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

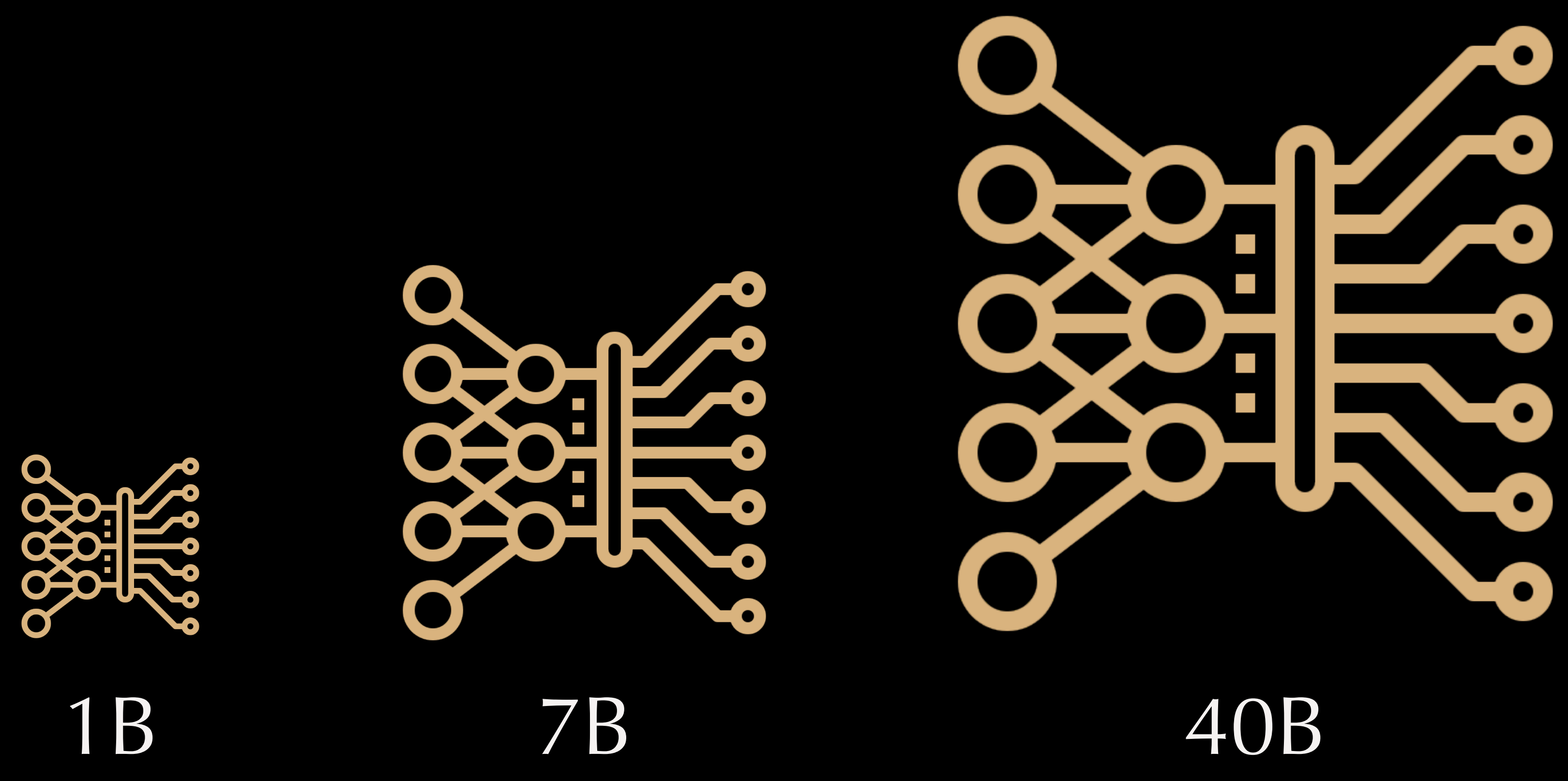
David Alvarez-Melis

Harvard SEAS | Kempner Institute | MSR

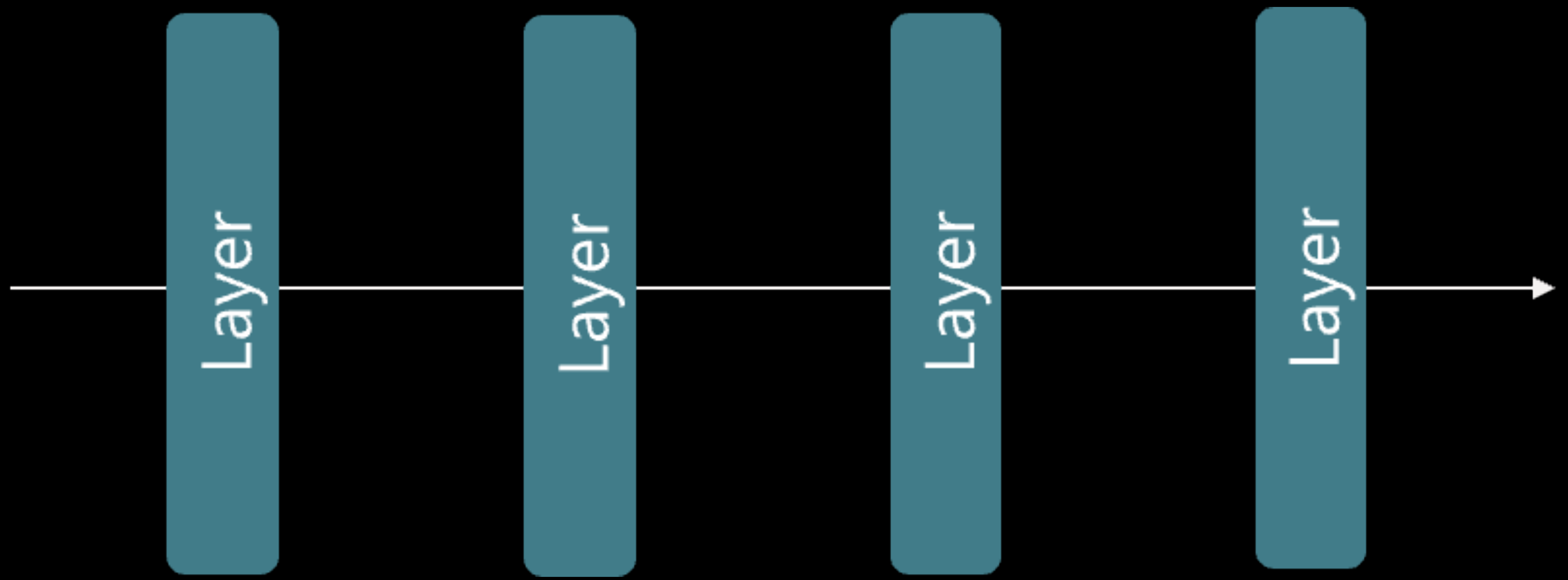
AI-SDM Seminar, CMU

Feb 20th 2026

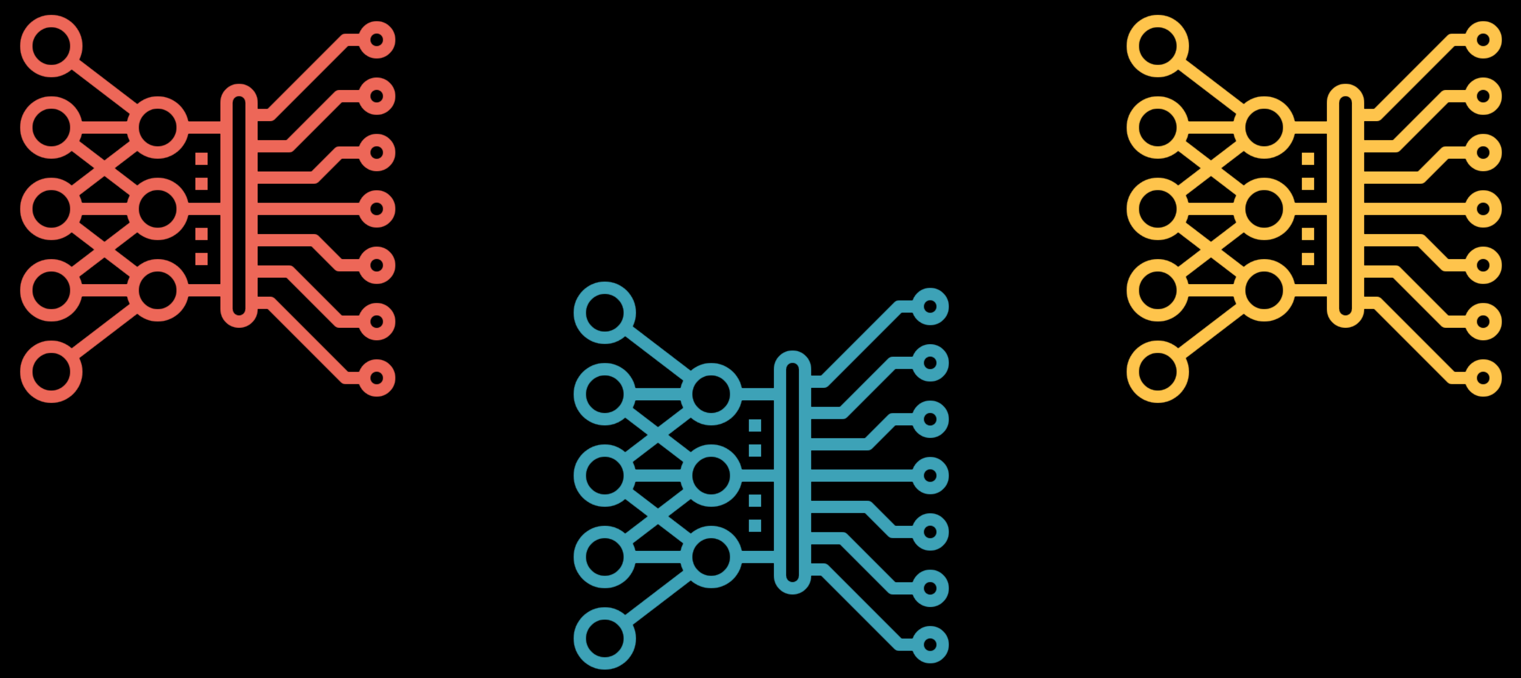
ML is full of discrete choices…



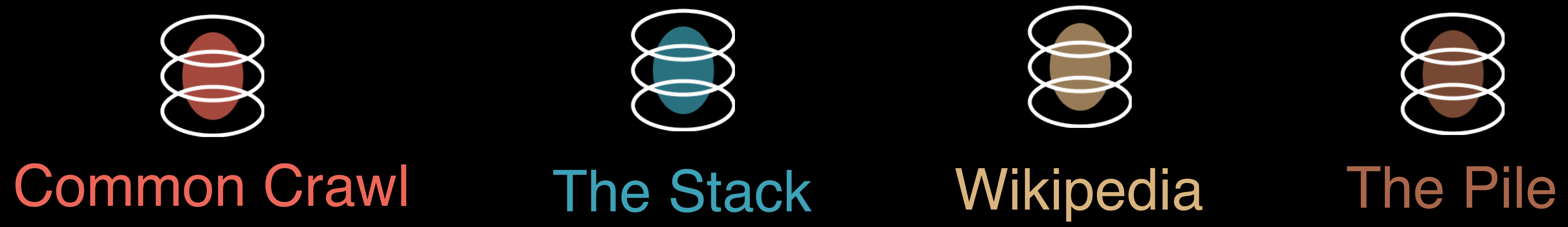
Model Sizes



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

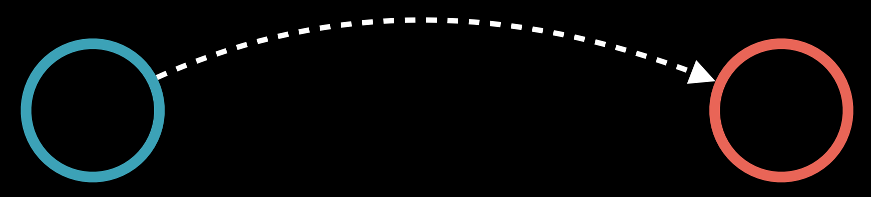
- Coarse-grained choices prevent smooth control over e.g compute.
