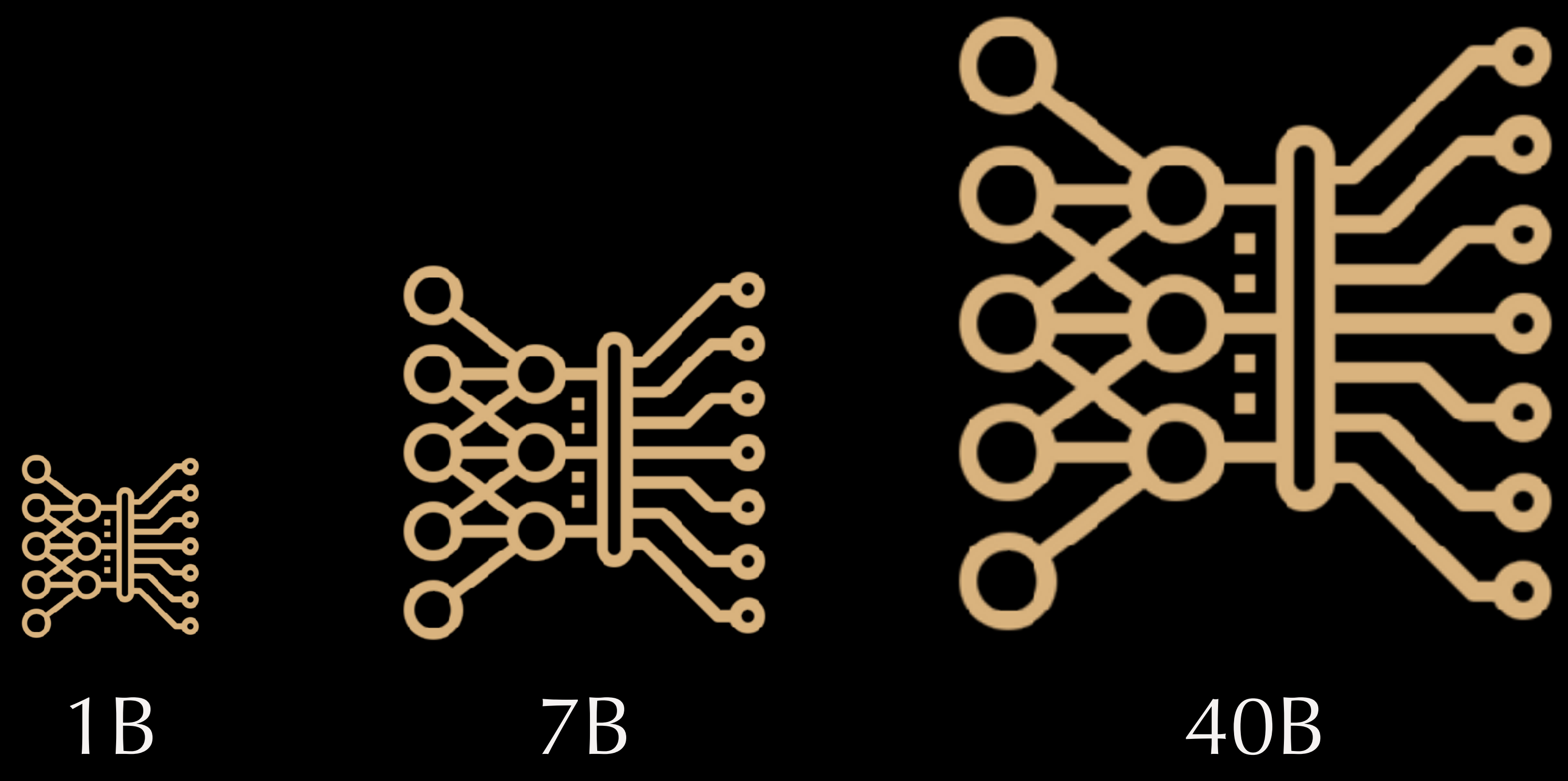
David Alvarez-Melis

Harvard SEAS | Kempner Institute | MSR

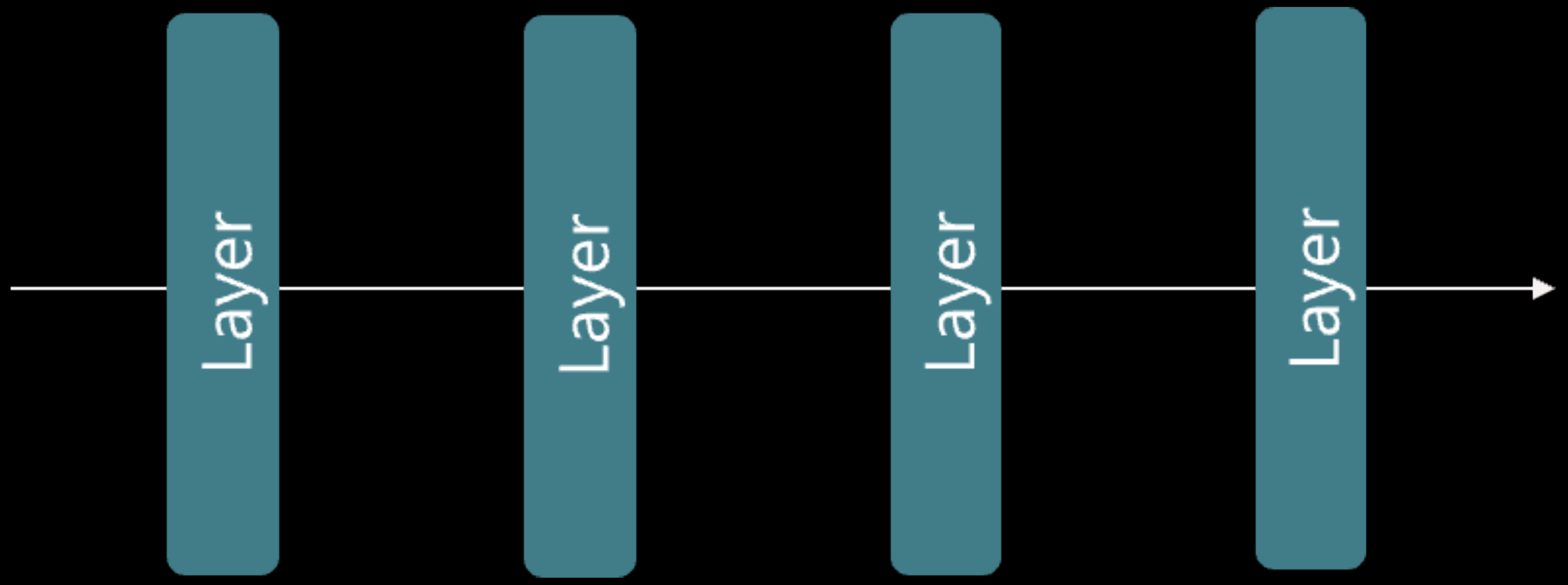
UniReps @ NeurIPS

Dec 6th, 2025

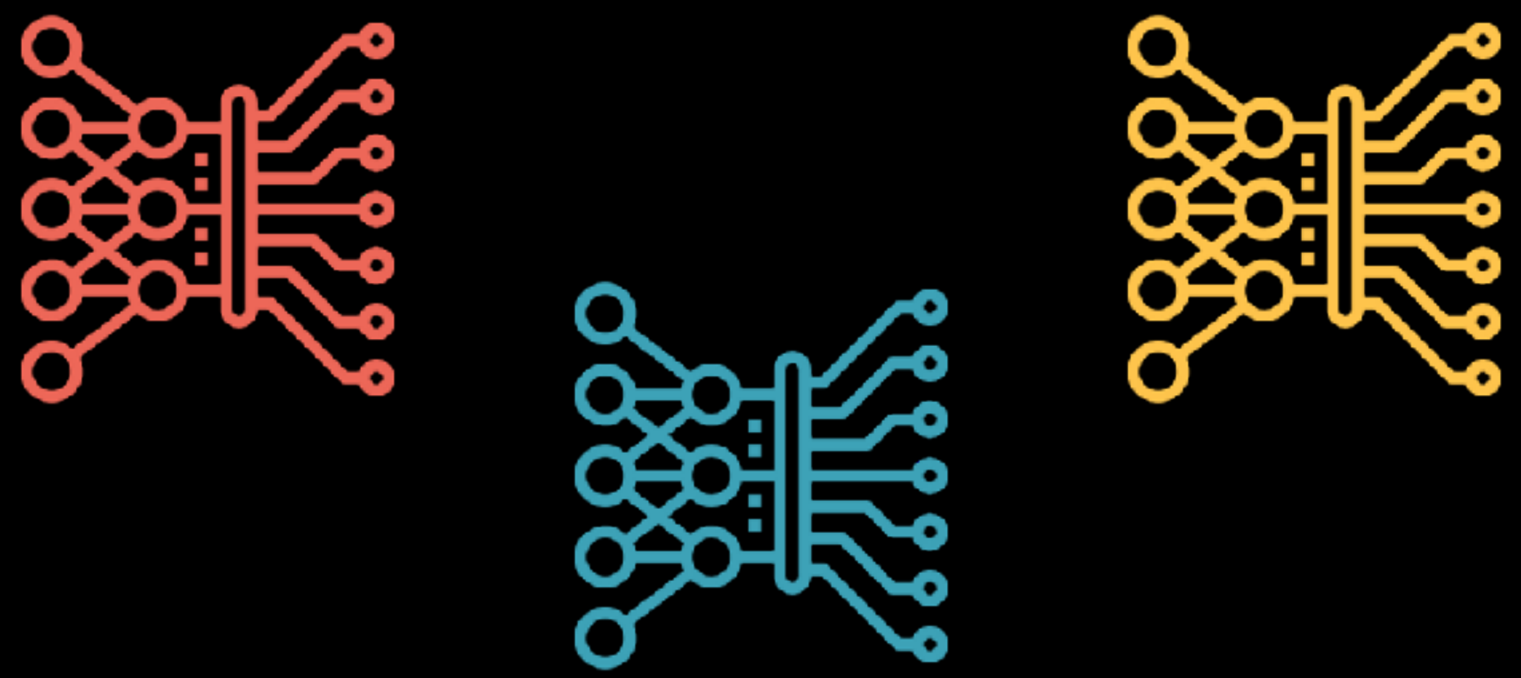
ML is full of discrete choices…



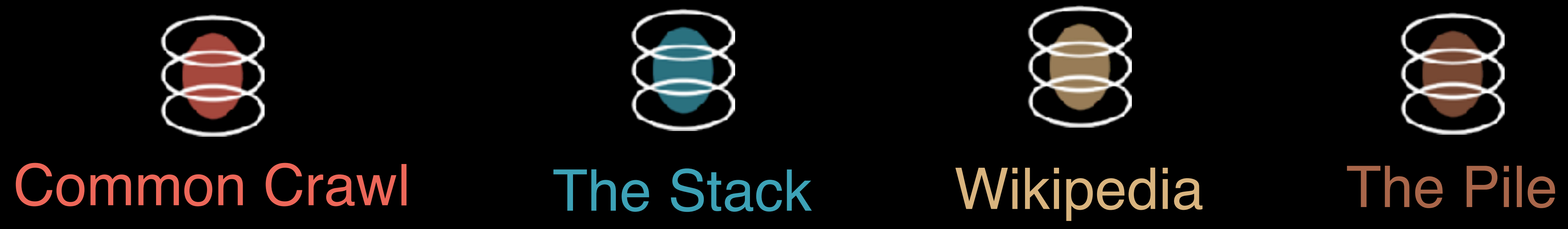
Model Sizes



Model Depth



Task-specific Models



Pretraining Datasets

Discrete by Design…Limited by Design

Discrete design spaces bring about many limitations:

Fragmented Design Space

- Combinatorial explosion, hard-to-optimize choices.
- Isolated points with no structure or trajectories between them.
- Can choose A or B, but have no notion of what lies “in between.”

Rigidity / Poor Adaptivity

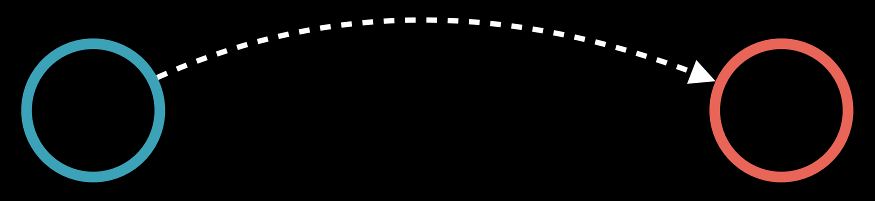
- Coarse-grained choices prevent smooth control over e.g compute.
- Discontinuous jumps in performance and compute cost.
- Hard to generalize or adjust capacity with locked-in configs.

Inefficiency and Duplication

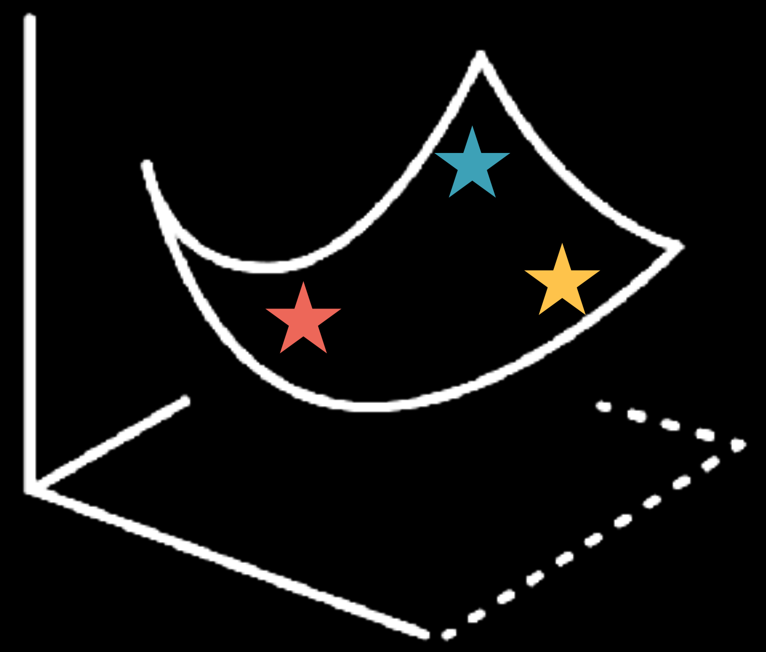
- Requires training + storing many separate models or datasets.
- Redundancy: structure not shared across isolated choices, wasteful.
- Domain shifts / new budgets require retraining from scratch.

Beyond Discrete Choices

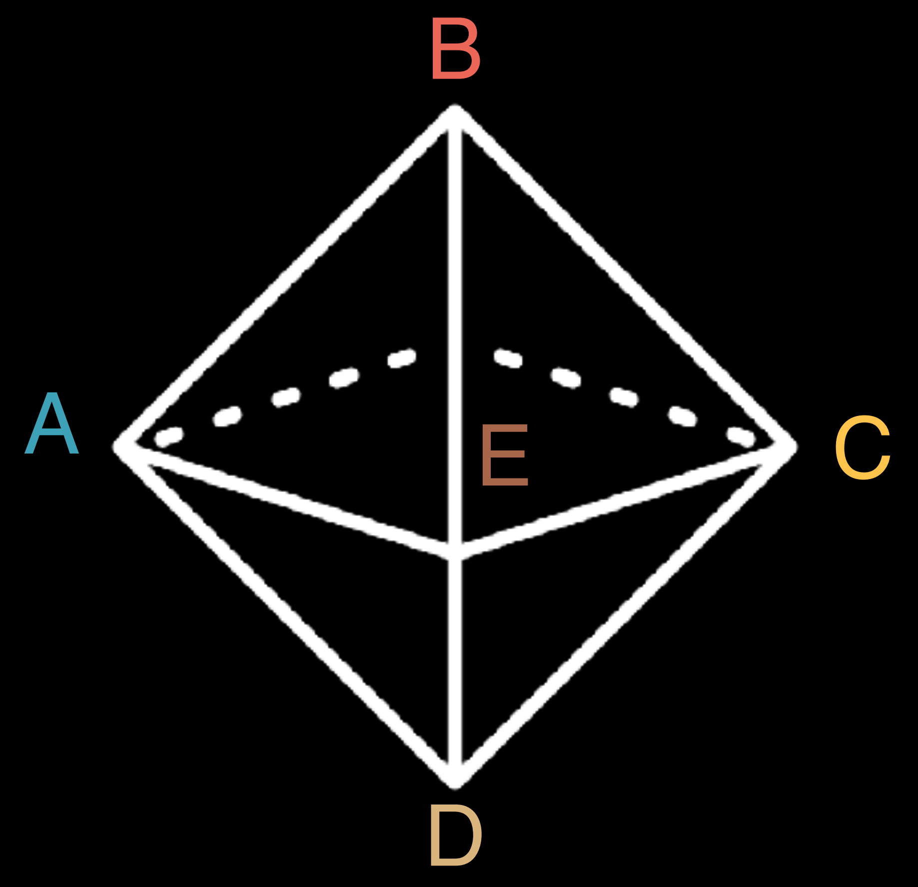
Can these discrete design choices be “smoothed”, embedded in continuous space?



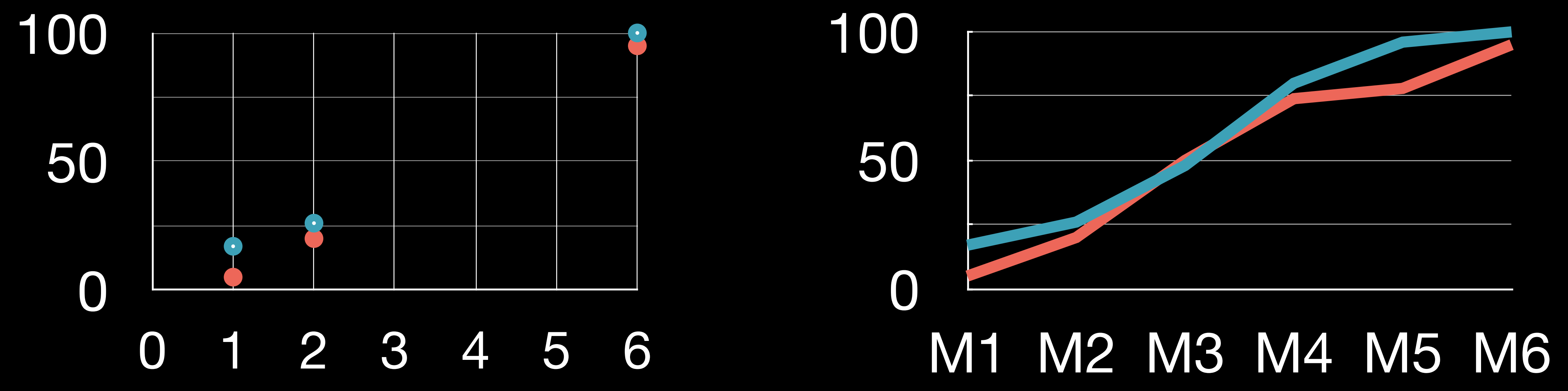
Disconnected points → Continuous paths
(configurations become connected rather than isolated)



One-off Solutions → Families of Related Solutions
(variations form curves, surfaces, manifolds)



Fixed Categories → Smooth Spectra
(depth, capacity, domains become adjustable axes)



Jumps → Transitions
(changes become gradual rather than abrupt)

Continuity adds structure between the choices we usually treat as separate.

Why Continuity Matters

Broader Design Space

- Continuous spaces give meaning to the “in-between.”
- Design choices form trajectories, not isolated points.

Fine-Grained Control

- Compute, capacity, and behavior can vary smoothly with context.
- Systems respond on a spectrum, not through discrete jumps.

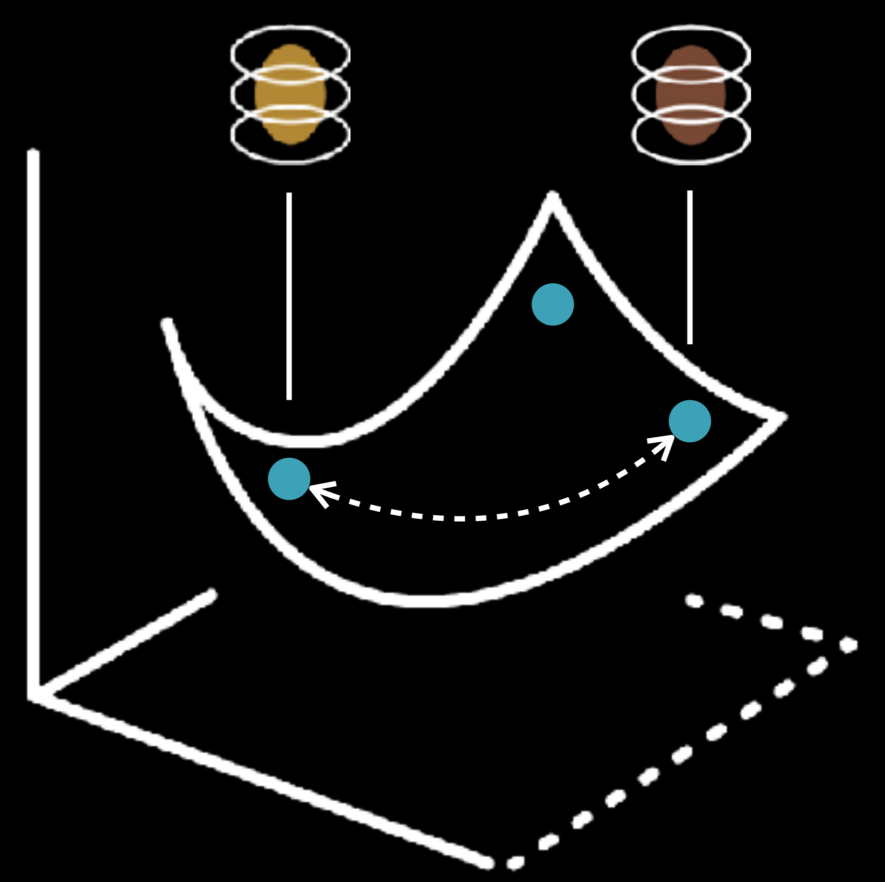
Efficiency by Sharing

- Shared structure across choices reduces training/storage cost.
- One continuous family can replace many discrete variants.

Continuum limits turn isolated choices into connected, navigable design spaces.

Three Continuum Limits in ML

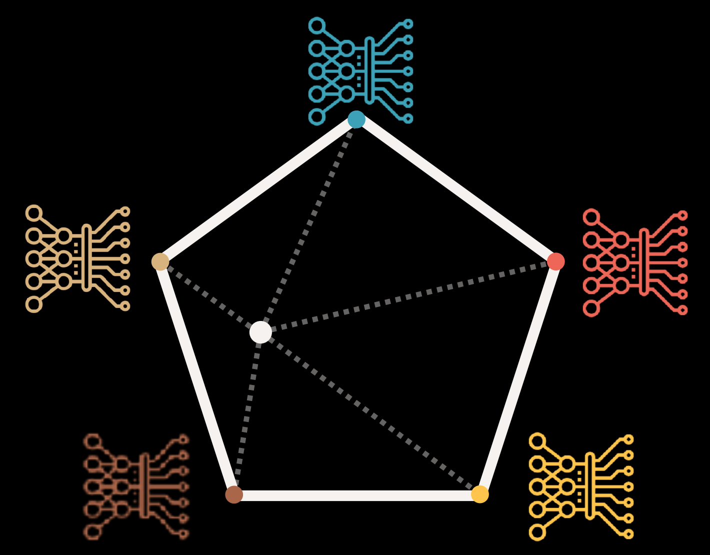
Datasets:



Interpolation along Generalized Geodesics

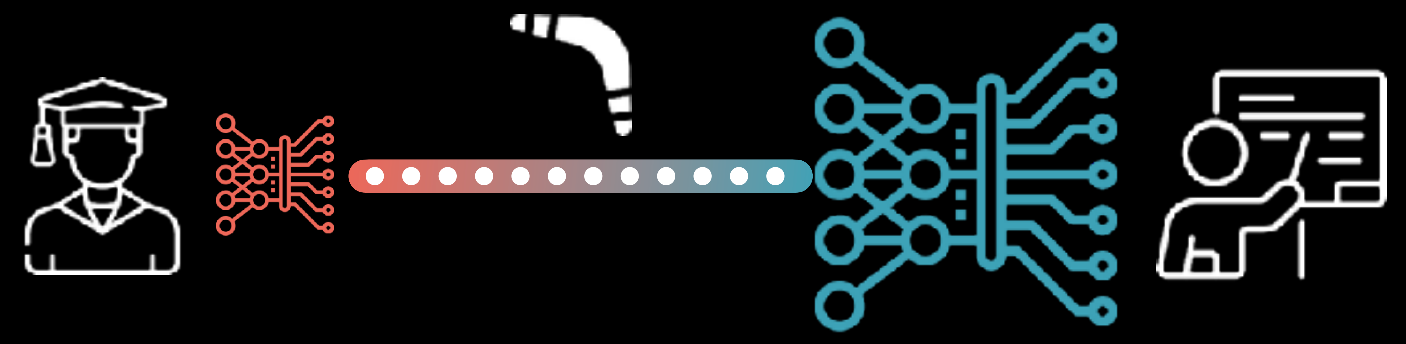
[Fan & AM, UAI 2023]

Models:



Weight-space interpolation

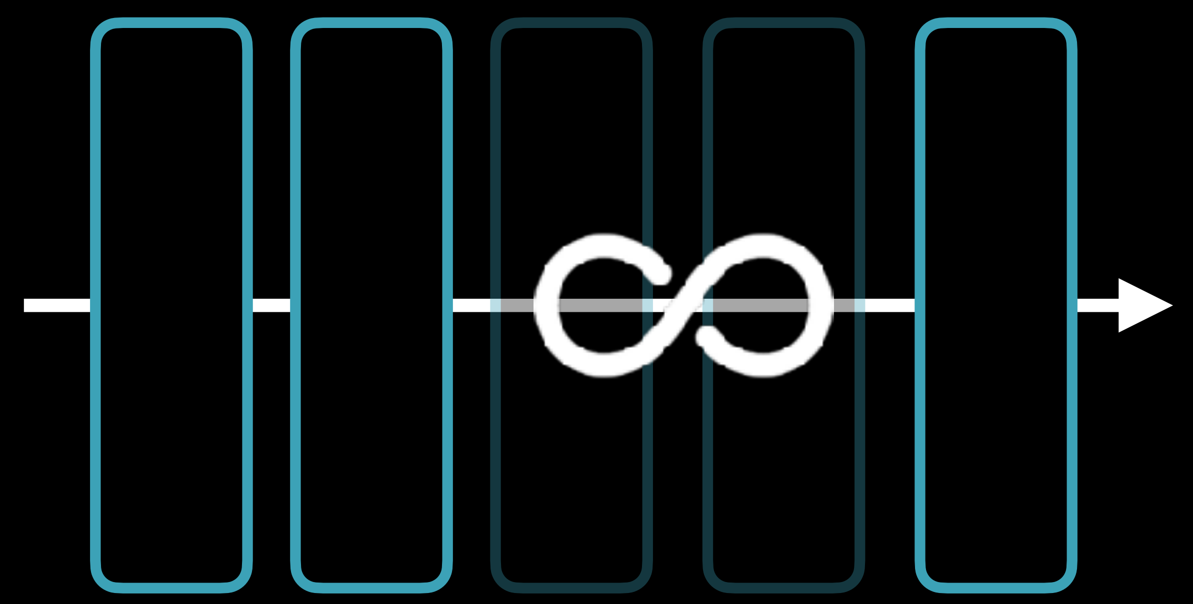
[Kangaslahti & AM, TMLR 2025]



Boomerang Distillation

[Kangaslahti et al., arXiv 2025]

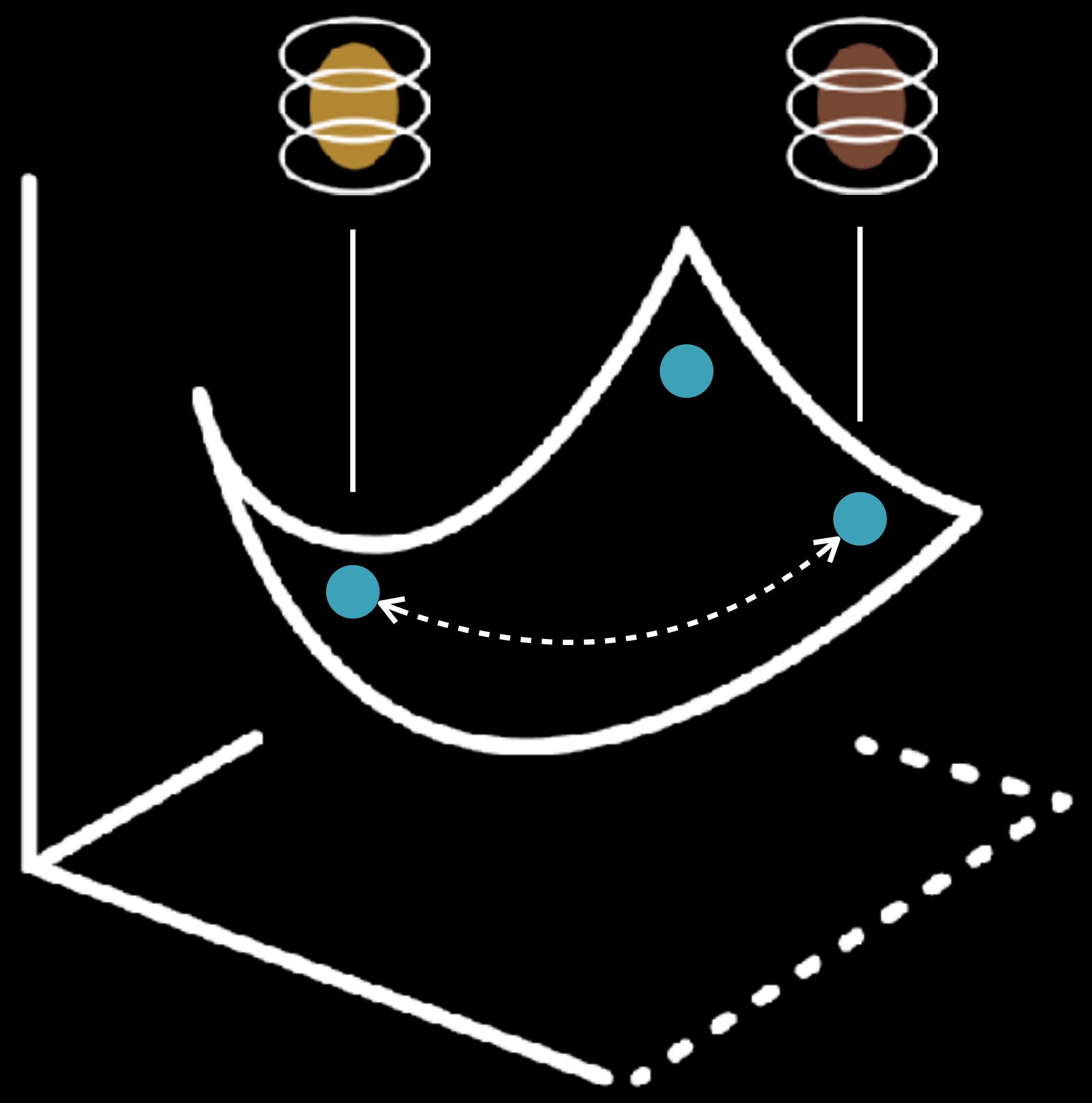
Compute:



Distributional Deep Equilibrium Models

[Geuter et al., AISTATS 2025]