- Discontinuous jumps in performance and compute cost.
- Hard to generalize or adjust capacity with locked-in configs.

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?

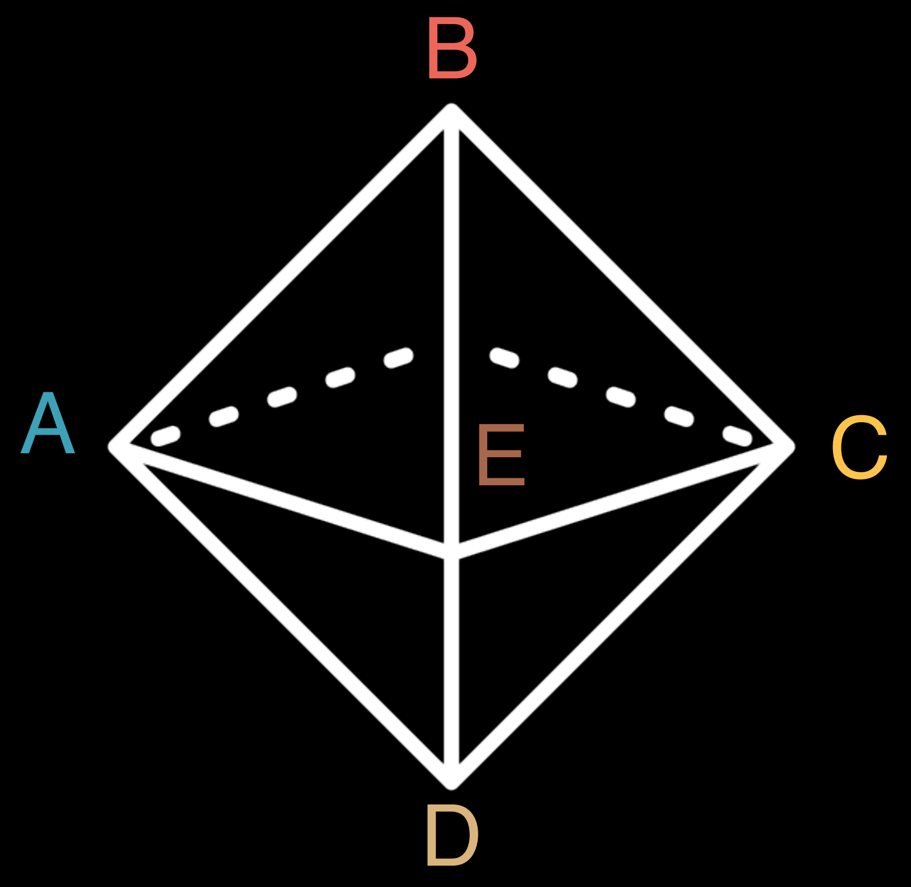


Disconnected points → Continuous paths

(configurations become connected rather than isolated)

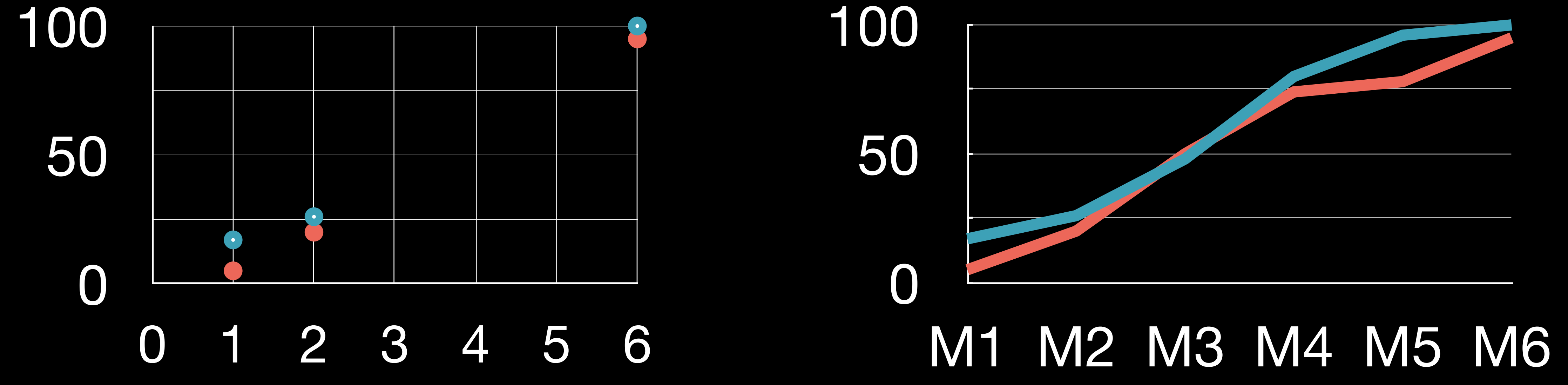
One-off Solutions → Families of Related Solutions

(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.

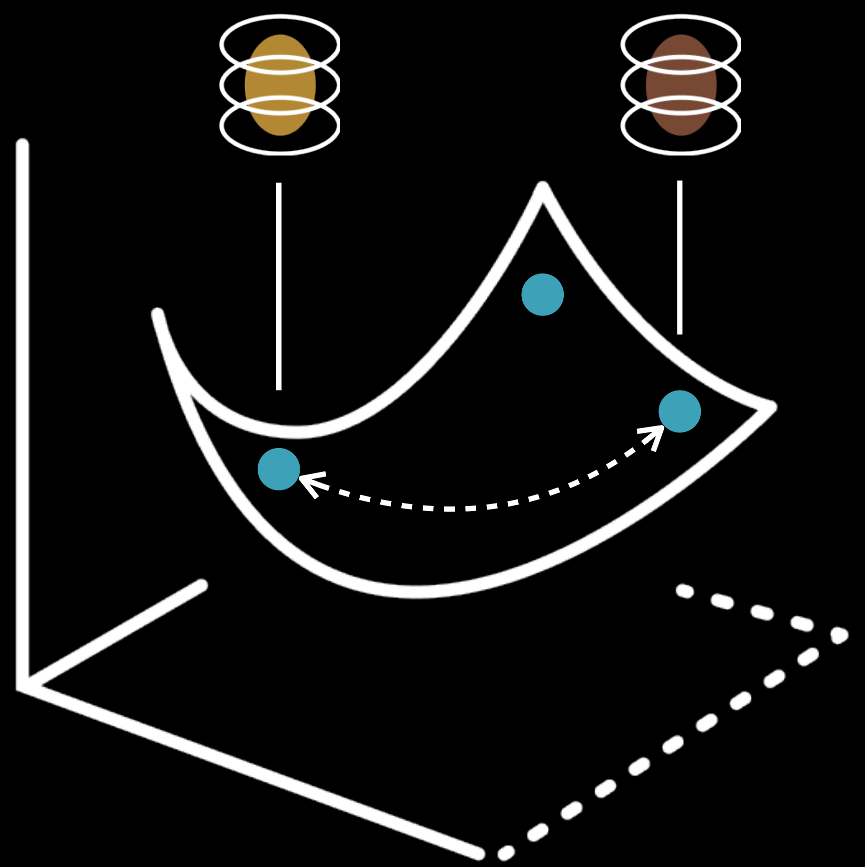
Efficiency by Sharing

- 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

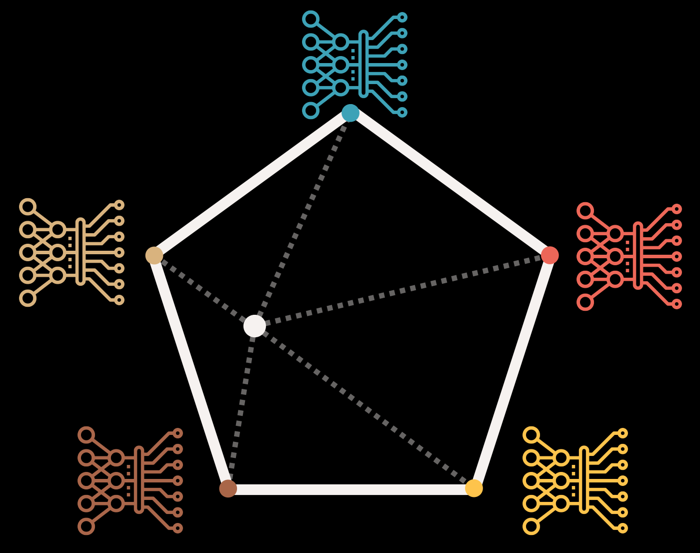
Datasets:



Interpolation along Generalized Geodesics

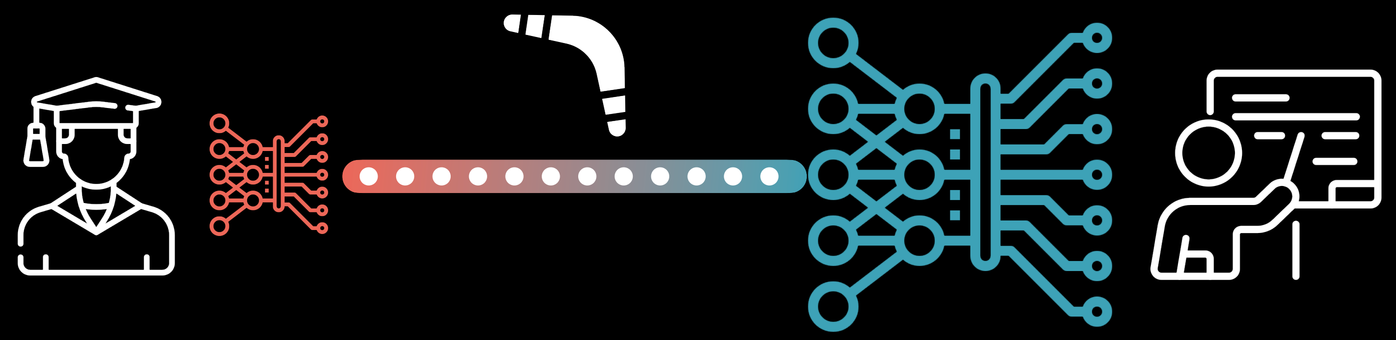
[Fan & AM, UAI 2023]

Models:



Weight-space interpolation

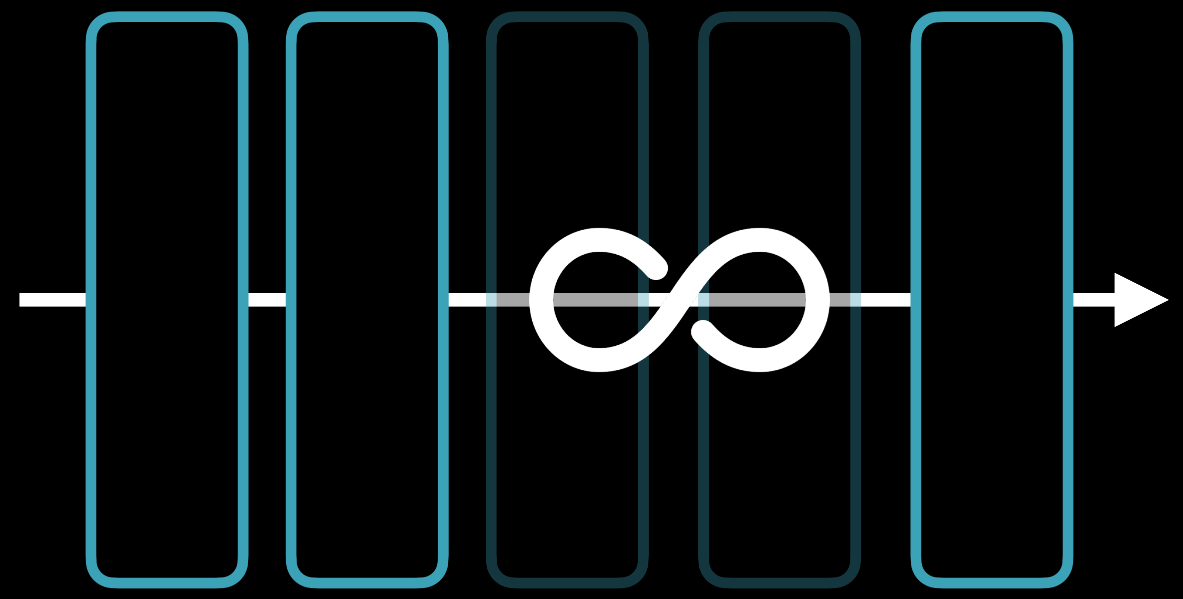
[Kangaslahti & AM, TMLR 2025]



Boomerang Distillation

[Kangaslahti et al., ICLR 2026]