Interpolating Datasets

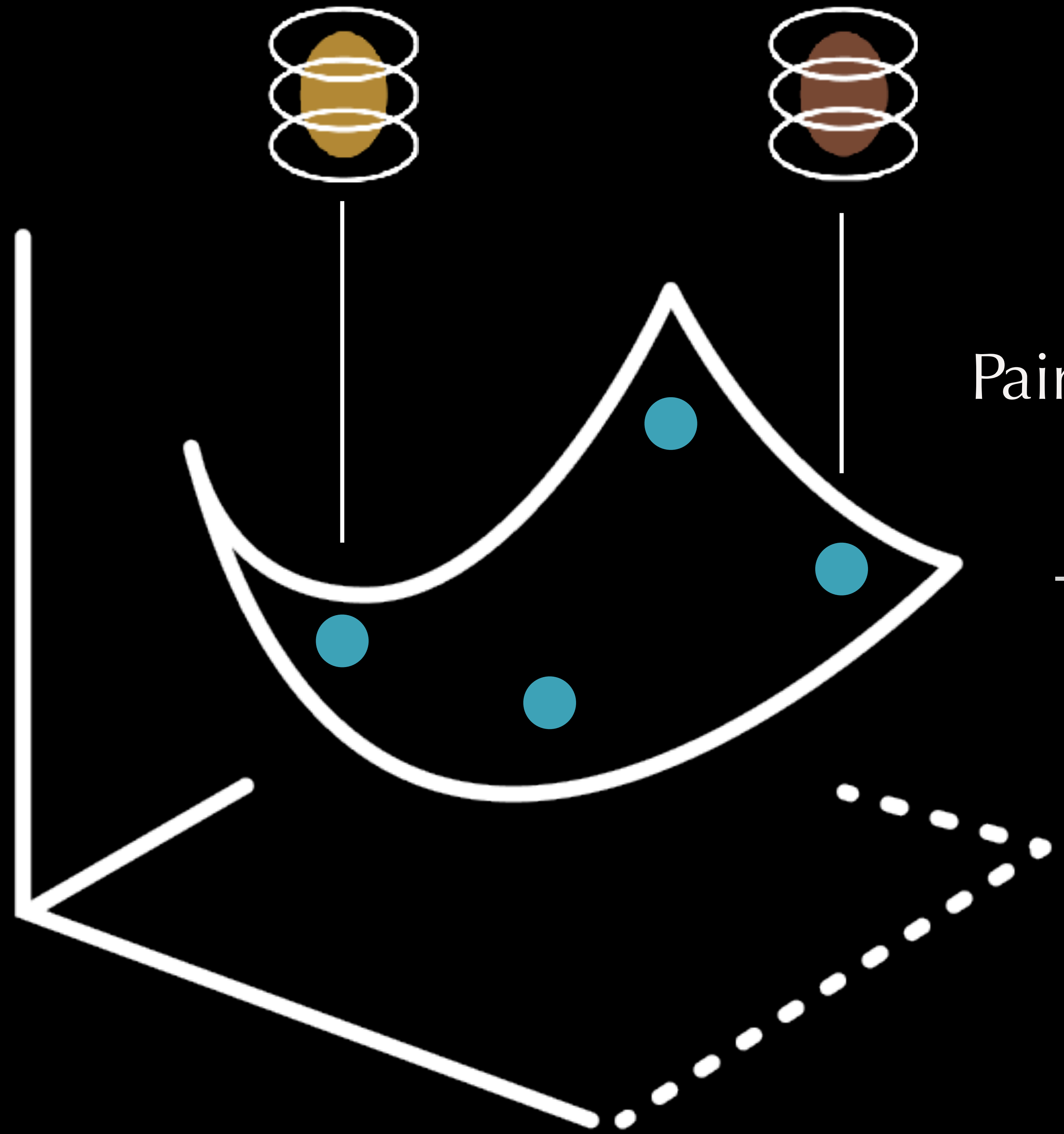


Generating Synthetic Datasets by Interpolating along Generalized Geodesics

Fan & AM, UAI 2023

Jiaojiao Fan

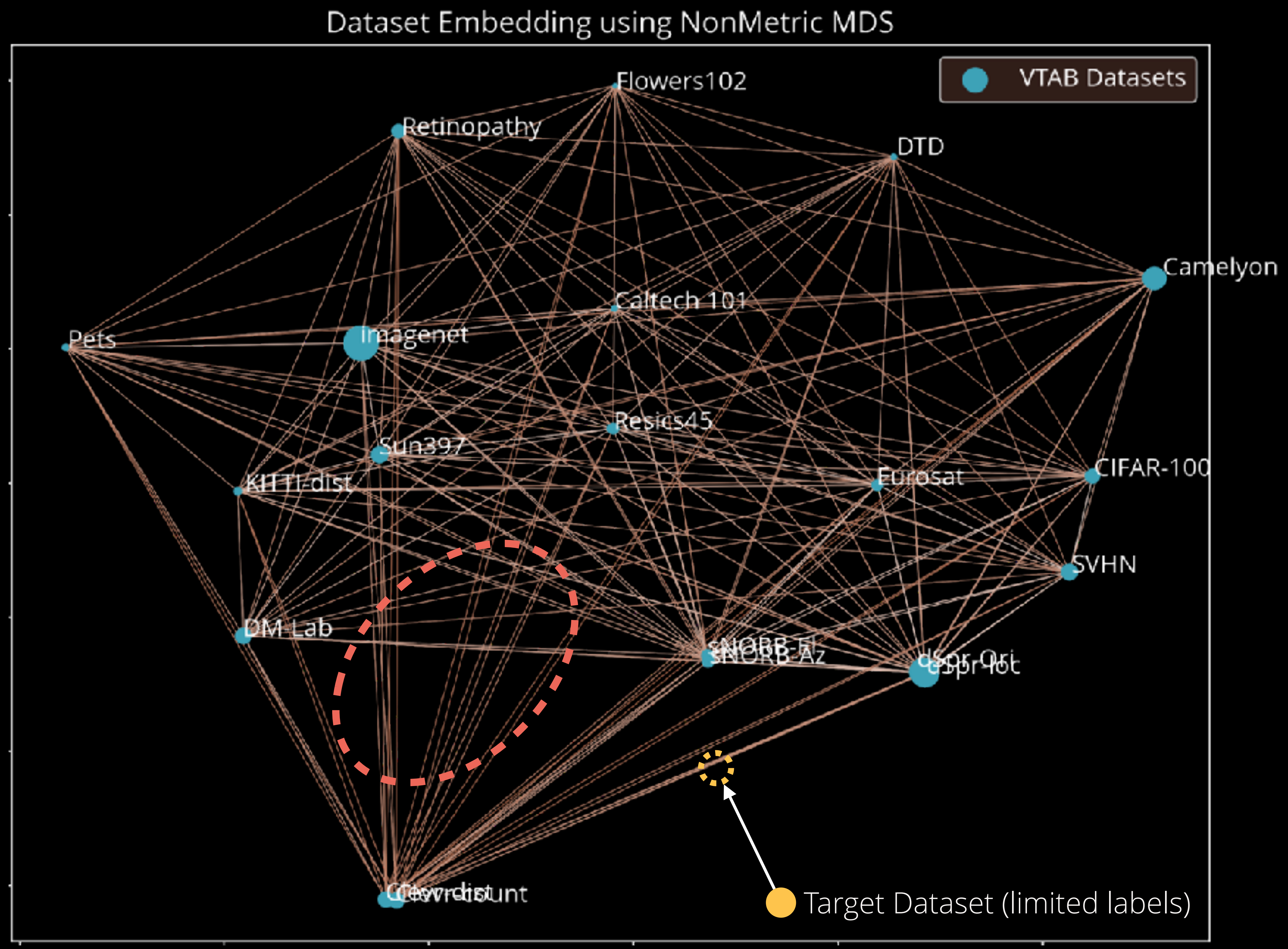
Dataset Space: A Continuous View



Training Datasets

Pairwise Optimal Transport dataset distances

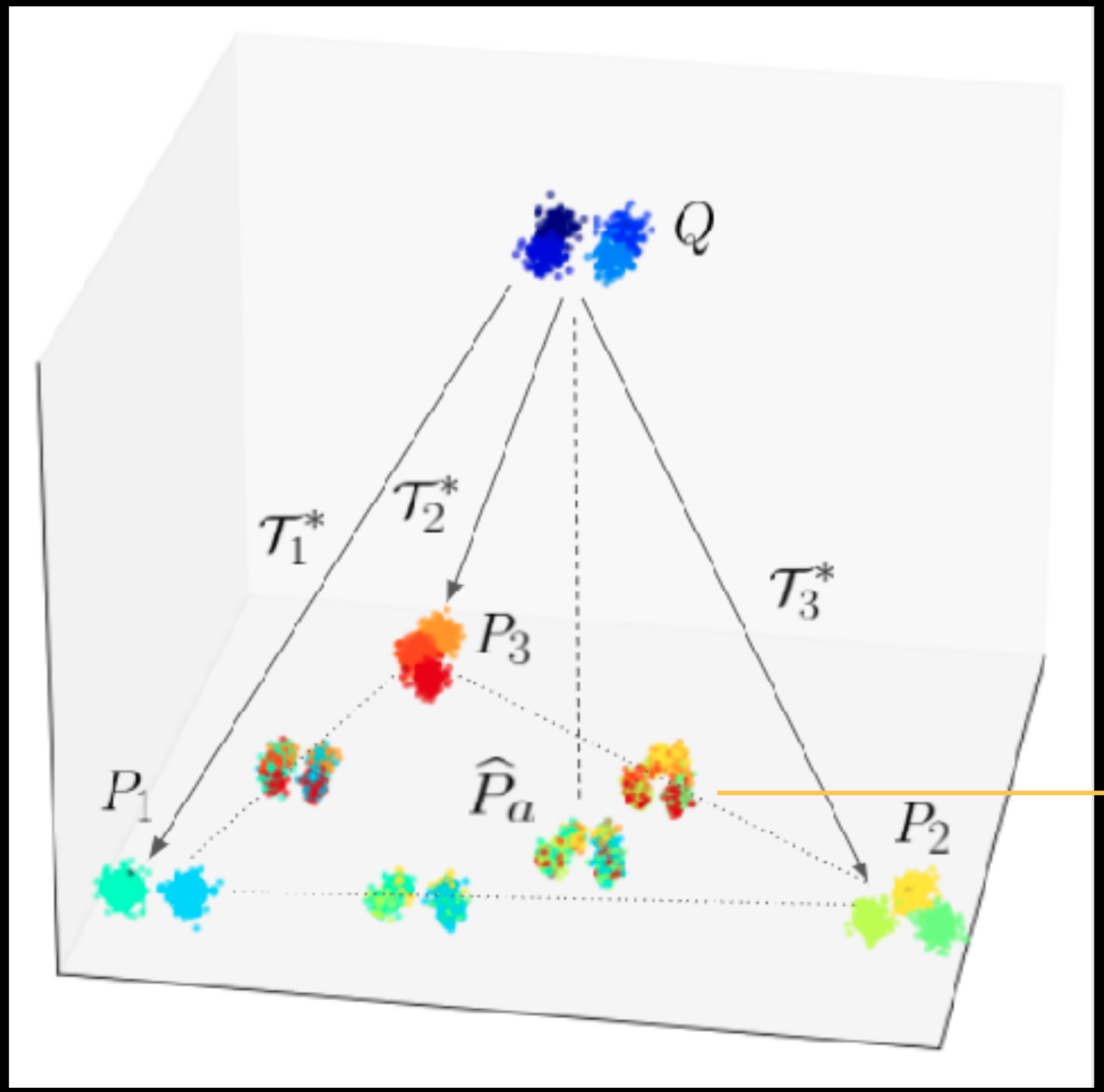
(AM & Fusi, 2020)



Can we 'fill gaps' in this manifold to generalize better?
Can we 'project' new datasets into this manifold?

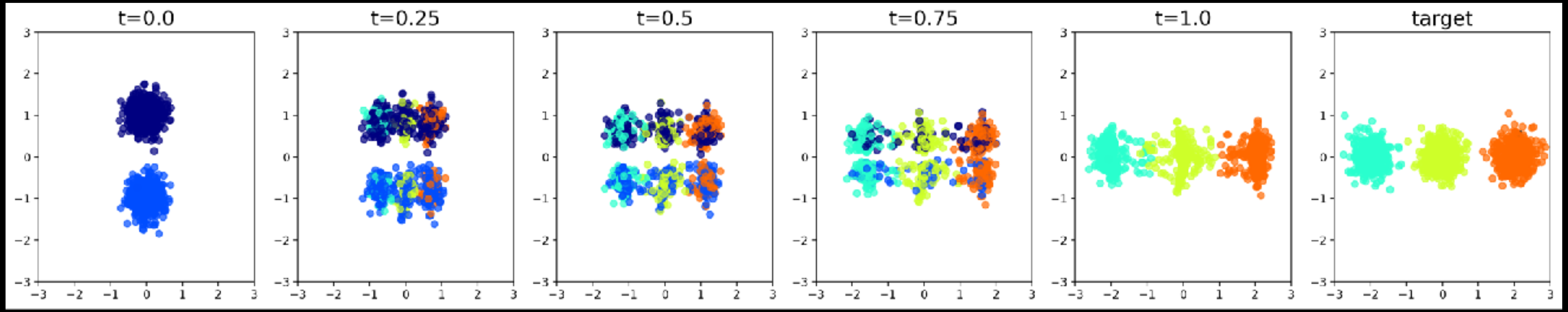
Synthesizing Datasets for Generalization

- Pre-training domains: $D_i \sim P_i$. Target domain $D_0 \sim Q$.
- Goal: generate the dataset in 'convex hull' of P_i closest to Q.
- Formalized as multi-dataset interpolations, using *generalized geodesics* from *OT theory*



Given interpolating weights $\mathbf{a} \in \Delta_m$, gen. geodesic with base Q is $P_a = \left(\sum_{i=1}^{m} a_i \mathcal{T}_i^*\right)_\sharp Q$

Example geodesic between two datasets



Our goal: find a minimizing Dist(P_a, Q). Can't be found directly, instead we solve:

$$a = \underset{a \in \Delta_m}{\operatorname{argmin}}\, \mathrm{W}_{2,Q}(P_a, Q) = \sum_{i=1}^{m} a_i \mathrm{W}_{2,Q}^2(P_i, Q) - \frac{1}{2} \sum_{i \neq j}^{m} a_i \mathrm{W}_{2,Q}^2(P_i, P_j)$$

Synthesizing Datasets for Generalization

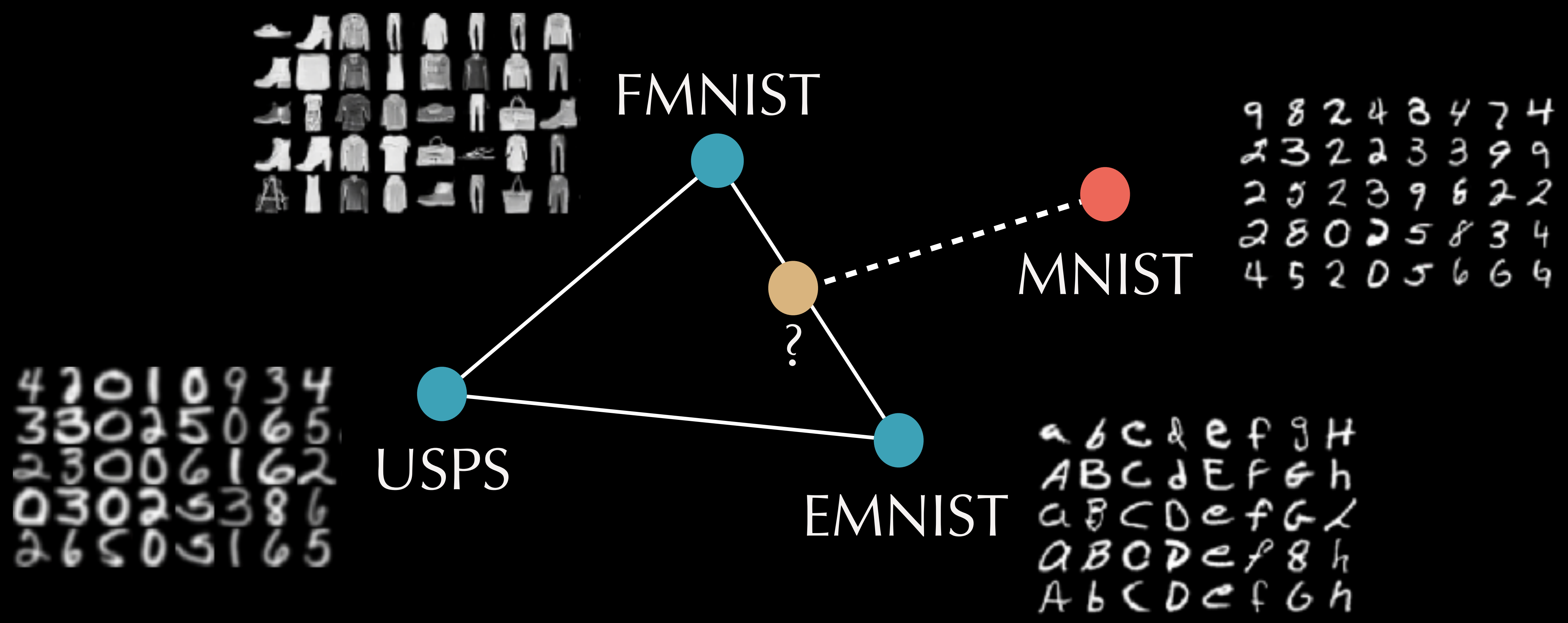
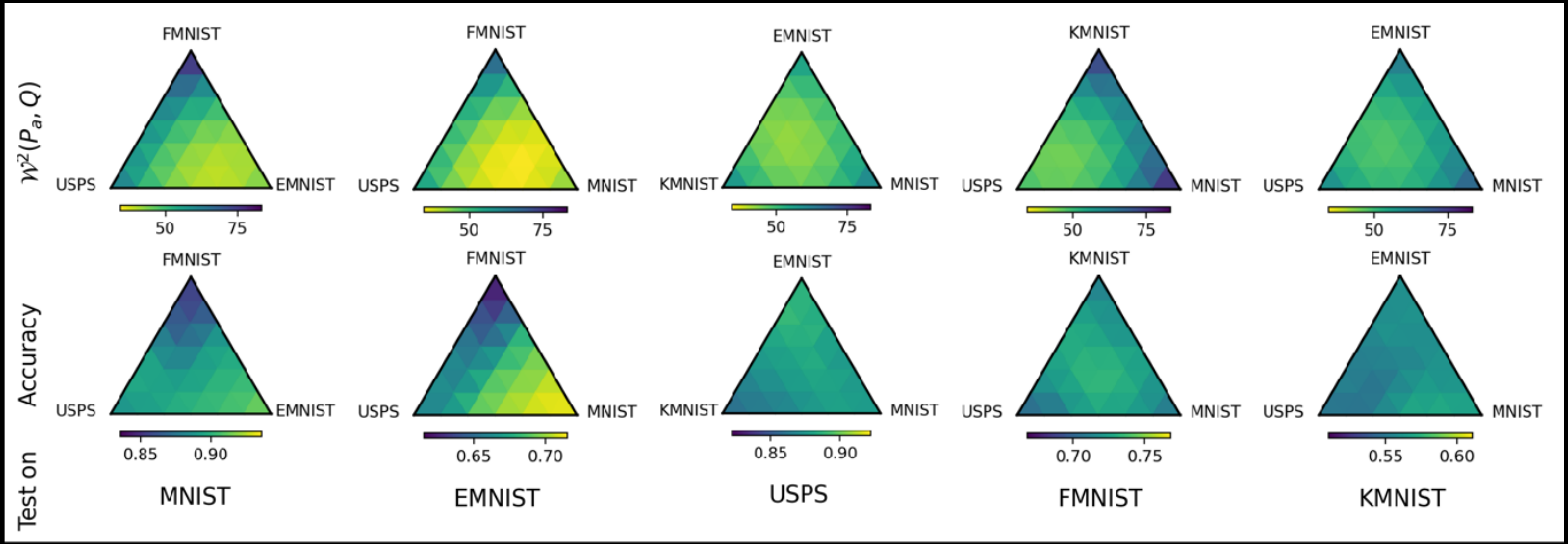


Table 1: **Pretraining on synthetic data.** Shown is 5-shot transfer accuracy (mean ± s.d. over 5 runs).

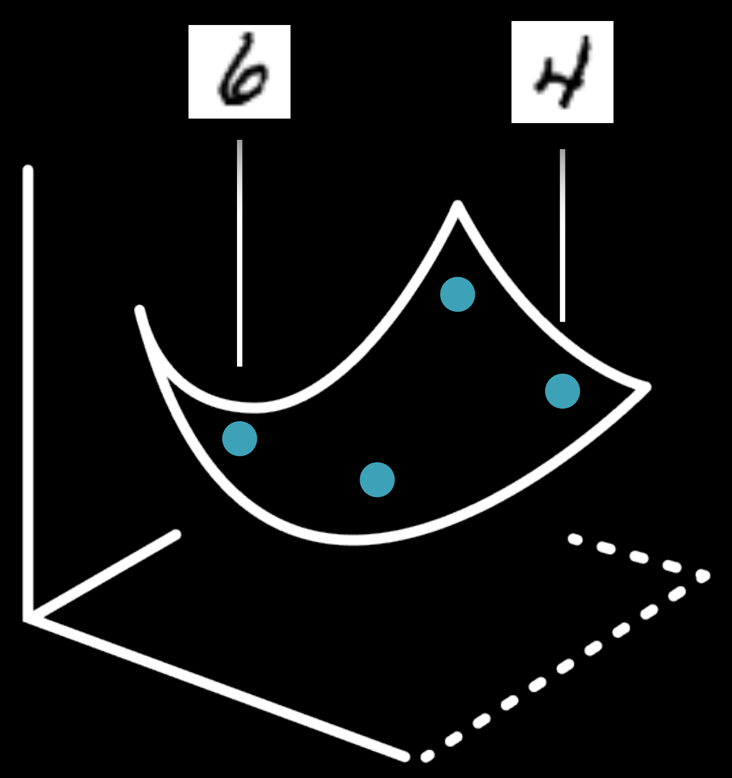
Methods	MNIST-M	MNIST	USPS	FMNIST	KMNIST	EMNIST
OTDD barycentric projection	**42.10±4.37**	**93.74±1.46**	86.01±1.50	**70.12±3.02**	**52.55±2.73**	**67.06±2.55**
OTDD neural map	40.06±4.75	88.78±3.85	83.80±1.60	70.02±2.59	50.32±3.10	65.32±1.80
Mixup	33.85±2.22	88.68±1.57	**88.61±2.00**	66.74±3.79	48.16±3.38	60.95±1.38
Train on few-shot dataset	19.10±3.57	72.80±3.10	80.73±2.07	60.50±3.07	41.67±2.11	53.60±1.18
1-NN on few-shot dataset	20.95±1.39	64.50±3.32	73.64±2.35	60.92±2.42	40.18±3.09	39.70±0.57

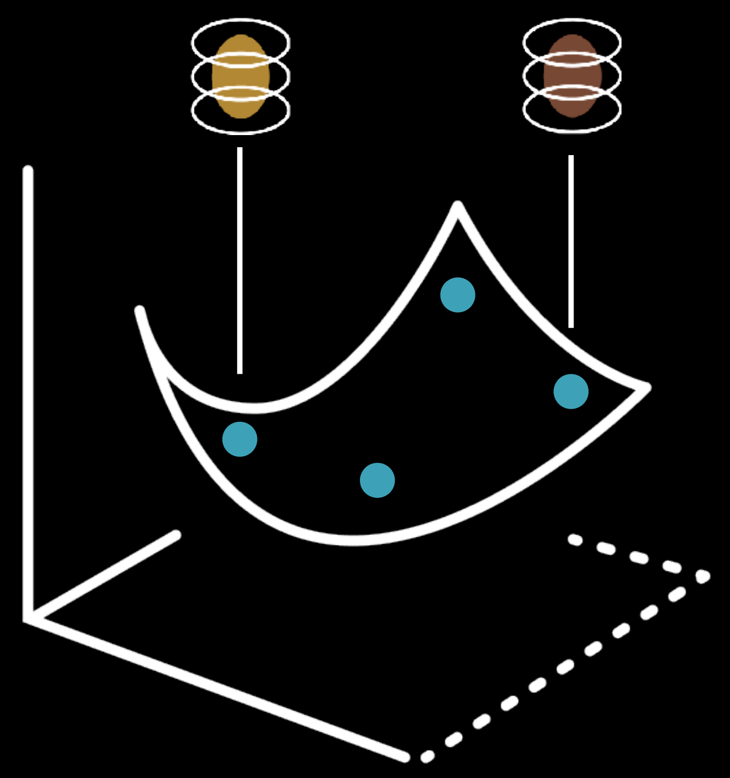
OT dist between synthetic dataset and target dataset