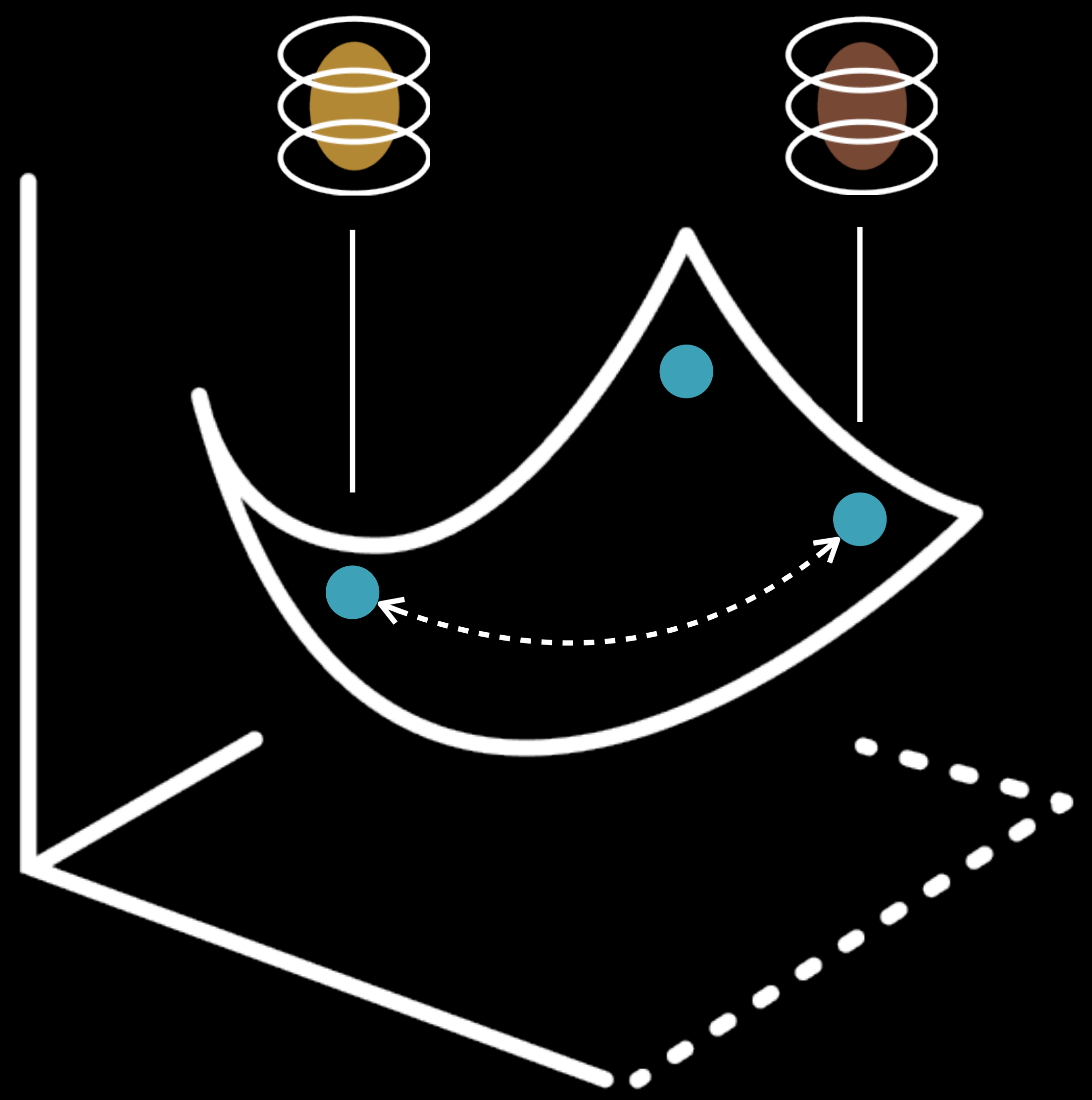
Compute:



Distributional Deep Equilibrium Models

[Geuter et al., AISTATS 2025]