Performance of a model pre-trained on synthetic dataset, transferred to target



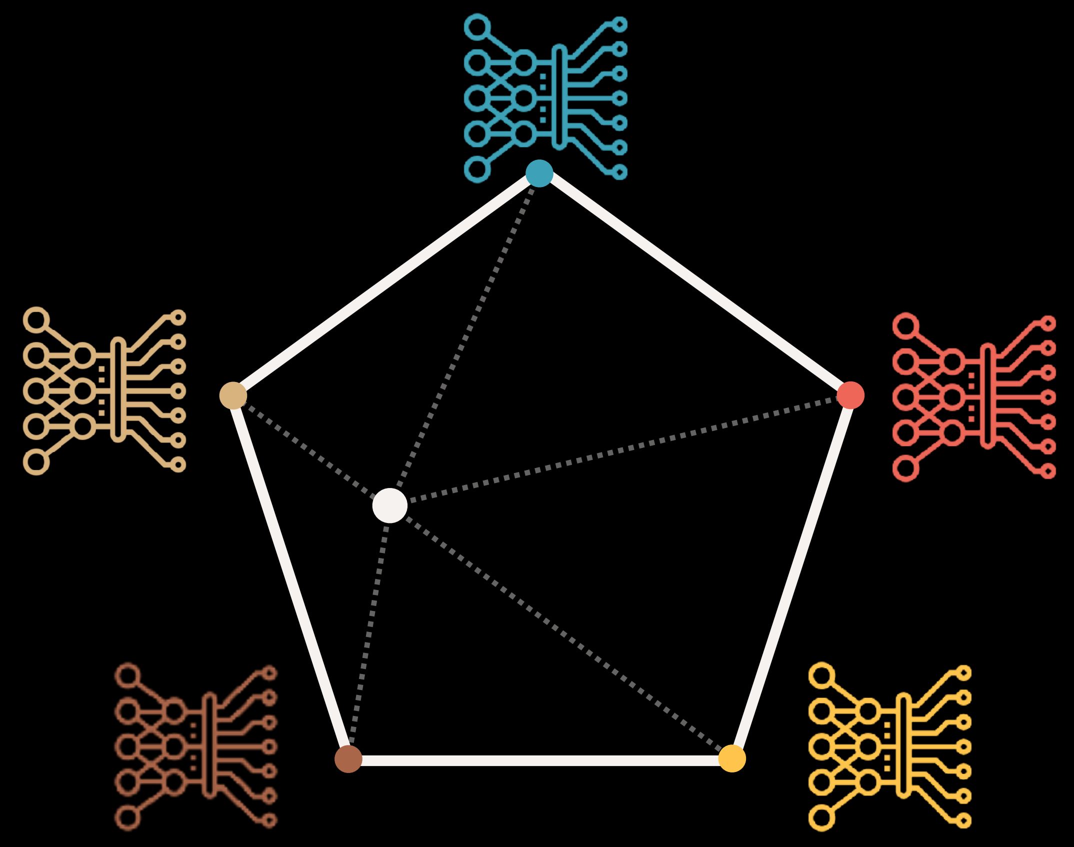
Takeaways


Classic ML:
Datapoint Space


Data-Centric ML:
Dataset Space

1. Datasets Can Be Embedded in Continuous Space
    - Treat datasets as (empirical) distributions
    - OT provides geometry for dataset space
    - Makes trajectories between domains meaningful
2. Geodesic Interpolation **Reveals the "In-Between" Domains**
    - Intermediate datasets aren't arbitrary blends, they follow principled geodesic paths.
    - Useful for analyzing domain shift and model behavior across transitions.
3. Generates **Synthetic Data in a Principled Way**
    - Can target specific regions of domain space for adaptation or robustness.
4. A Tool for Understanding and Designing Data, Not Just Augmenting
    - Helps visualize dataset geometry and relationships.
    - Provides a foundation for data-driven model shaping and evaluation.

Interpolating Models

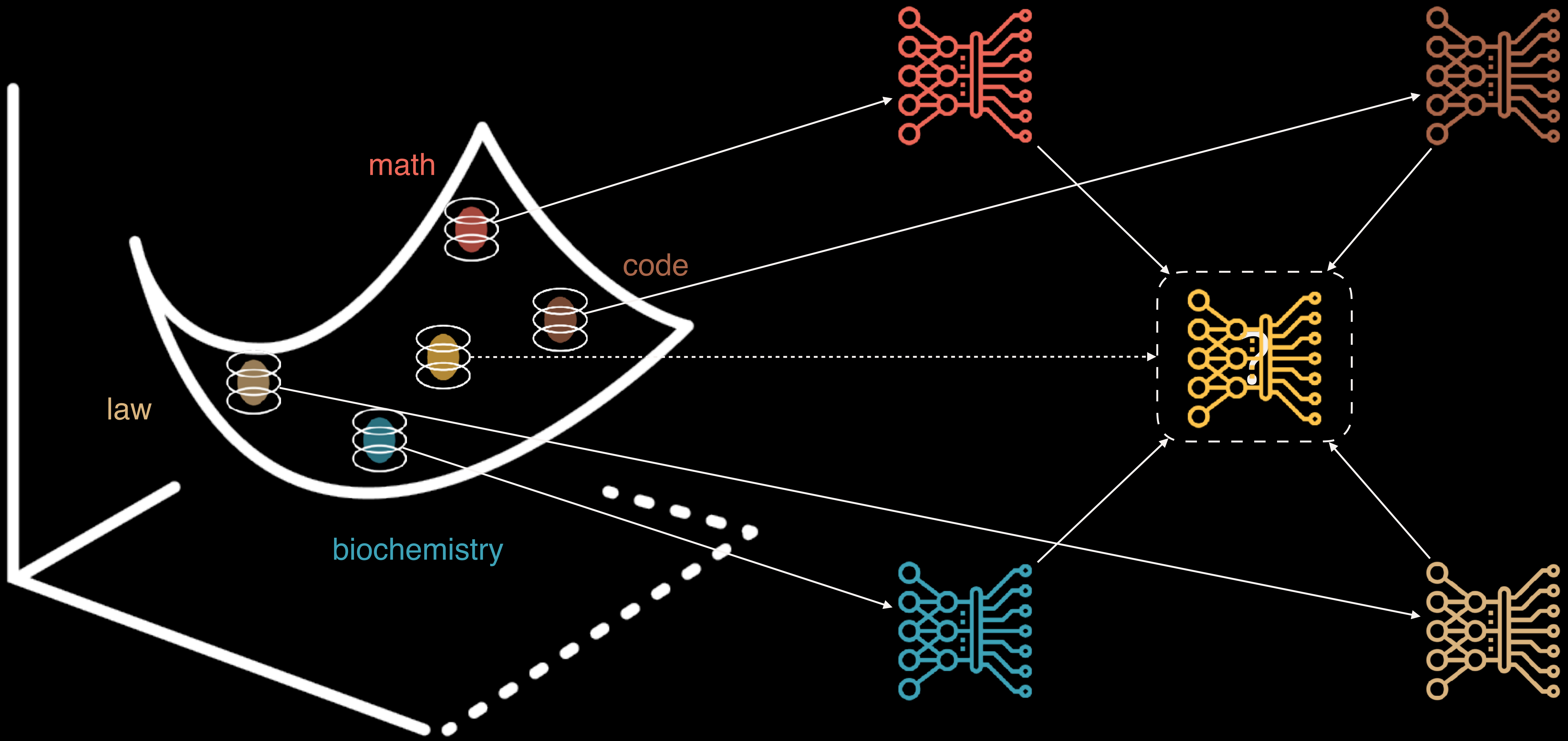


Continuous Language Model Interpolation for Dynamic an Controllable Text Generation

Kangaslahti & AM, TMLR 2025

Sara Kangaslahti

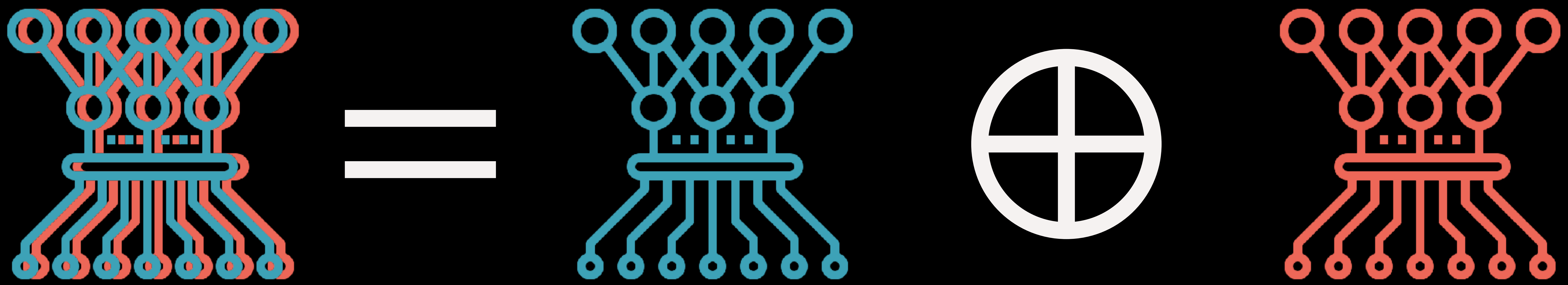
Cross-domain Interpolation



Can we “link” the dataset continuum with a model continuum?

Model Continuity via Weight Interpolation

Model “Soup” [Wortsman et al. 2022] is a simple linear weight interpolation:



$$\tilde{W}_l \triangleq \frac{1}{2}\left(W_l^a + W_l^b\right) \quad \forall l$$

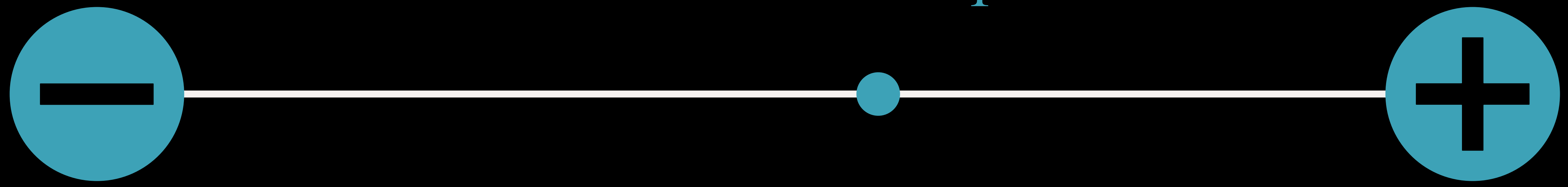
Continuous Model Interpolation

θ_0 : weights of base pretrained model

A, B : LoRA weight updates

i.e. Linear weight merging *alla* model soups (Wortsman et al. 2022)

$$\theta_{\alpha_1} = \theta_0 + \alpha_i \theta_{+i} + (1 - \alpha_i)\theta_{-i}$$



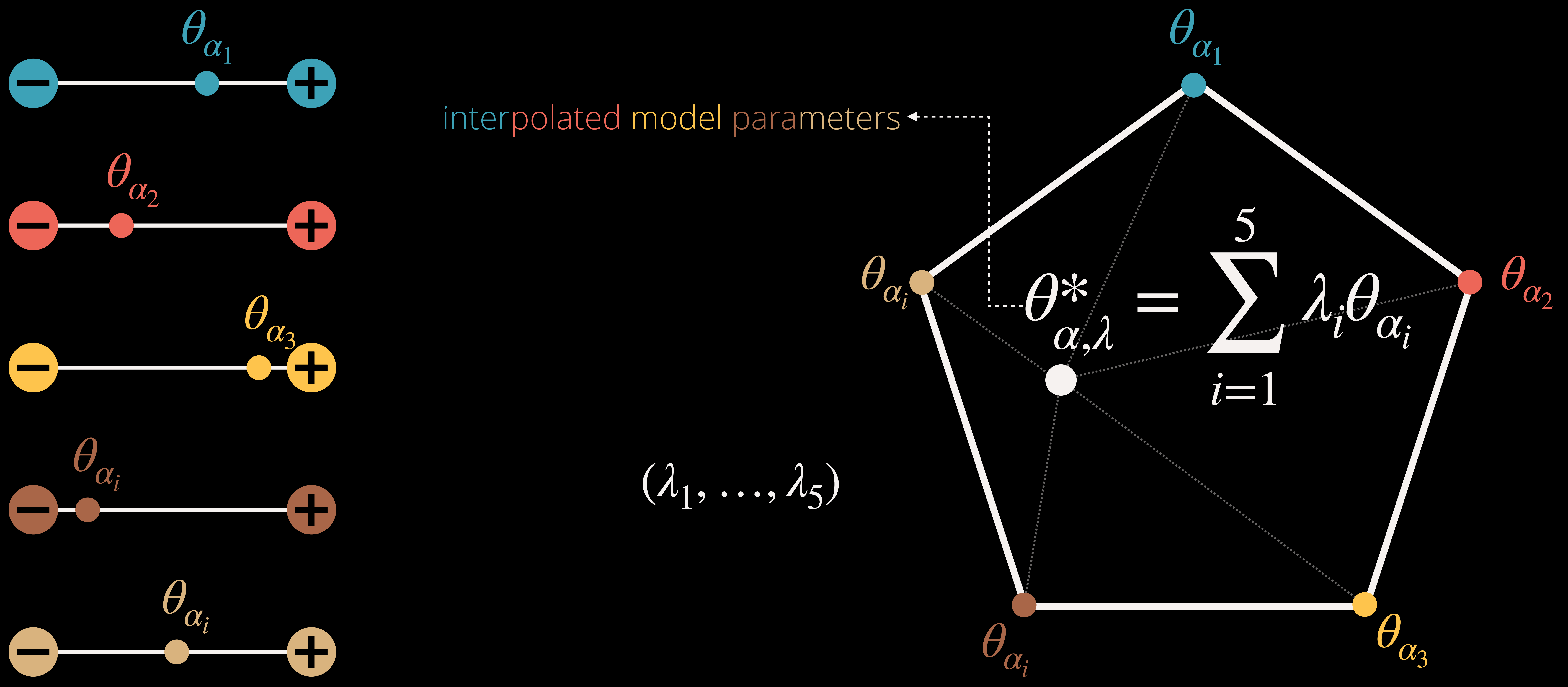
$$\theta_{-i} = \theta_0 + A_{-i} B_{-i}^{\top}$$

$$\theta_{+i} = \theta_0 + A_{+i} B_{+i}^{\top}$$

e.g. simple language

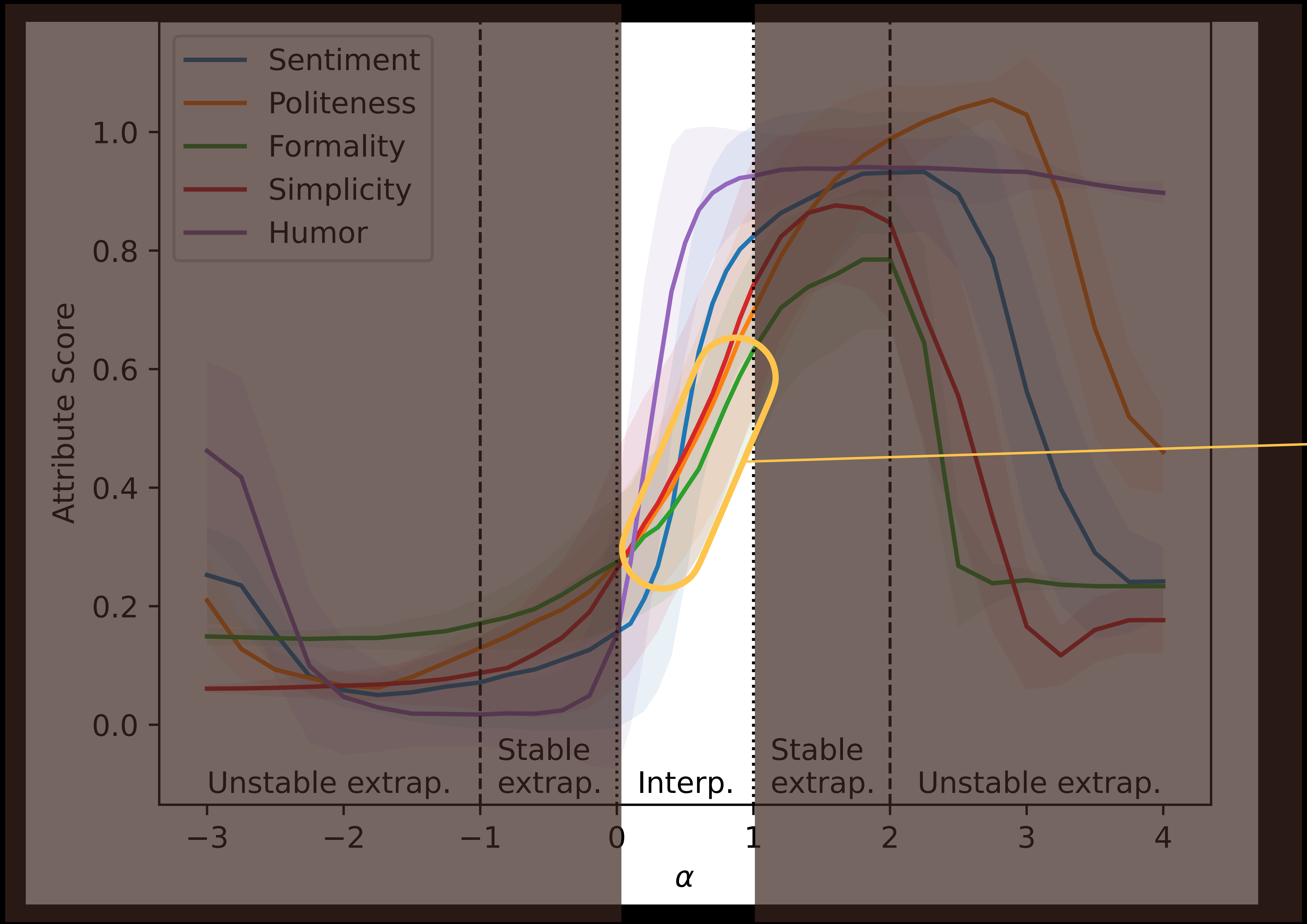
e.g. complex language

Continuous Model Interpolation

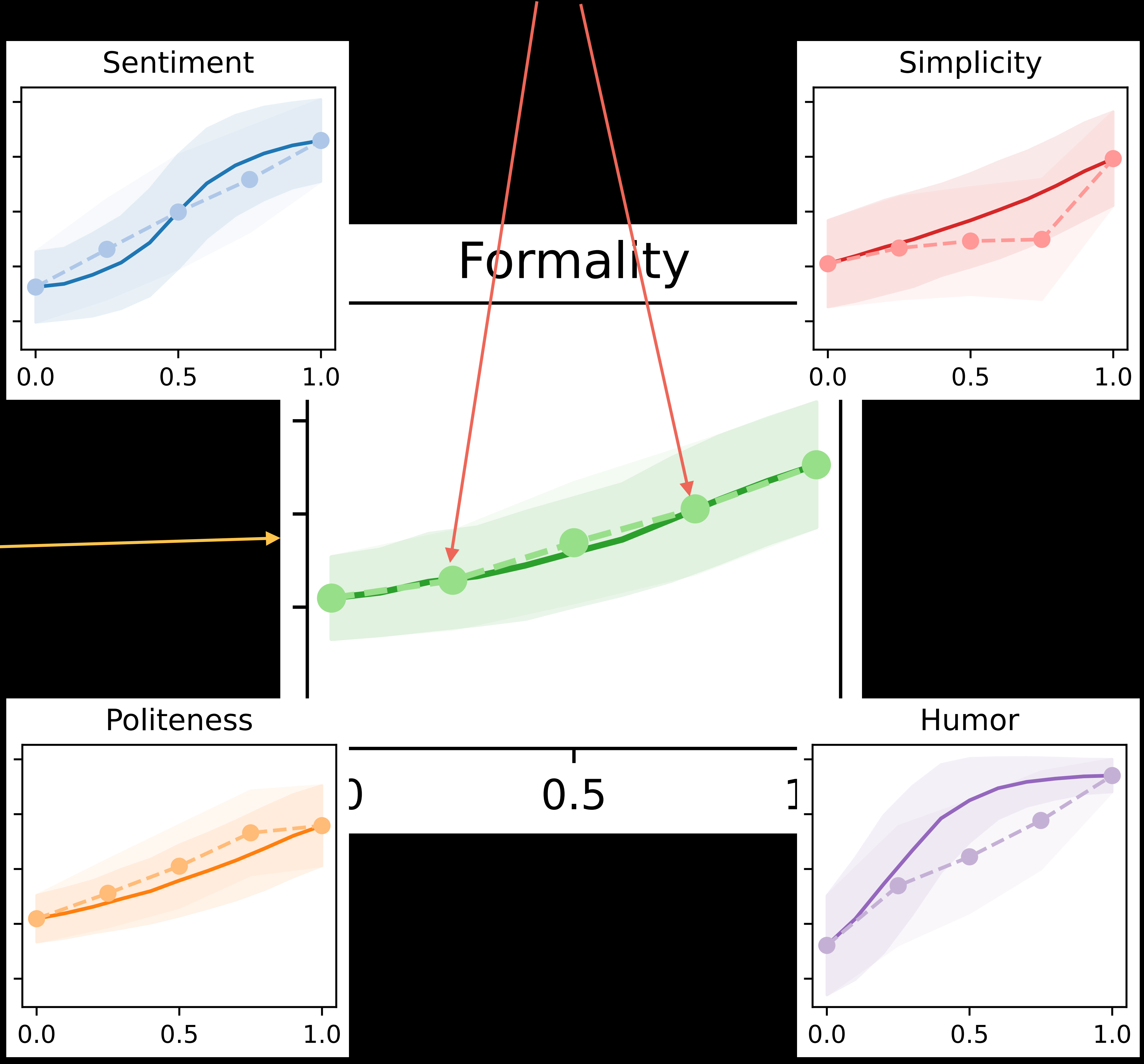


How faithful is the interpolation?

Single-Attribute Inter/Extra-polation



models **fine-tuned** on α-data mix

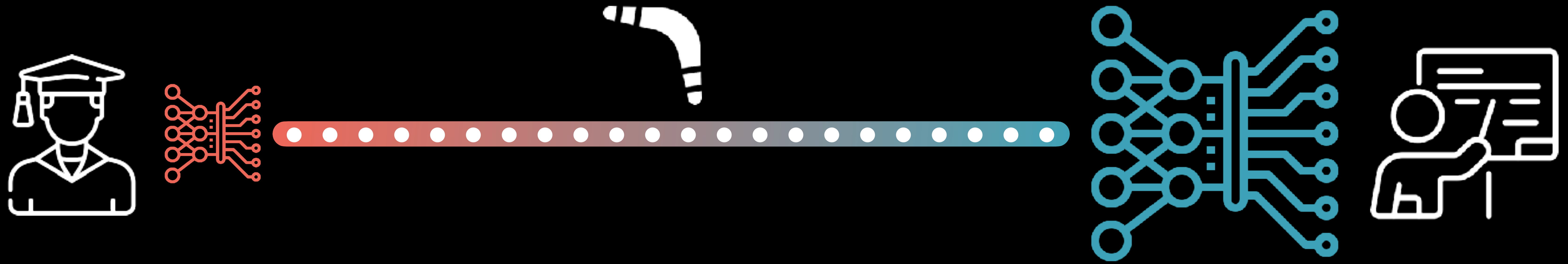


synthetic models **closely track** fine-tuned ones!

Sara Kangaslahti

Boomerang Distillation Enables Zero-Shot Model Size Interpolation

Kangaslahti, Geuter*, Nayak*, Fumero, Locatello, AM; arXiv 2025



Current landscape: pretrained model families

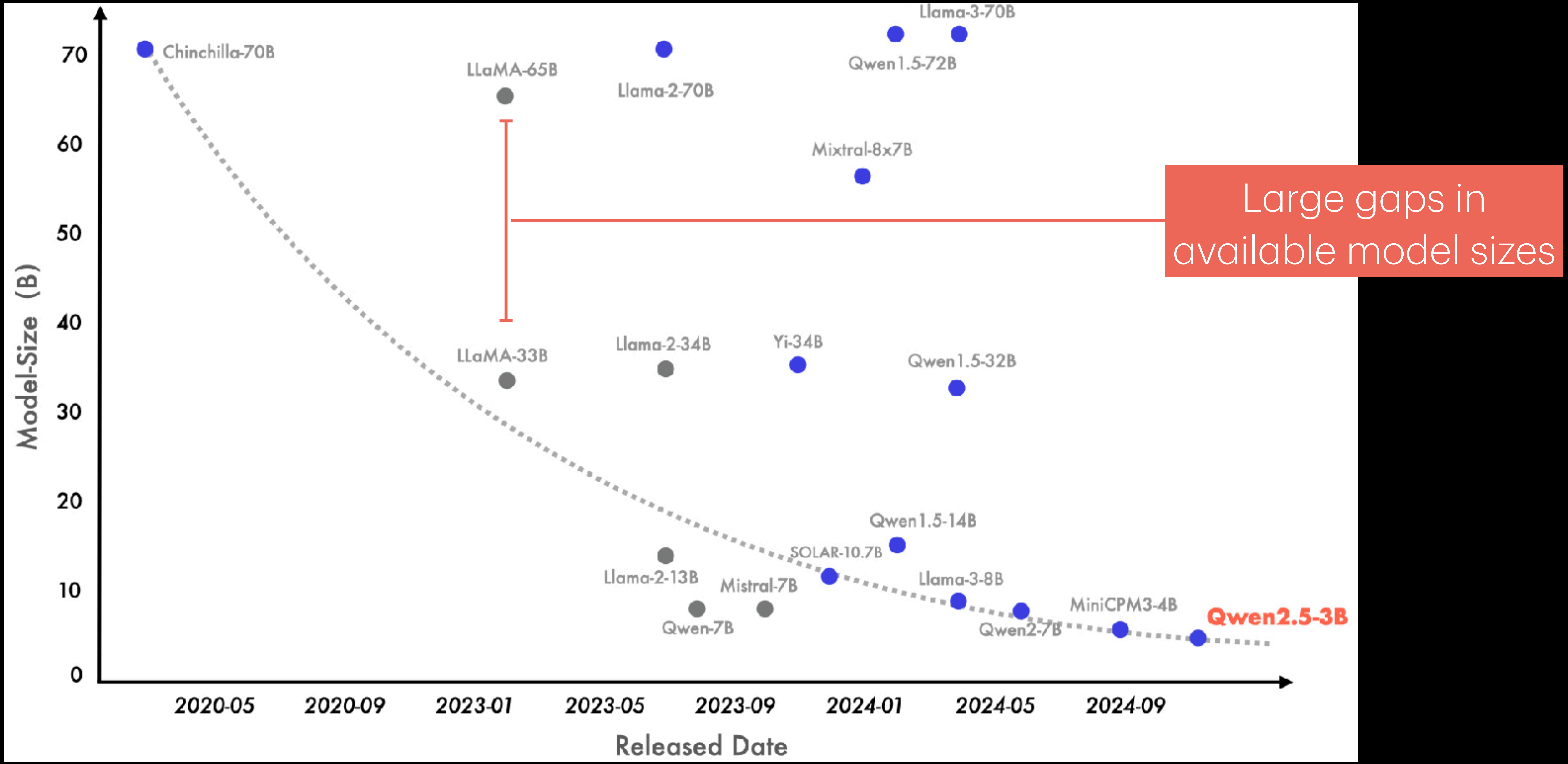


Figure credit: https://qwen.ai/research

Toward fine-grained model families

- We want **fine-grained model families** for:
  - **Adaptation**: maximizing performance under constrained settings and tailoring to resources of individual users (eg max out GPU utilization)
  - **Science of LLMs**: e.g., better resolution for scaling laws / behavior emergence

But pre-training **too computationally expensive** for large models!

- What about distillation?

Current distillation approaches save compute
but still require **independently distill-training each model**

- Can we bypass that?

Yes! We identify *boomerang distillation*, a phenomenon that creates a set of fine-grained intermediate models **without additional training**!

Boomerang distillation

Interpolated model

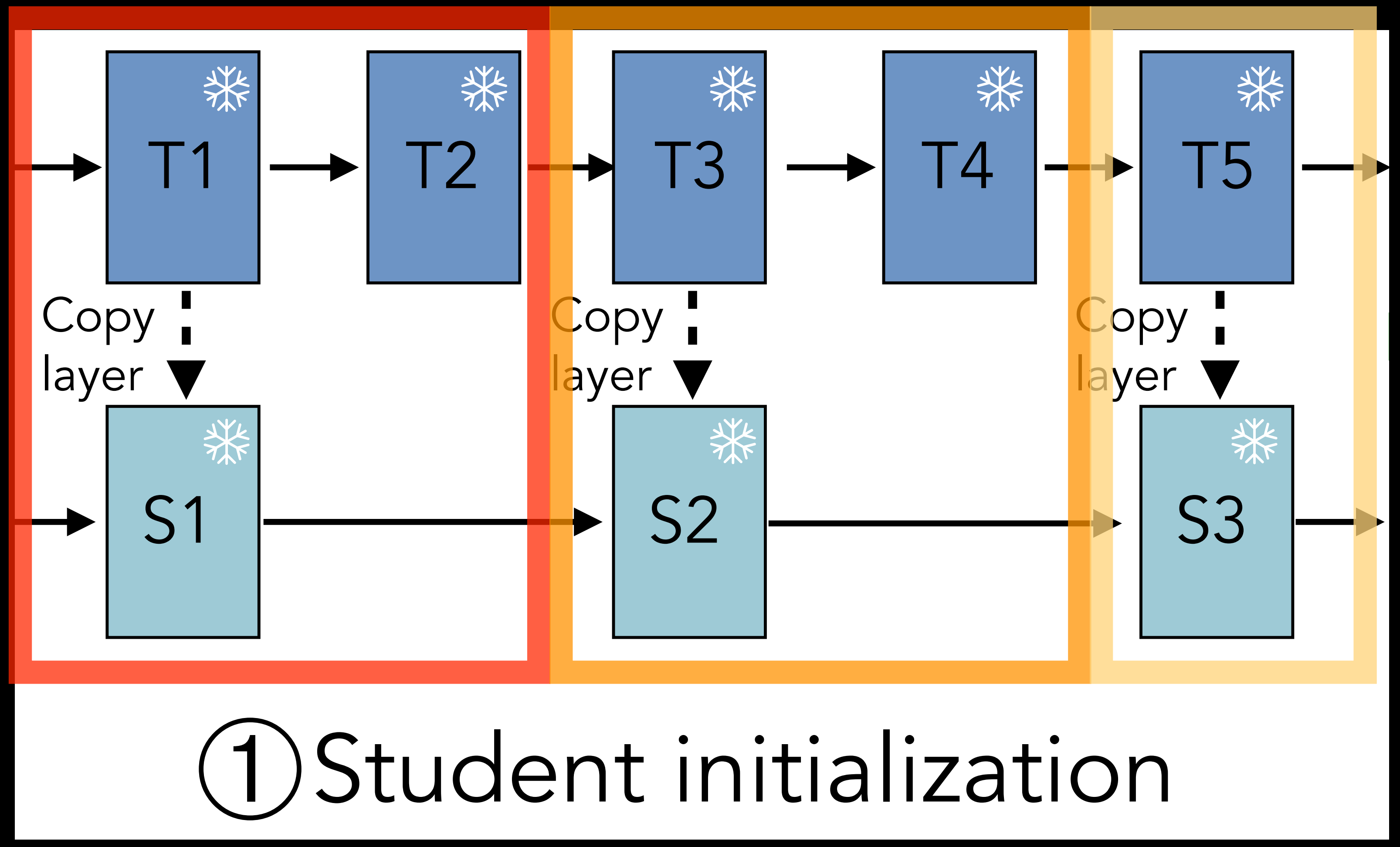
Teacher model

Add back teacher layers to student

Prune and distill

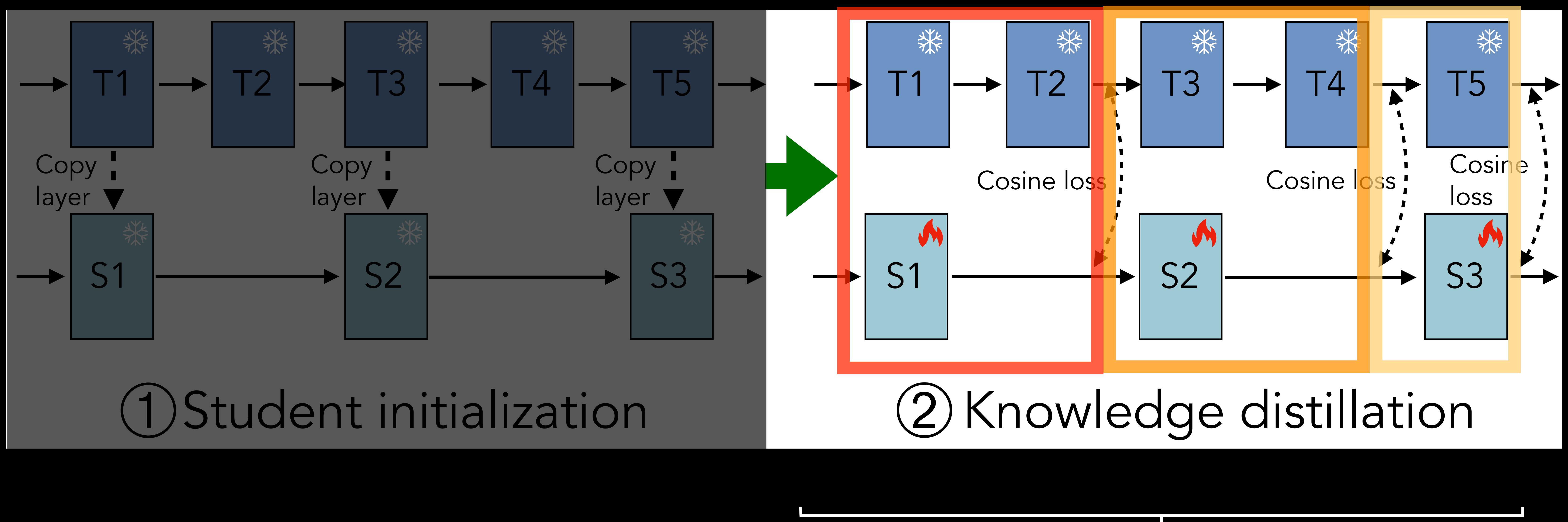
Model size

Boomerang 🪃 distillation ⚗️



Every student layer corresponds to a block of teacher layers

Boomerang 🪃 distillation ⚗️



Enforce alignment between student layer and teacher block outputs during distillation

Boomerang 🪃 distillation ⚗️

Interpolate by patching any student layers with their corresponding teacher blocks

Boomerang distillation

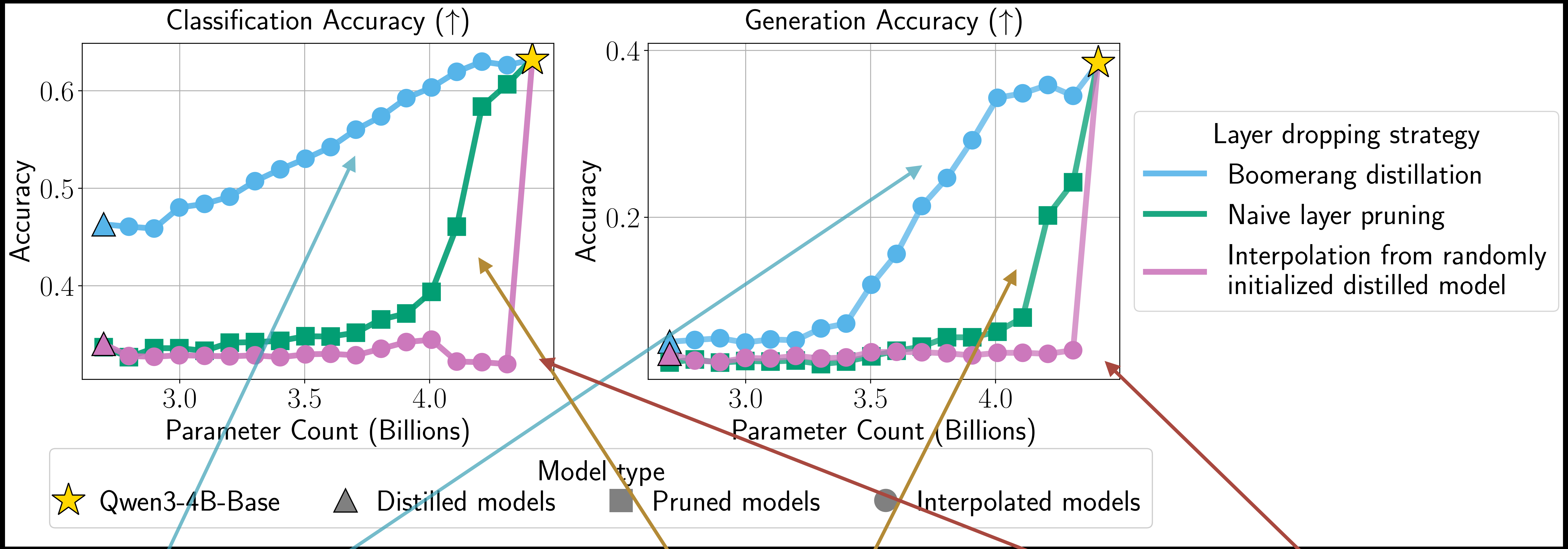
Note: Only first (smallest) model is trained with distillation objective, all other intermediate models are materialized **without additional training!**

Experimental setup

- **Teacher model**: Qwen3-4B-Base
- **Student model:**
    - initialized by copying every other layer and last layer from teacher
    - Distilled on 2.1B tokens of The Pile deduplicated (Gao et al., 2021)
- **Evaluation:**
    - Full spectrum of interpolated models evaluated on 10 classification datasets, 3 generation datasets, and Wikitext perplexity (Merity et al., 2017).