Interpolating Datasets

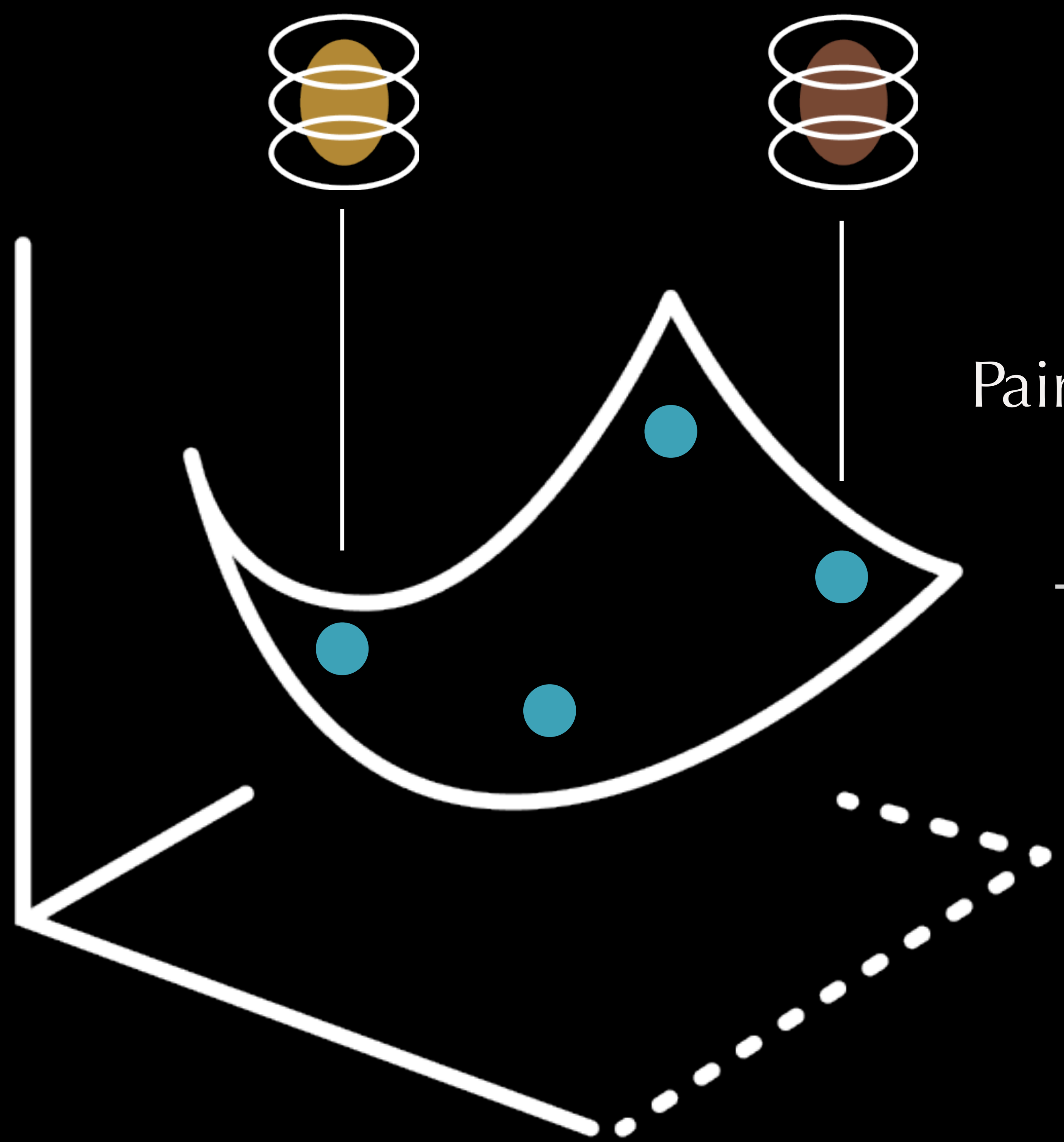


Generating Synthetic Datasets by Interpolating along Generalized Geodesics

Fan & AM, UAI 2023

Jiaojiao Fan

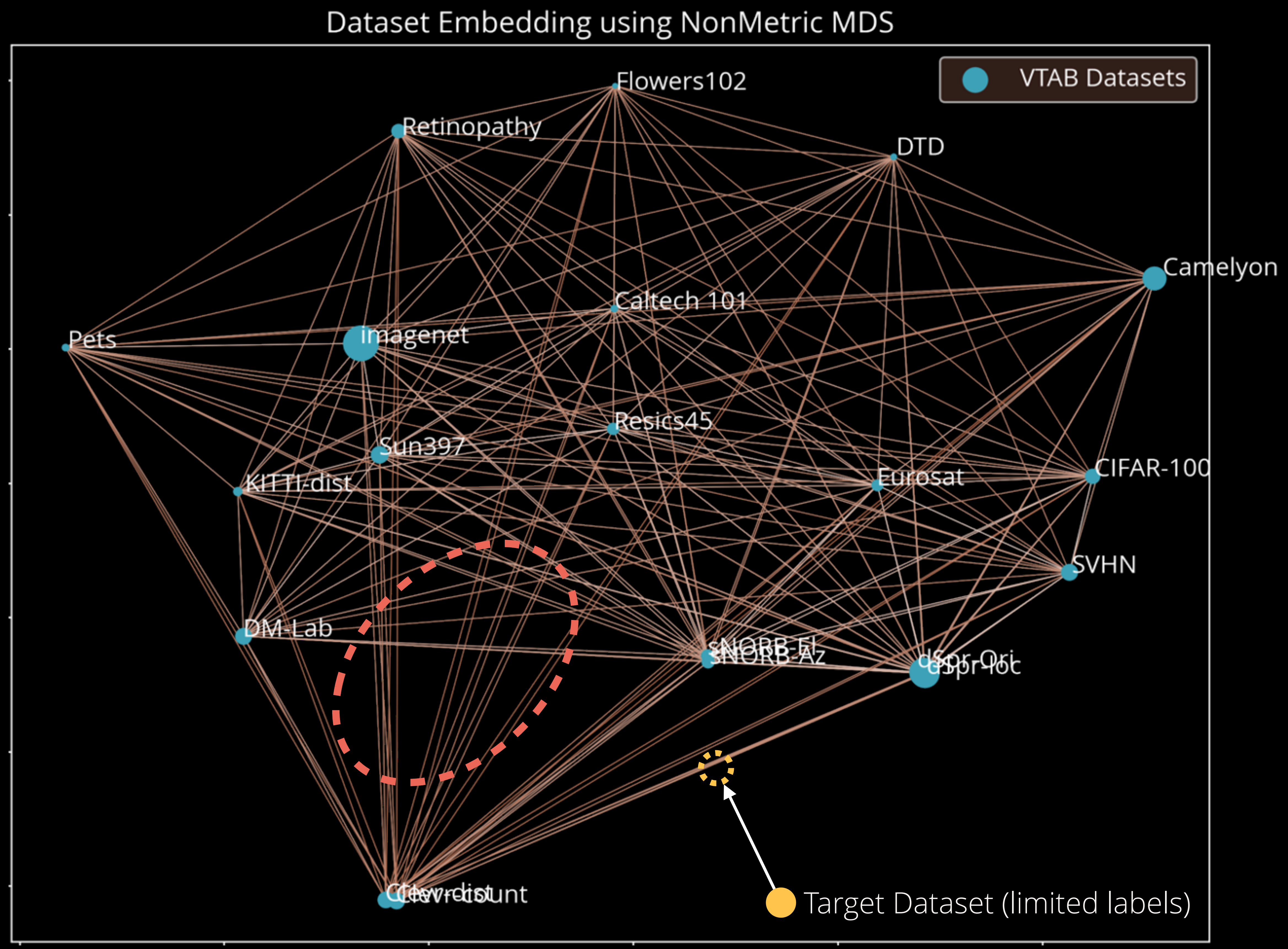
Dataset Space: A Continuous View



Training Datasets

Pairwise Optimal Transport dataset distances

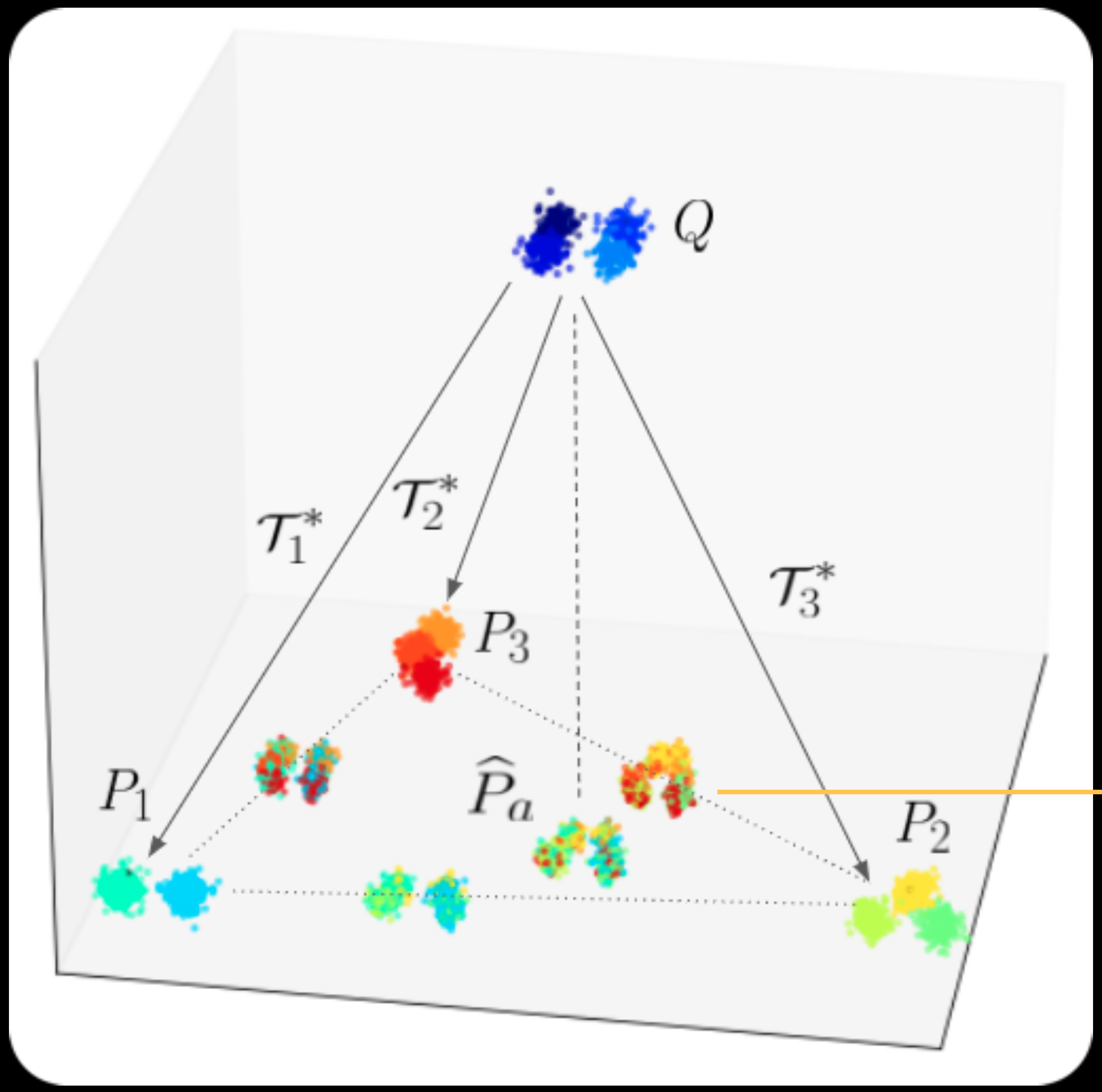
(AM & Fusi, 2020)



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

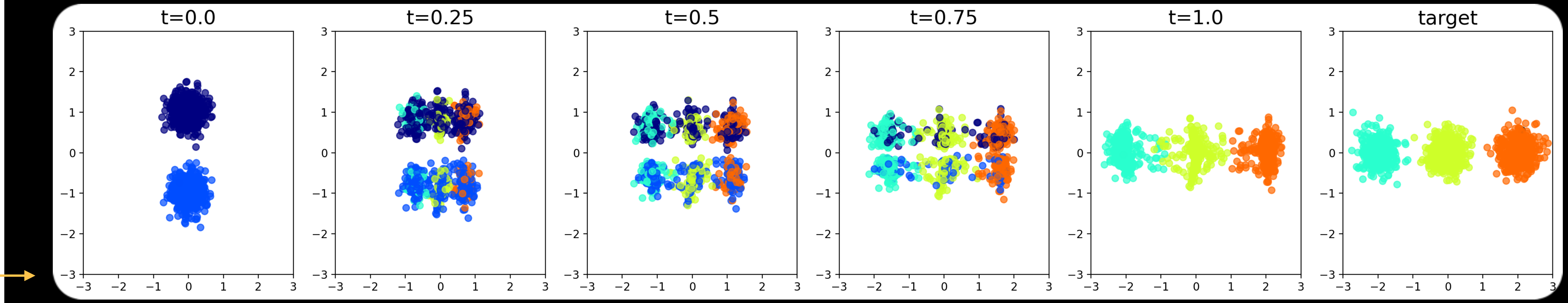
Synthesizing Datasets for Generalization

- 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 $P_a = \left(\sum_{i=1}^{m} a_i \mathcal{T}_i^*\right)_\sharp Q$

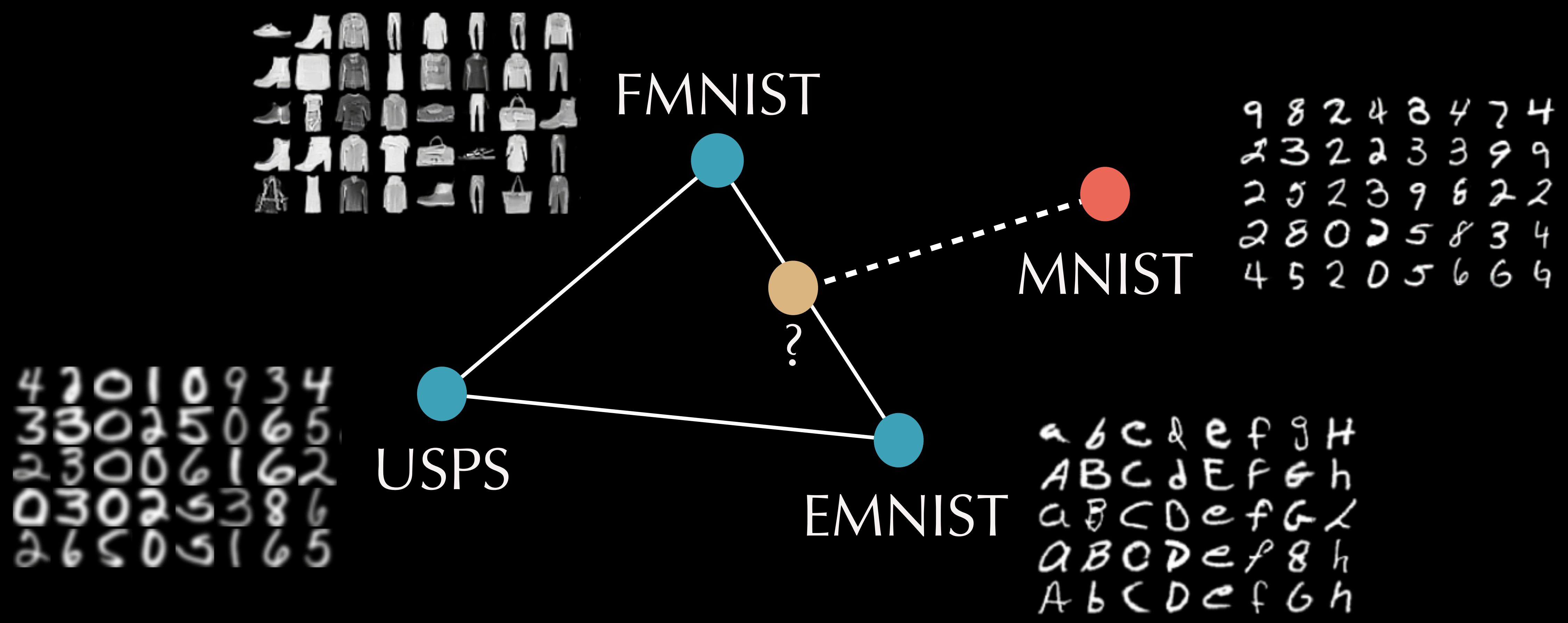
Example geodesic between two datasets



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

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

Synthesizing Datasets for Generalization



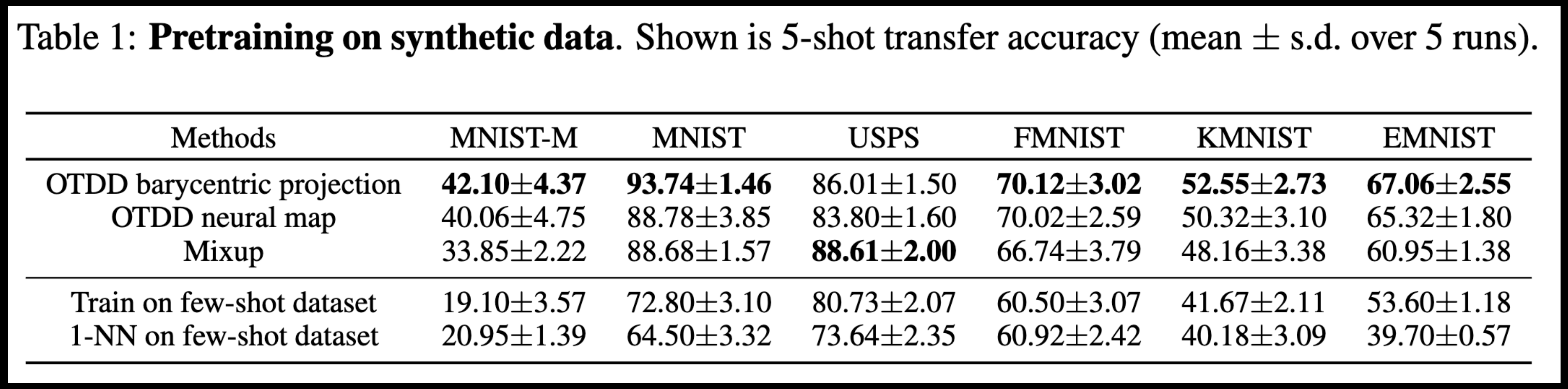
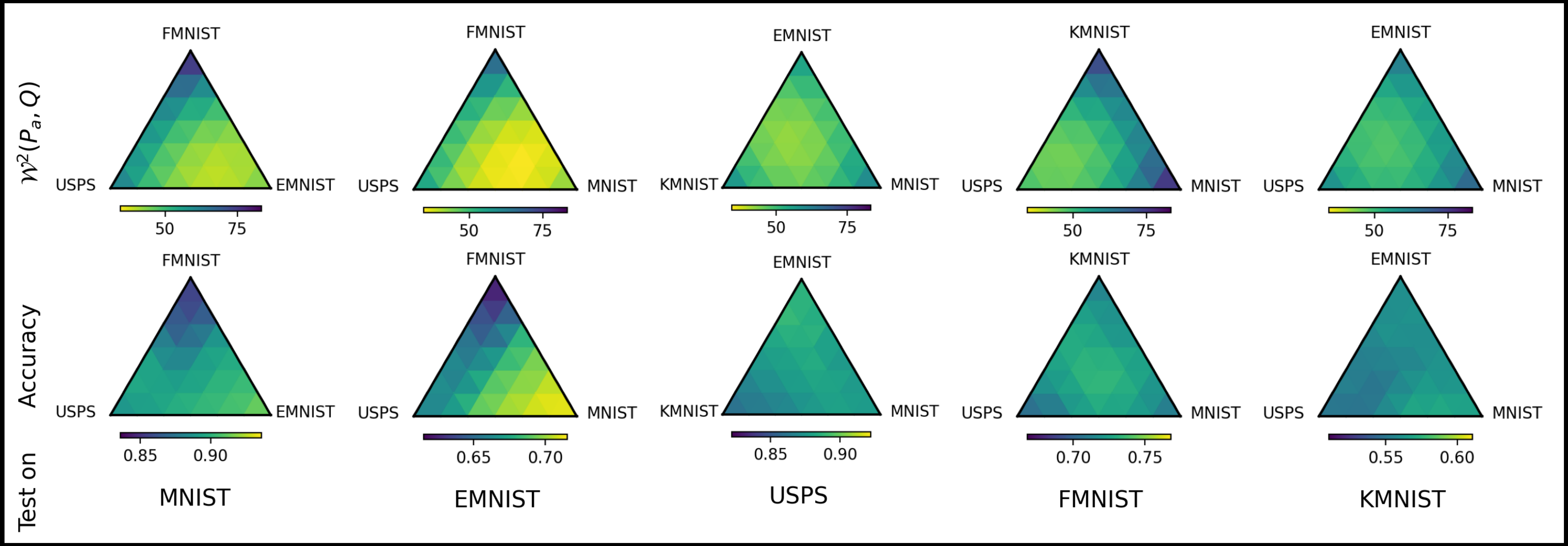


Table 1: **Pretraining on synthetic data**. Shown is 5-shot transfer accuracy (mean ± s.d. over 5 runs).