🪃 ⚗️ yields smooth performance interpolation

Setup: comparing boomerang distillation to naive pruning and random initialization

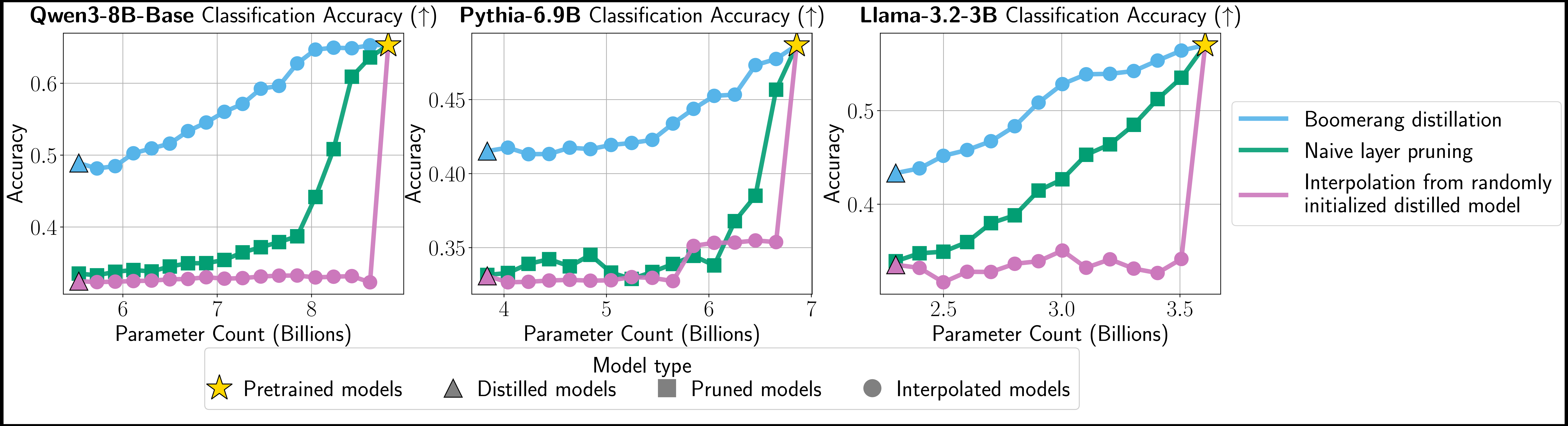


Boomerang distillation yields smooth performance interpolation behavior

Naive pruning (without distillation) fails at interpolation

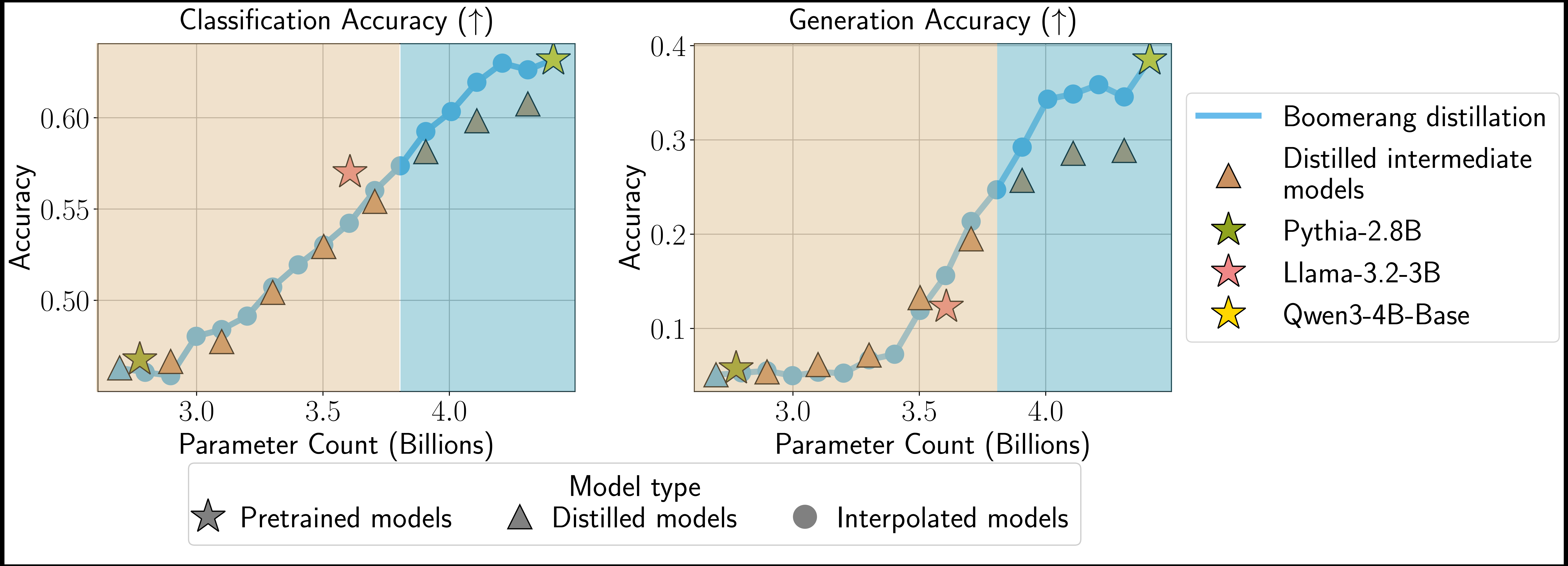
Distillation alone (without teacher initialization) cannot interpolate either

🪃⚗️ occurs across model families and sizes



🪃 🔮 matches intermediate distilled + pertained models

Setup: comparing boomerang distillation to standard distillation and pretrained models

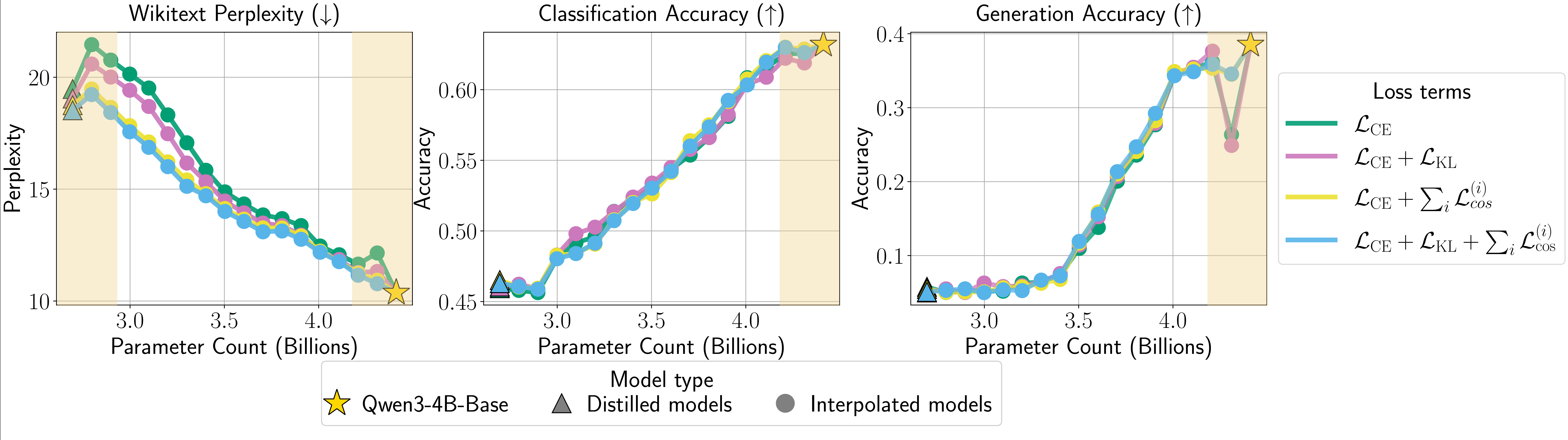


Interpolated models have similar performance to distilled models at small sizes

Interpolated models outperform larger distilled models

Layer alignment loss is crucial for stability

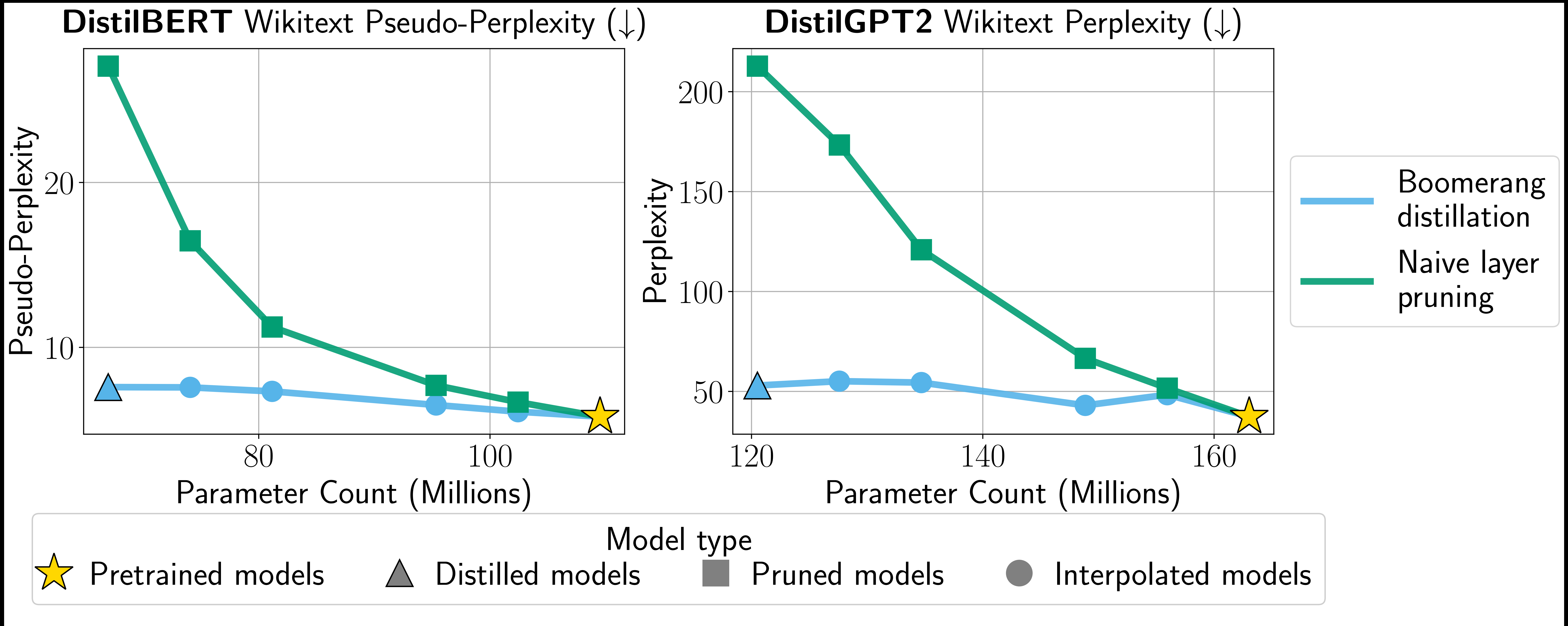
Setup: testing different combinations of loss terms during distillation



Alignment loss provides stability at interpolation extremes

🪃 ⚗️ works for off-the-shelf already-distilled models

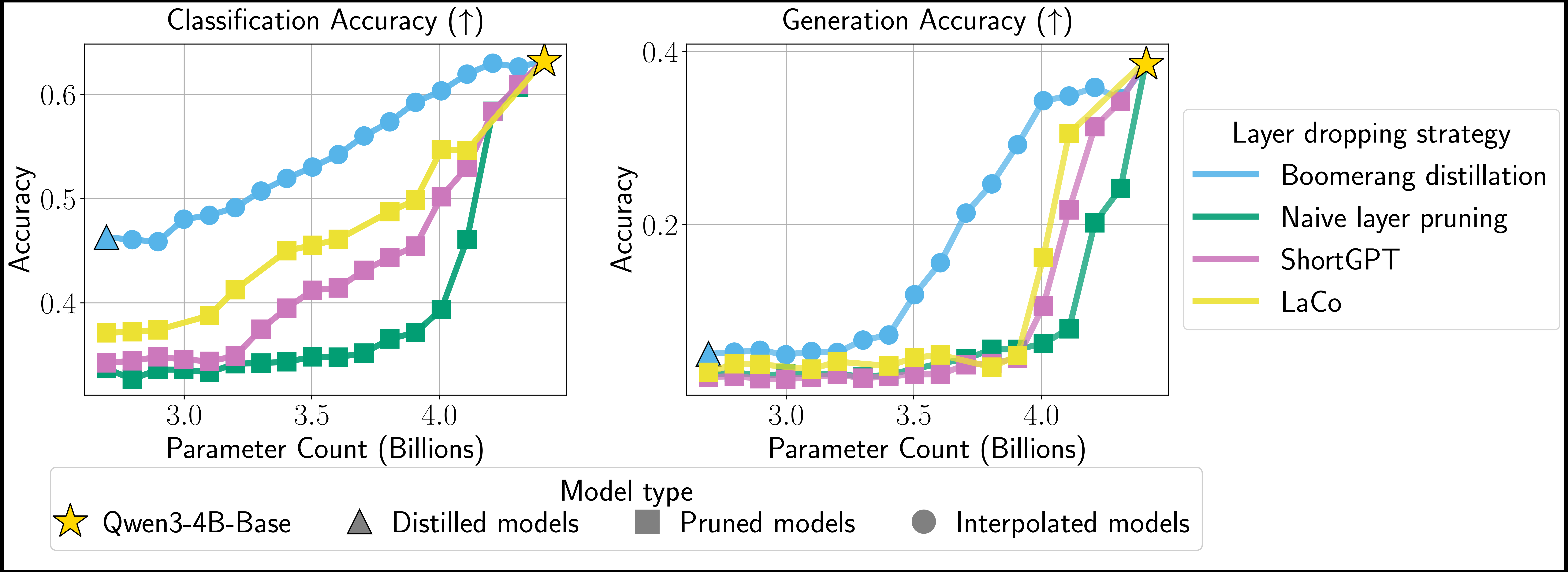
Setup: interpolation between DistilBERT and BERT and DistilGPT2 and GPT2



Boomerang distillation works here too!

🪃 ⚗️ significantly outperforms layer-dropping methods

Setup: comparison between interpolation and layer pruning methods

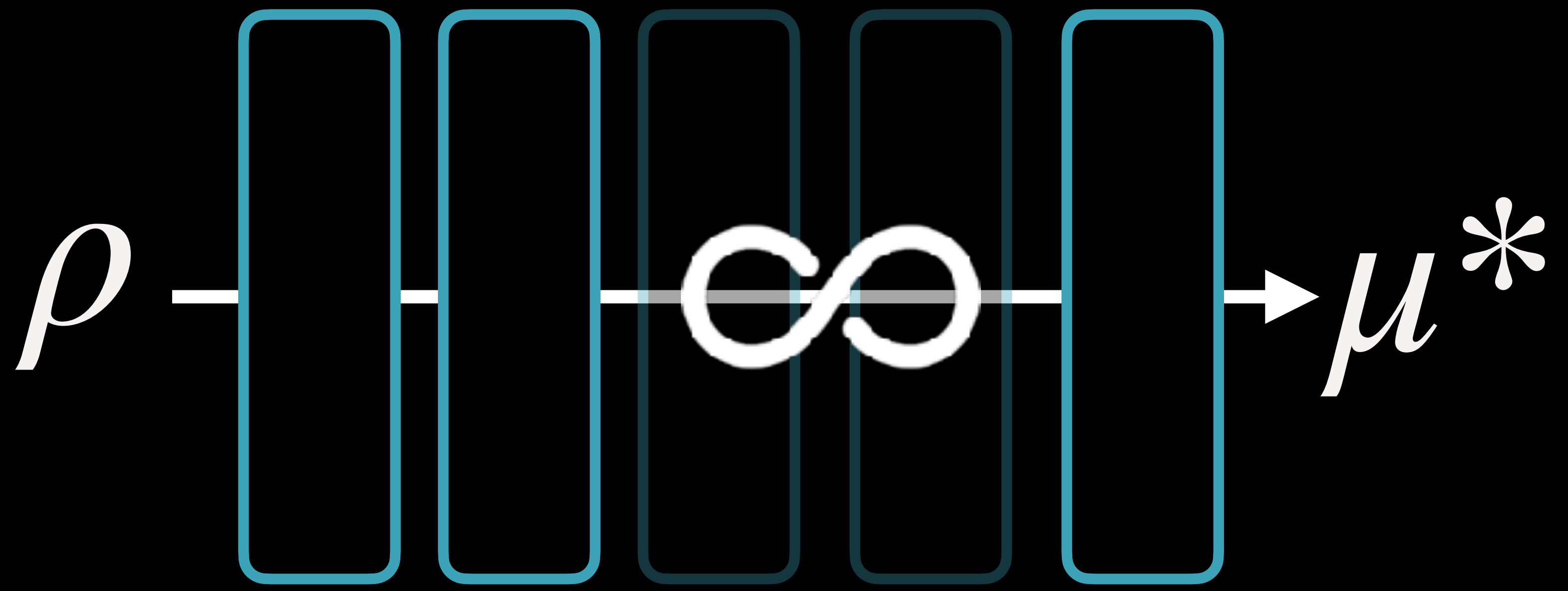


Boomerang distillation significantly outperforms baseline layer pruning methods, **especially in the high-compression regime**

🪃⚗️: Takeaways

- Boomerang distillation yields models that smoothly interpolate in size and performance between a given student and teacher model **without any additional training**
- Student model needs to be:
  - Initialized from the teacher with layer pruning
  - Distilled after pruning to recover performance and alignment
- Boomerang distillation outperforms pruning and **matches individually distilled models**

Interpolating Compute



DDEQs: Distributional Deep Equilibrium Models through Wasserstein Gradient Flows

Jonathan Geuter, Clément Bonet, Anna Korba, DAM

AISTATS 2025

Jonathan Geuter

Motivation: Adaptive Test-Time Compute

- **Fixed architecture = fixed compute**
  - Standard architectures perform a preset number of layers/iterations.
  - Same cost whether the input is trivial or difficult.
- But….. not all inputs are created equal!
  - Some examples need many refinement steps; others need almost none.
  - A **discrete depth forces one-size-fits-all computation**.
- Rigid Compute–Accuracy Tradeoffs
  - Need higher accuracy? Need to train a (new) larger/deeper model.
  - Need faster inference? Need to train a (new) smaller/shallower model.
  - **No smooth way** to trade off speed/accuracy properties on the fly!