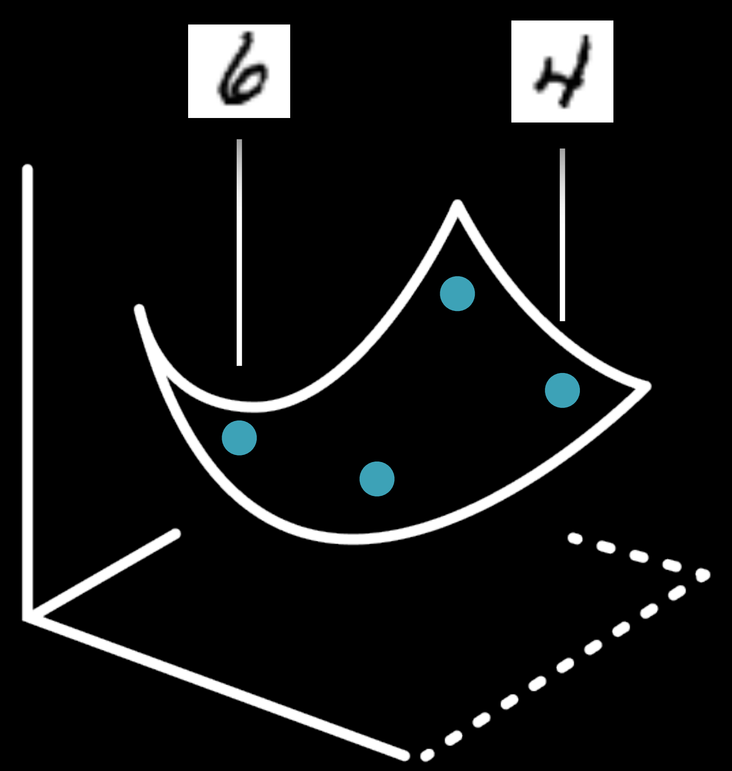
Methods	MNIST-M	MNIST	USPS	FMNIST	KMNIST	EMNIST
OTDD barycentric projection	**42.10±4.37**	**93.74±1.46**	86.01±1.50	**70.12±3.02**	**52.55±2.73**	**67.06±2.55**
OTDD neural map	40.06±4.75	88.78±3.85	83.80±1.60	70.02±2.59	50.32±3.10	65.32±1.80
Mixup	33.85±2.22	88.68±1.57	**88.61±2.00**	66.74±3.79	48.16±3.38	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

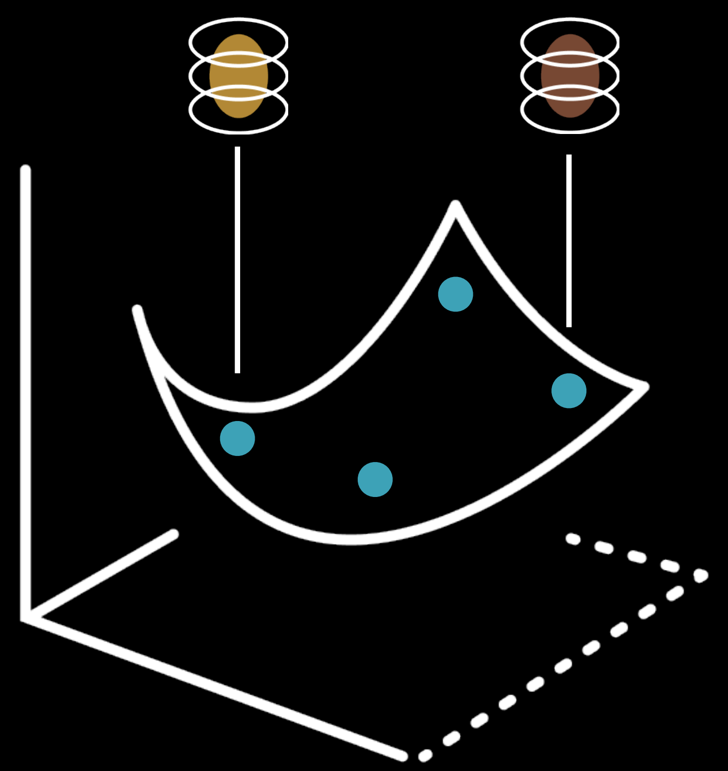
OT dist between synthetic dataset and target dataset