Background: Deep Equilibrium Models

Deep Equilibrium Models

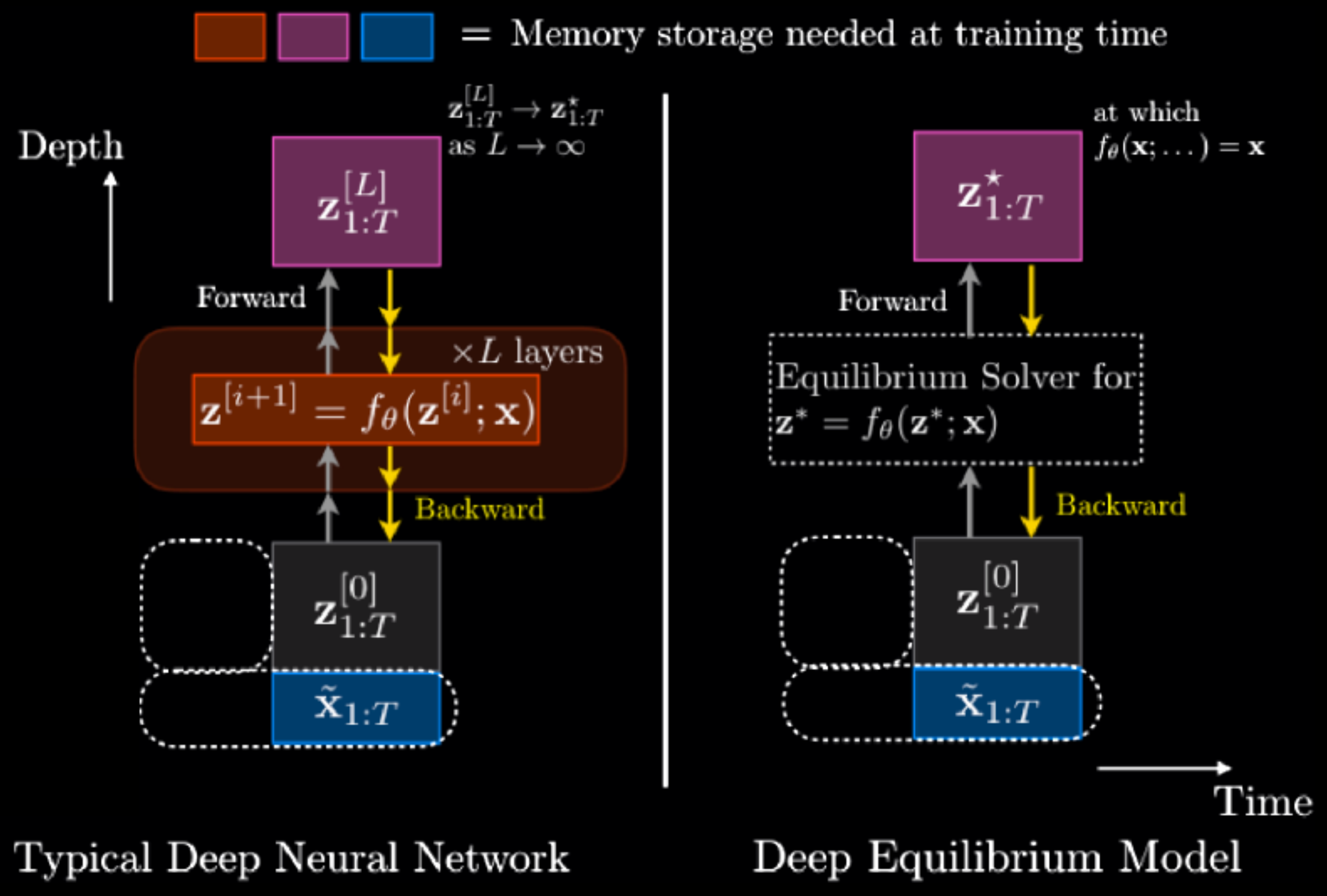
Shaojie Bai
Carnegie Mellon University

J. Zico Kolter
Carnegie Mellon University
Bosch Center for AI

Vladlen Koltun
Intel Labs

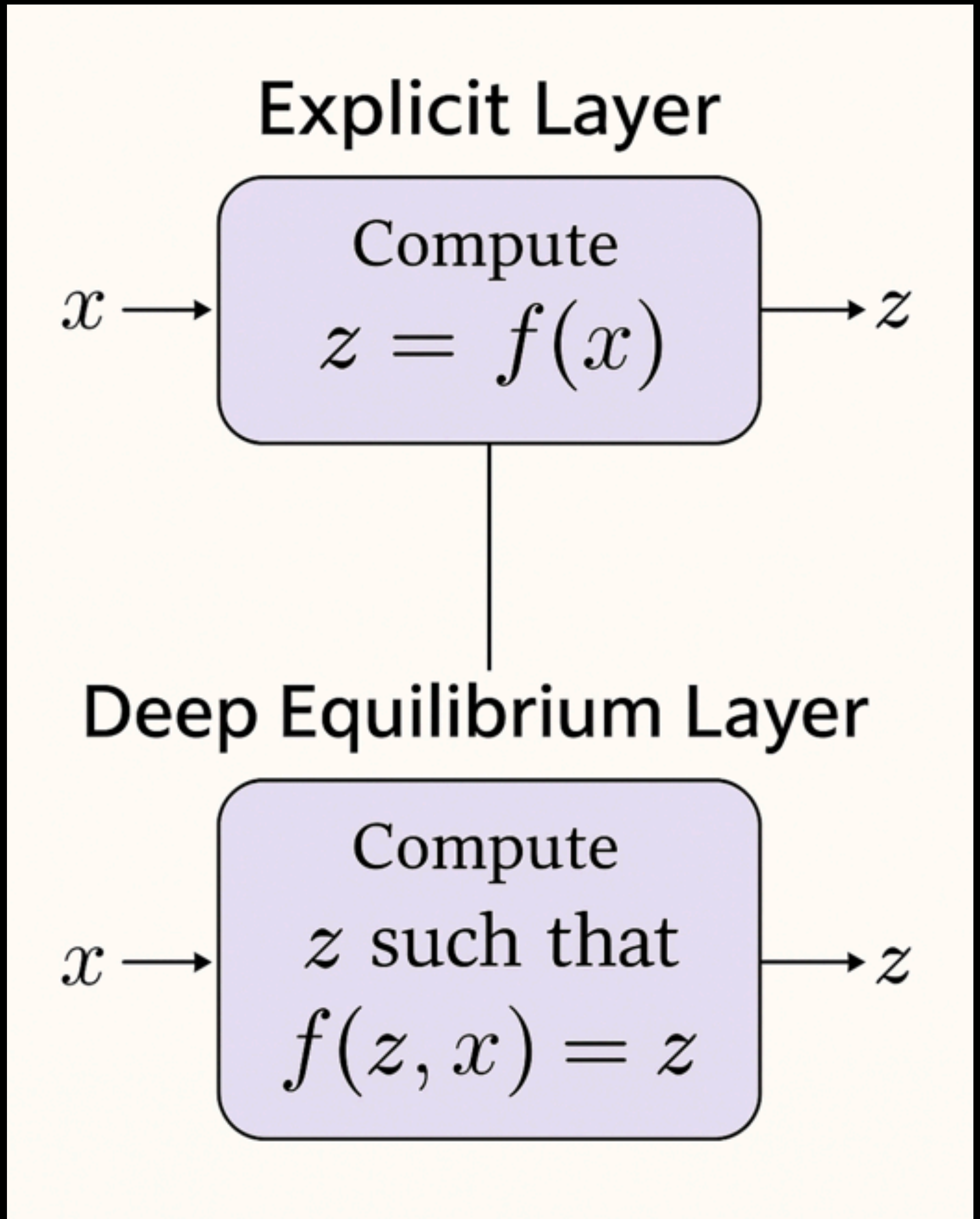
Abstract

We present a new approach to modeling sequential data: the deep equilibrium model (DEQ). Motivated by an observation that the hidden layers of many existing deep sequence models converge towards some fixed point, we propose the DEQ approach that *directly* finds these equilibrium points via root-finding. Such a method is equivalent to running an *infinite* depth (weight-tied) feedforward network,



Bai, Kolter, and Koltun 2019

Background: Deep Equilibrium Models

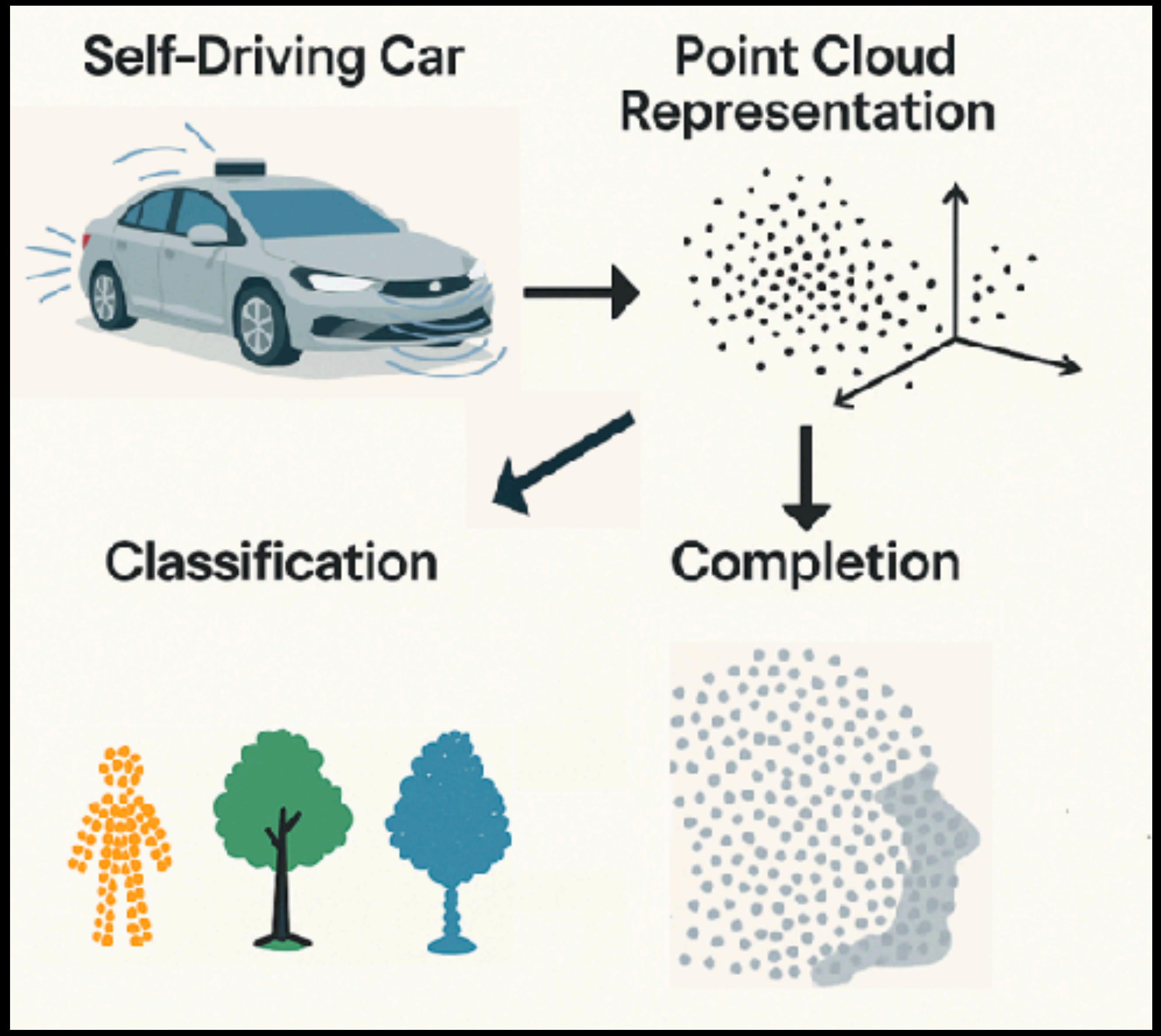


- Similar to weight-tied models, but Instead of iterating $z_{k+1} = f_\theta(z_k, x)$ K times, (e.g. fixed-depth), a DEQ layer solves $z^* = f_\theta(z^*, x)$
- Fixed point z^* is the output/representation of the network
- Can be viewed as neural net **infinite-depth limit**
- Training/Inference:
  - Forward pass = find the fixed point (via root-finding algo, to desired tolerance)
  - Backward pass = one implicit gradient step (uses implicit f'n theorem), **no need to store intermediate activations**!
- Makes **compute a quasi-continuous quantity / adaptive**:
  - More solver iterations ⇒ more refinement
  - Fewer iterations ⇒ faster but approximate inference.

Bai, Kolter, and Koltun 2019

Beyond Fixed-Sized Inputs

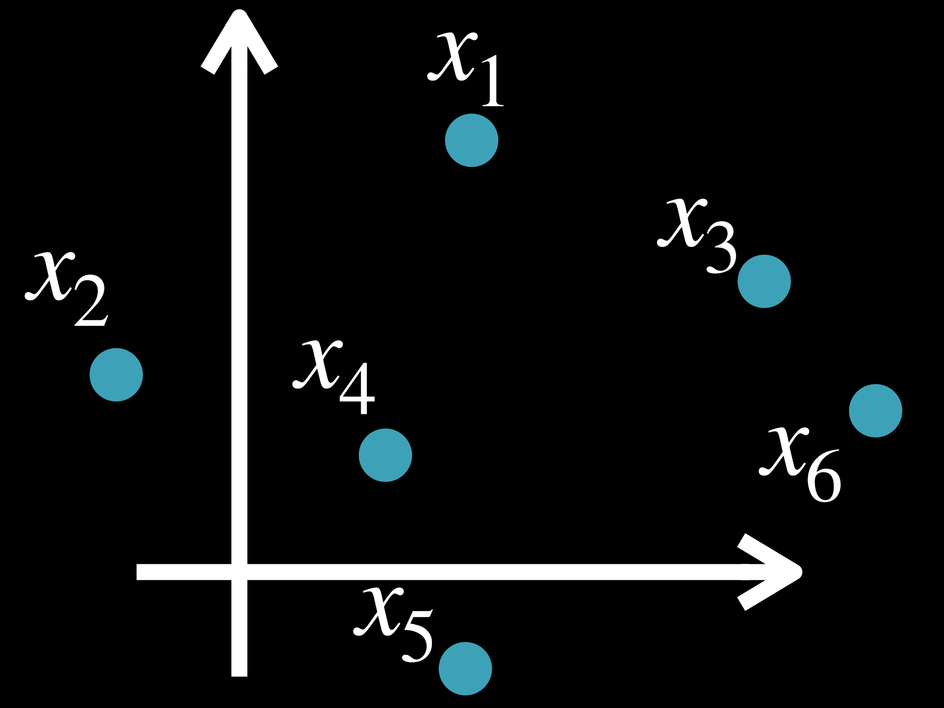
- Standard DEQs assume vector/matrix inputs/outputs of **fixed shape/dimensionality**
- But many real-world inputs are sets or distributions
  - e.g., point clouds, graphs, multi-particle systems
  - **variable size, permutation invariant**
- Need a model that operates **directly on sets/distributions**
  - representations should adapt to variable-size inputs
  - computation should be size-agnostic and permutation-invariant



To generalize DEQs beyond fixed-size inputs, we **lift them from finite dimensions to distributional spaces**!

Distributional DEQs: Formulation

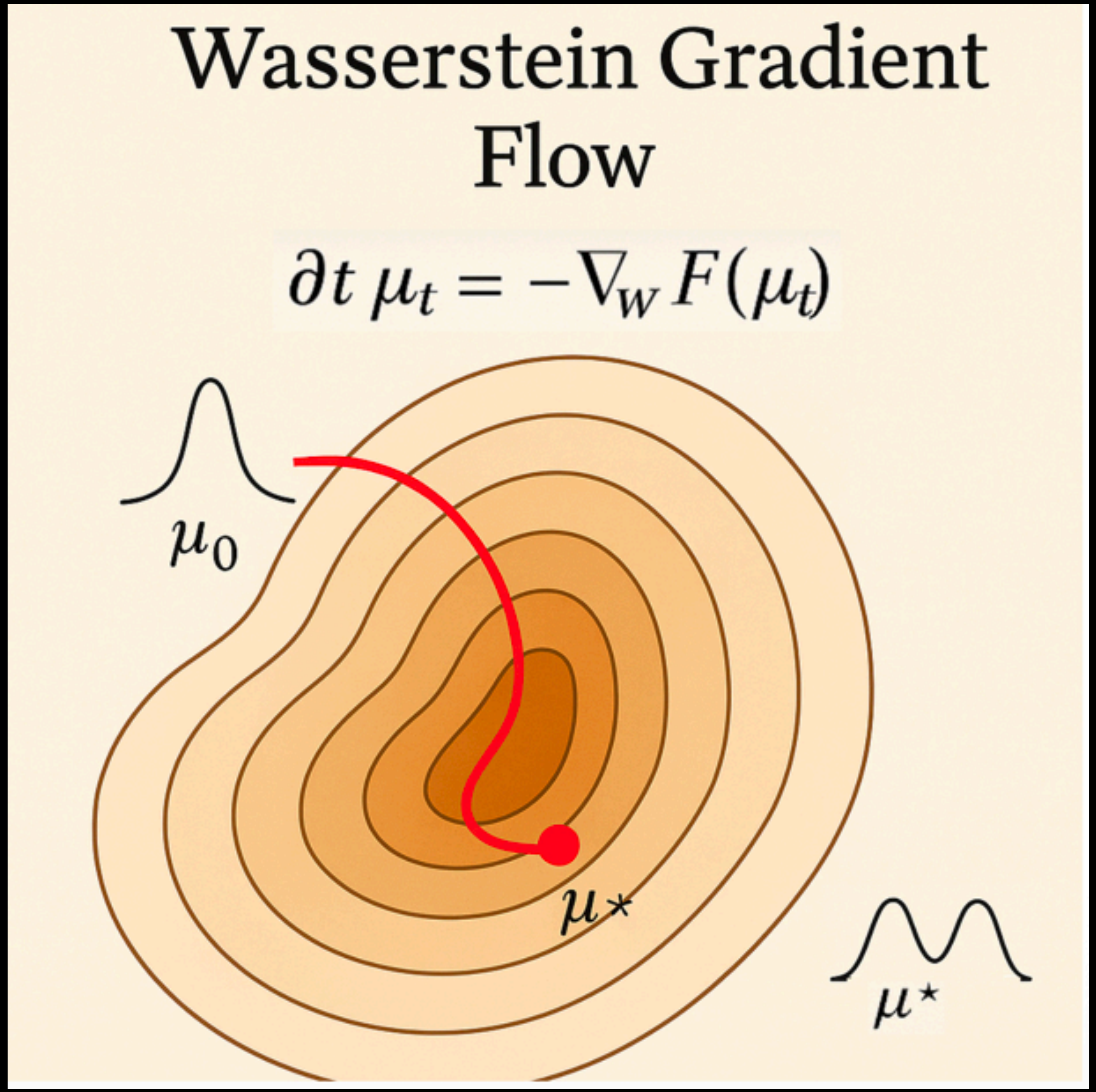
- Point clouds are (discrete) probability measures!
- Ingredients:
  - Data: $(\rho, y) \in \mathcal{P}_2(\mathbb{R}^d) \times \mathcal{Y} \sim P_{\text{data}}$
  - Latent measure: $\mu \in \mathcal{P}_2(\mathbb{R}^p)$
  - DEQ layer: $F_\theta(\,\cdot\,, \rho) : \mathcal{P}_2(\mathbb{R}^p) \to \mathcal{P}_2(\mathbb{R}^p)$
  - "Post-processor": $h_\theta : \mathcal{P}_2(\mathbb{R}^p) \to \mathcal{Y}$



$\rho = \sum_1^n \mathbf{a}_i \delta_{x_i}, \mathbf{a} \in \Delta_n$

- Given fixed point $\mu^*_{\theta,\rho} = F_\theta(\mu^*_{\theta,\rho}, \rho)$, define loss as: $\mathcal{L}(\theta) := \mathbb{E}_{\rho,y\sim P_{\text{data}}}\left[\ell\left(h_\theta(\mu^*_{\theta,\rho}), y\right)\right]$
- **How do we find the fixed point?**
  - Wasserstein Gradient Flow!
- **How to compute gradient**?
  - Implicit Gradient Theorem!

Wasserstein Gradient Flows

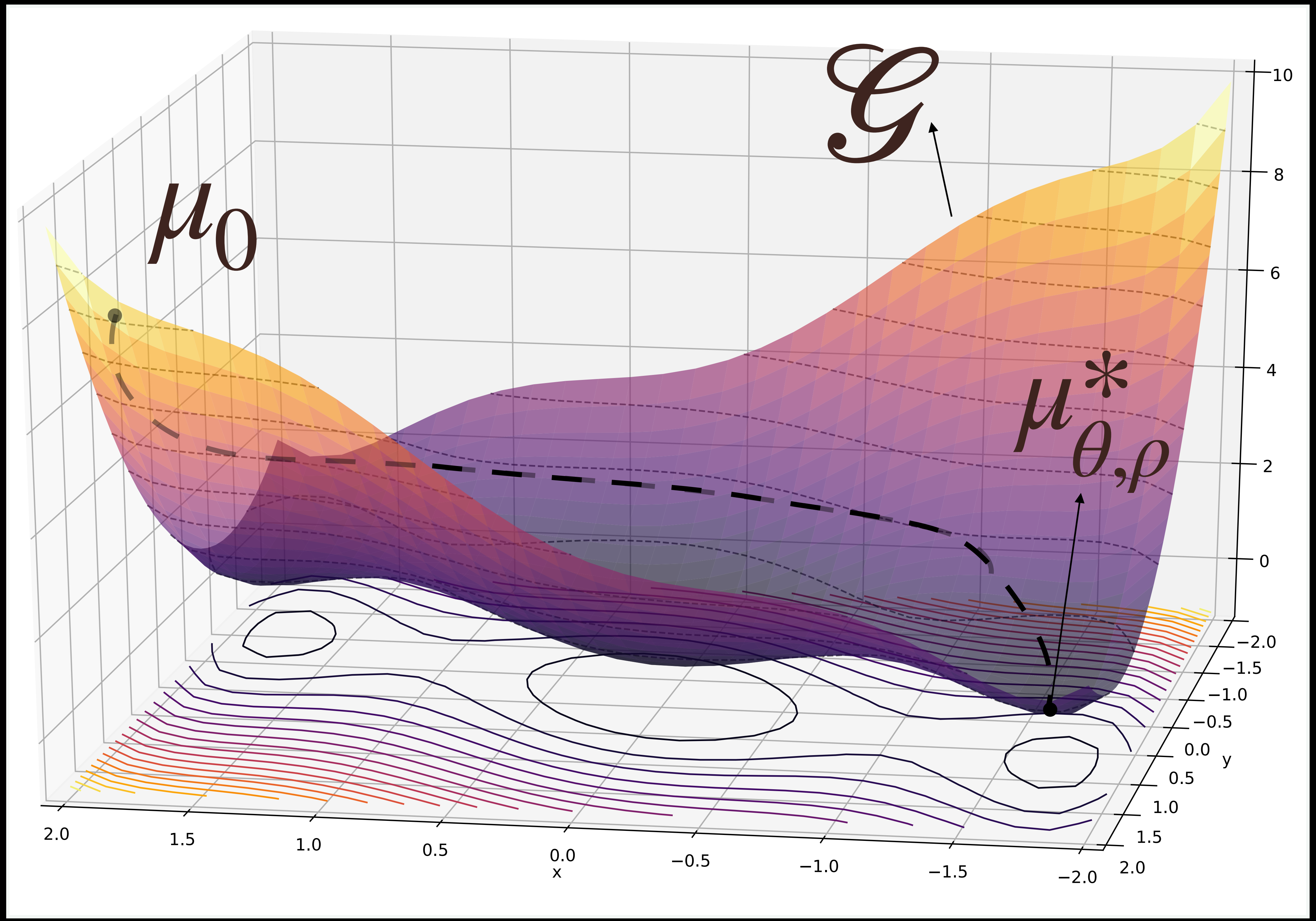


- Gradient descent: $x_{n+1} = x_n - \gamma \nabla f(x_n)$, $\gamma > 0$ stepsize
- Taking limit $\gamma \to 0$ leads to an ordinary differential equation called the **gradient flow:** $\dot{x}(t) = -\nabla f(x(t))$
- This is **a curve** $x(t)$ **in** $\mathbb{R}^d$ that starts at $x(0) = x_0$ and moves at each instant in the direction of steepest descent of f
- Gradient flows can be defined over distributions via the Wasserstein gradient: $\nabla_W \triangleq \nabla \cdot (\rho \nabla \frac{\partial(\cdot)}{\partial \rho})$

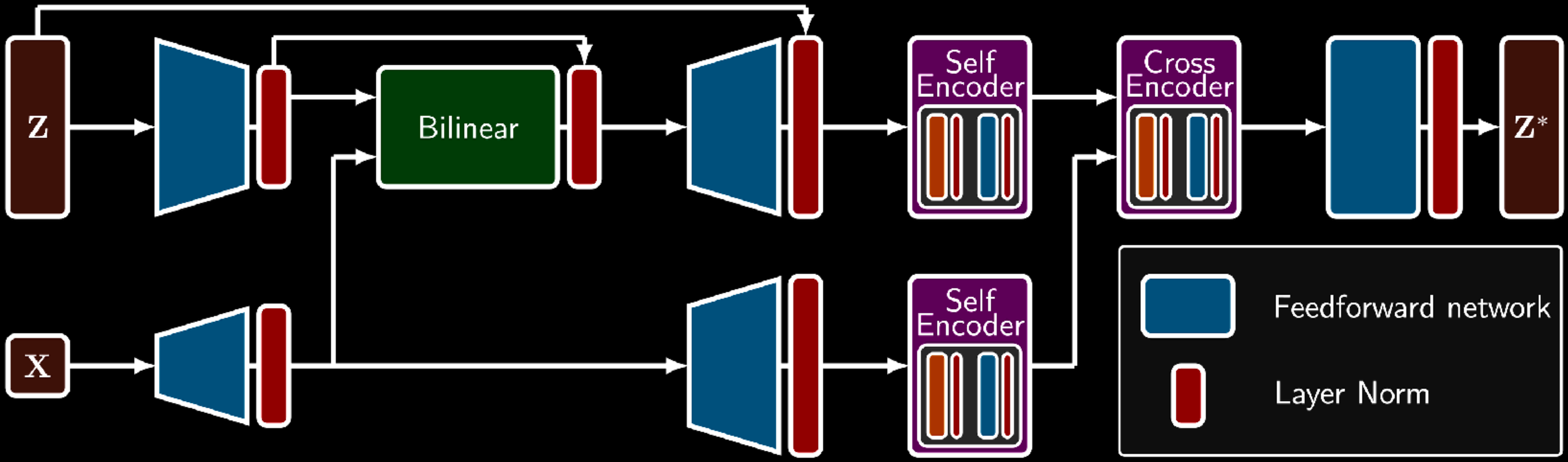
(e.g., Santambrogio 2016; Ambrosio, Gigli, Savare 2008)

Distributional Fixed Points

- Goal: Find the fixed point $\mu^*_{\theta,\rho} = F_\theta(\mu^*_{\theta,\rho}, \rho)$
- Can operationalize this as: $\mathrm{MMD}(\mu, F_\theta(\mu, \rho)) = 0$
- Let $\mathcal{G}(\mu) = \frac{1}{2}\mathrm{MMD}^2(\mu, F_\theta(\mu, \rho))$
- Finding fixed point of $F_\theta \Rightarrow$ following (time-discretized) Wasserstein gradient flow of $\mathcal{G}$

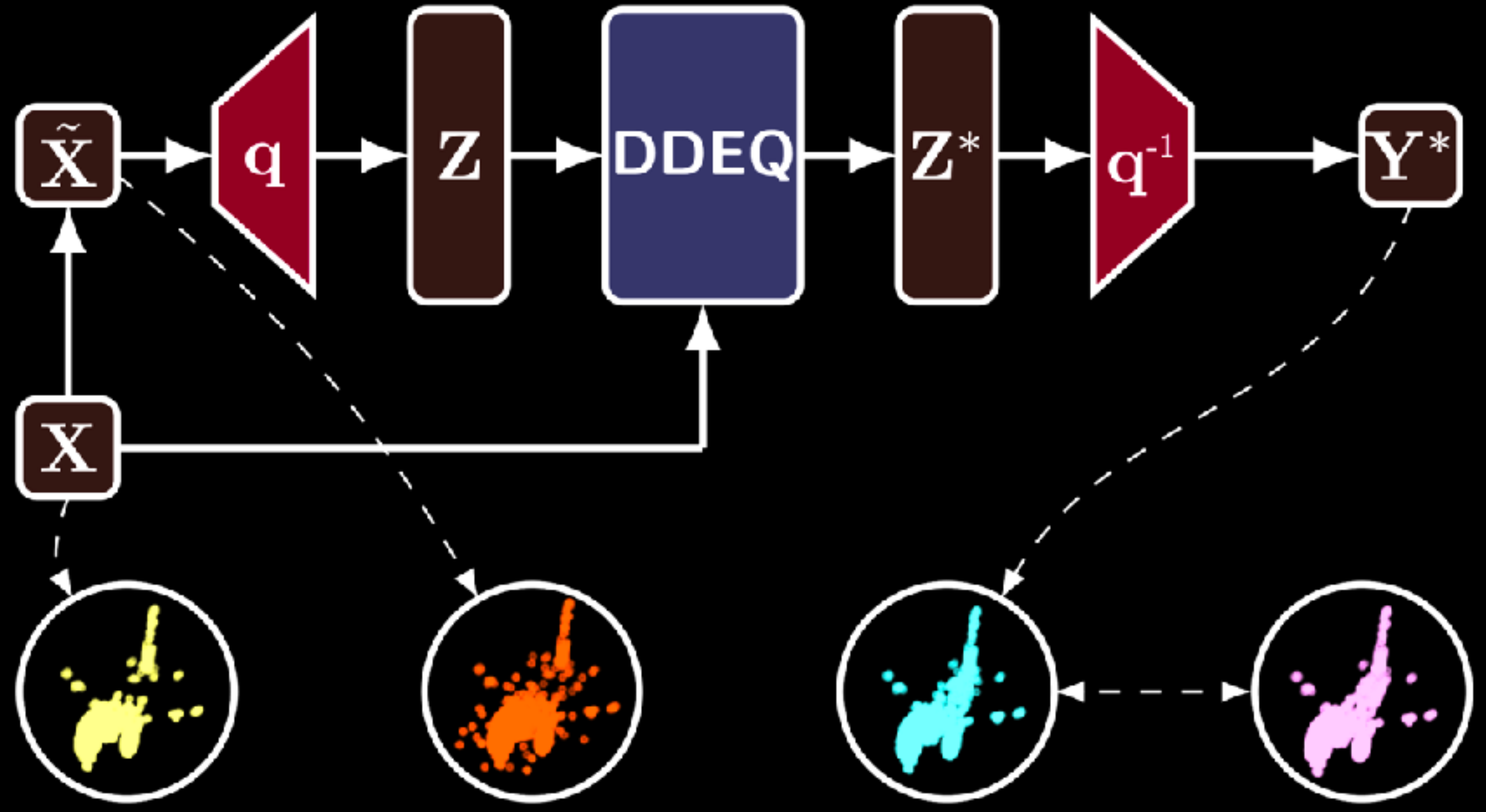


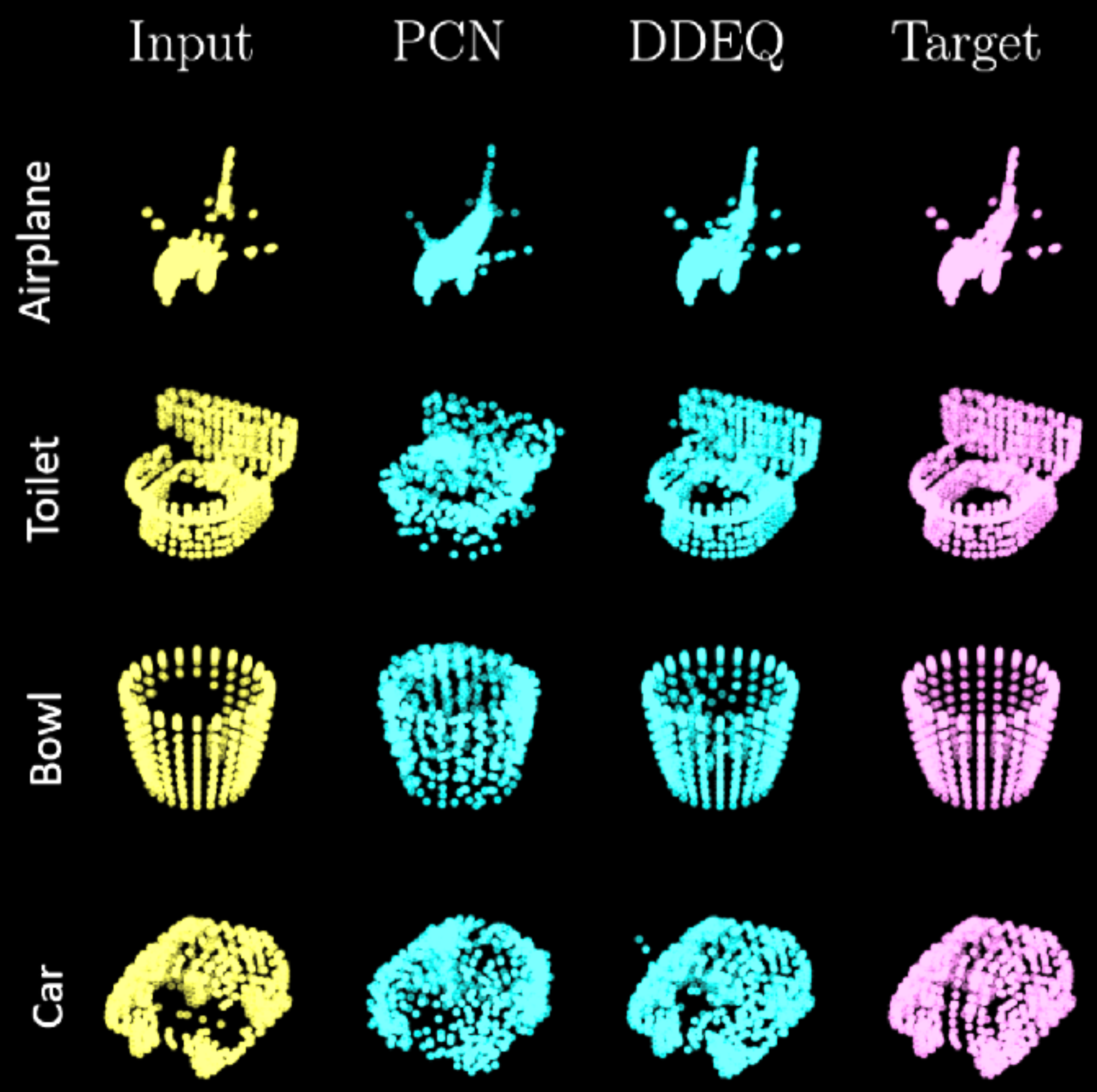
Distributional DEQ Architecture



We prove the architecture we use (transformer with self- and cross-attention) fulfills desirable properties (equivariance in μ, invariance in ρ)

DDEQs for Point-Cloud Completion





DDEQs for Point-Cloud Classification

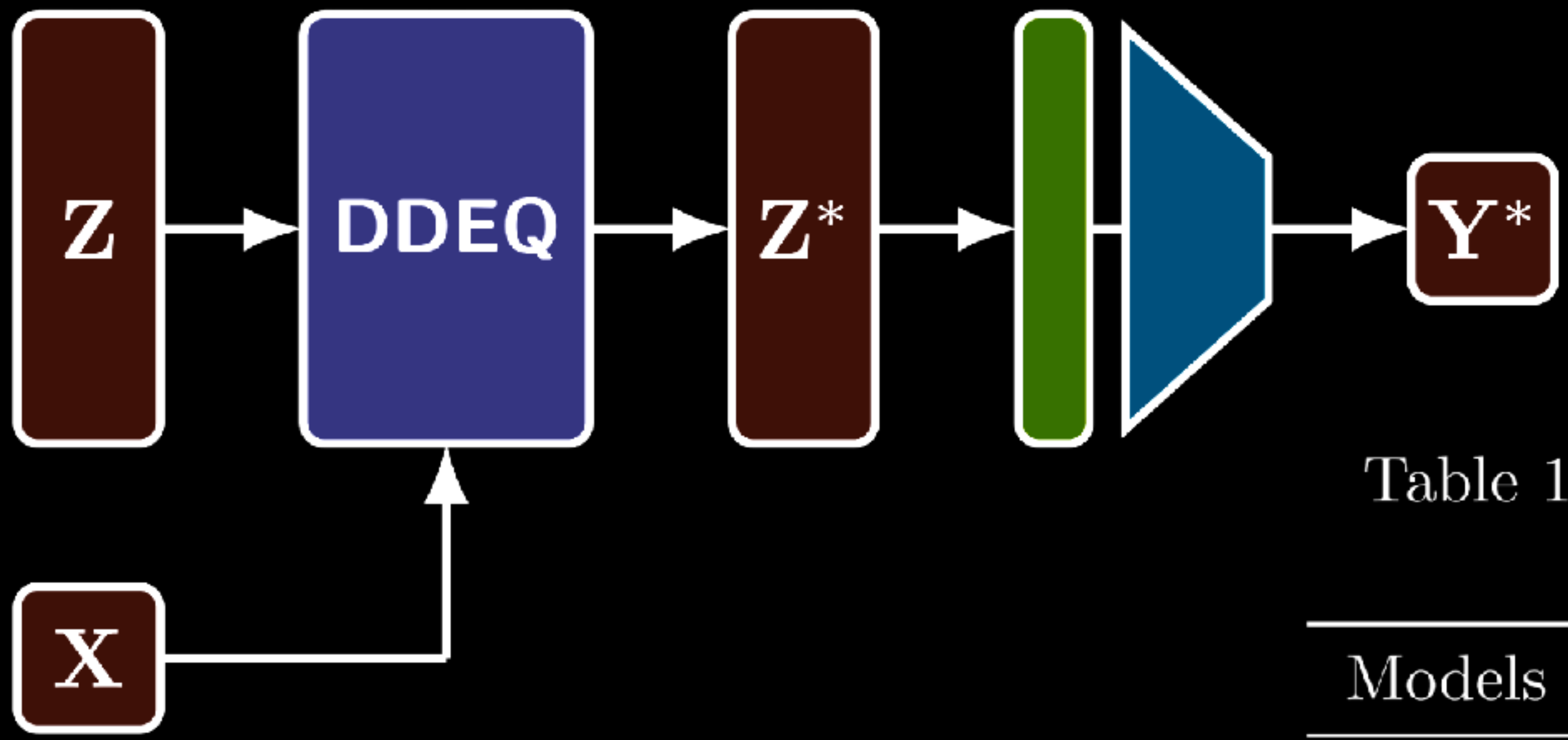


Table 1: Accuracies on Point Cloud Classification

Models (size)	**MNIST-pc**	**ModelNet40-s**
PointNet (1.6M)	97.5	77.3
PT (3.5M)	98.6	79.2
DDEQ (776k/1.2M)	98.1	78.2

DDEQs: Takeaways

- Fixed points in distribution space
  - Extends DEQs from fixed-size vectors to probability measures
  - Equilibrium is a distribution refined until self-consistency.
- Handles variable-Size, permutation-Invariant Inputs
  - Point clouds, sets, samples, and multimodal collections fit naturally.
  - No padding, ordering, or architectural hacks needed.
- Connections to **dynamic optimal transport** and PDEs (via Wasserstein gradient flows)
- A **unified continuum view** of depth (via DEQ) and input data (via distributions)

Takeaways

1. Continuous interpolation addresses key challenges of discrete choice
2. Interpolation and alignment are two sides of the same coin (eg OT!)
3. Why pick one model/dataset when you could "*have them all*"?
4. Stop Choosing. Start Interpolating.