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



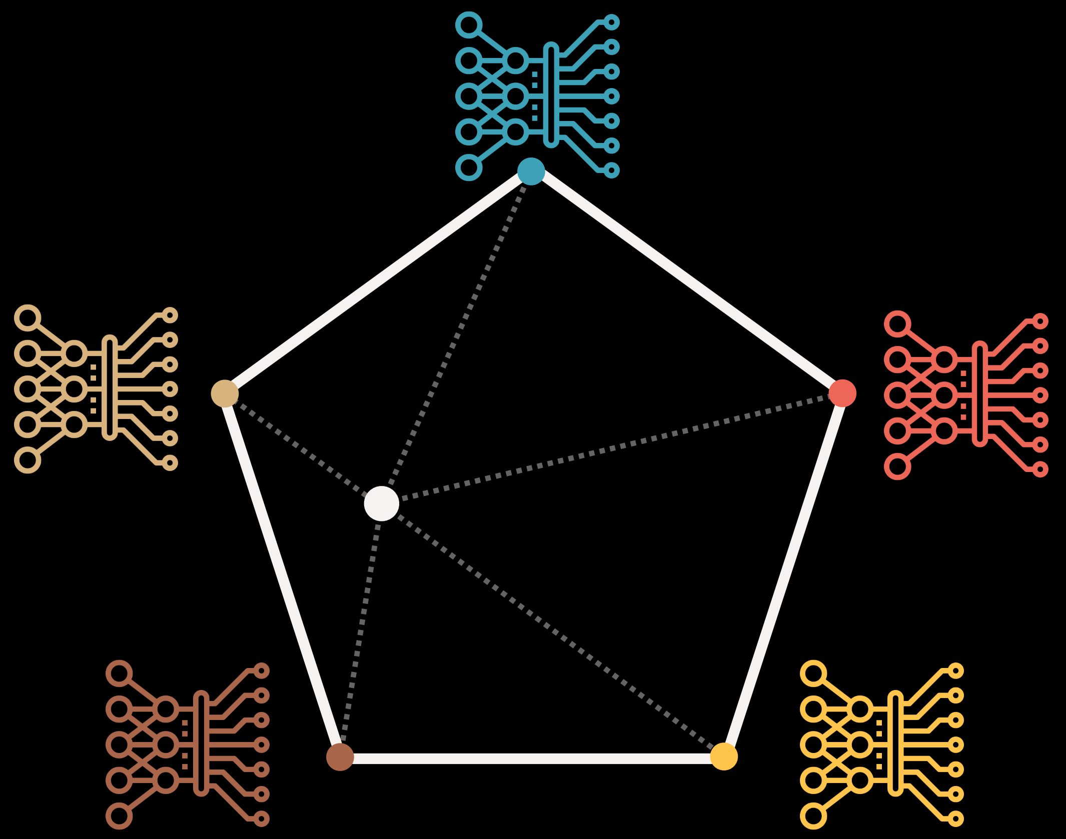
Takeaways


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

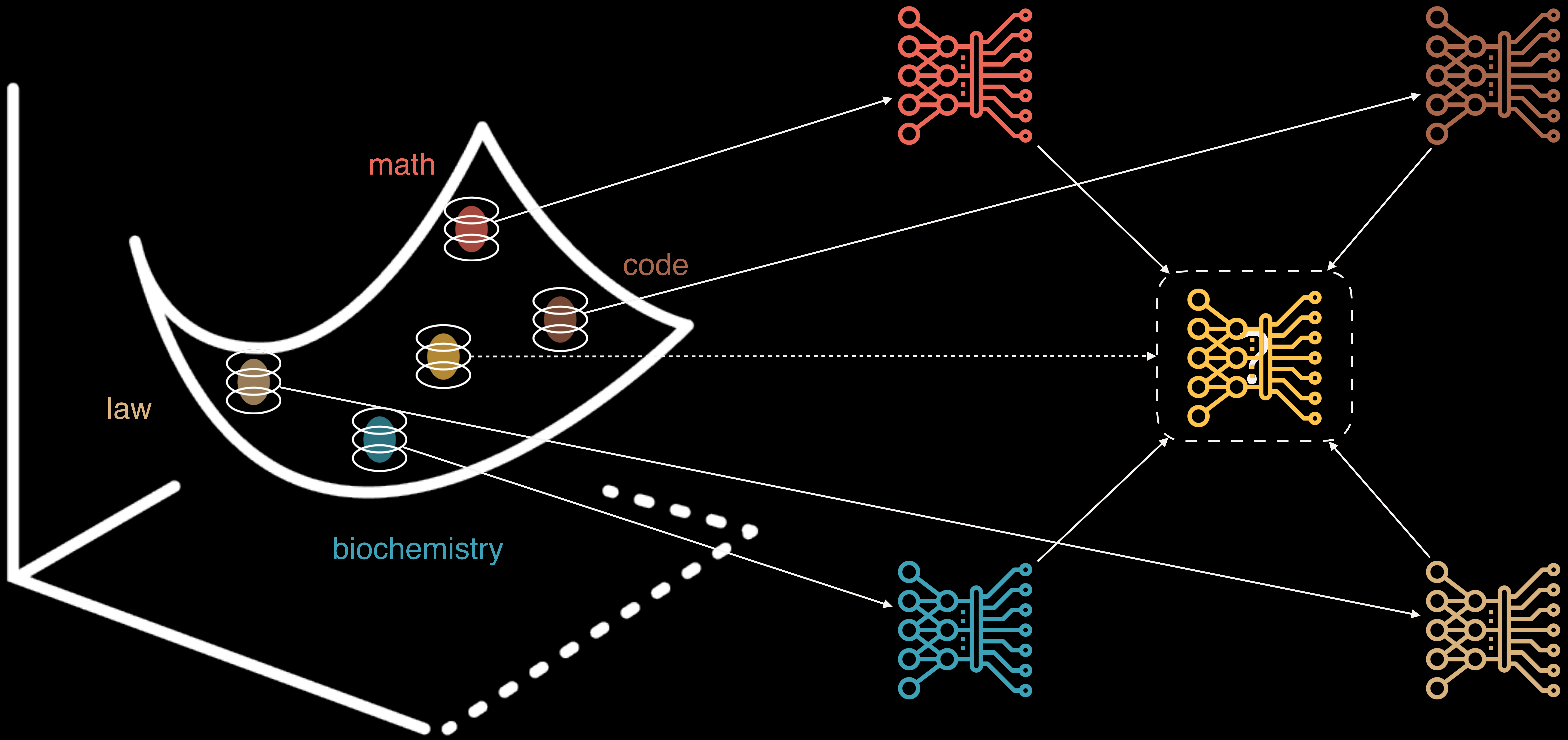


Continuous Language Model Interpolation for Dynamic an Controllable Text Generation

Kangaslahti & AM, TMLR 2025

Sara Kangaslahti

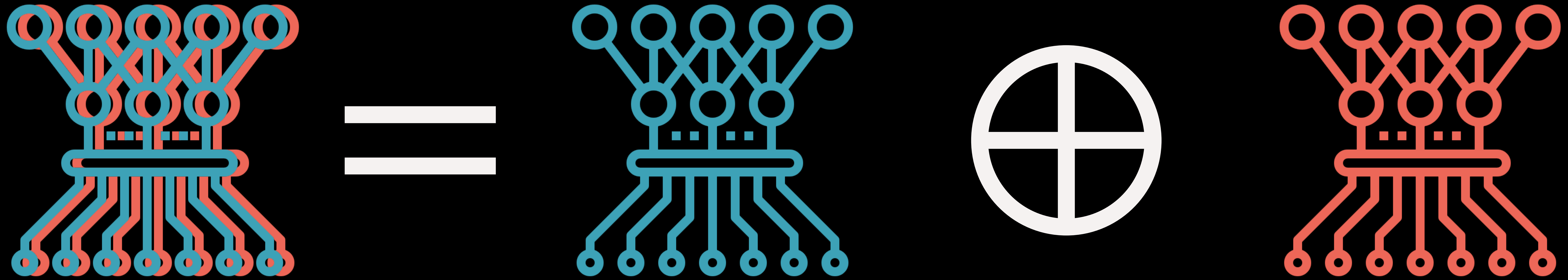
Cross-domain Interpolation



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:



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

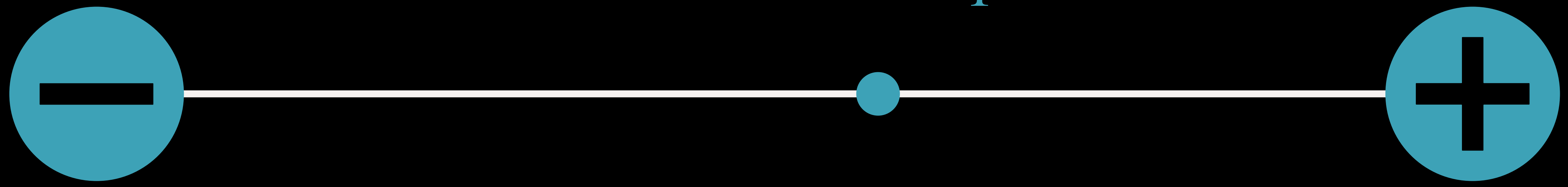
Continuous Model Interpolation

θ_0 : weights of base pretrained model

A, B : LoRA weight updates

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

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



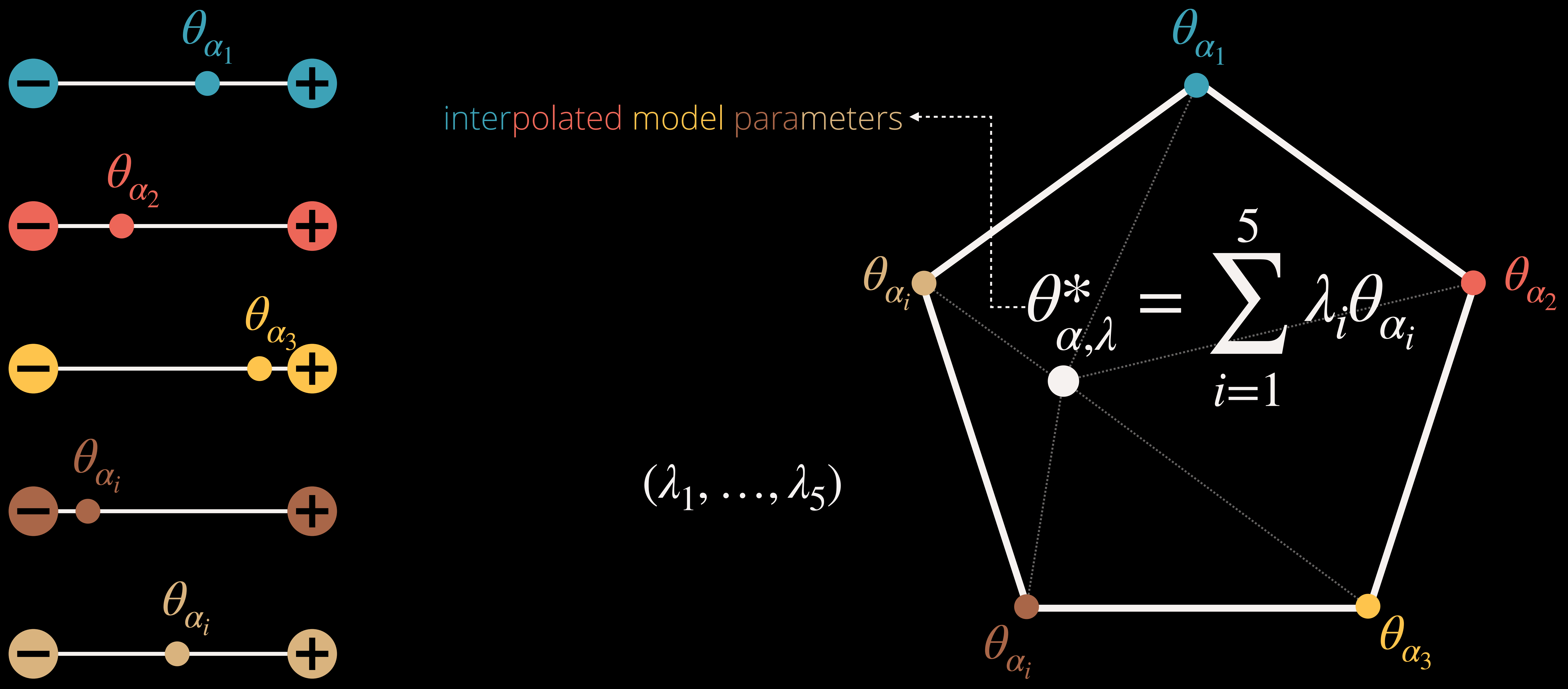
$$\theta_{-i} = \theta_0 + A_{-i}B^{\top}_{-i}$$

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

e.g. simple language

e.g. complex language

Continuous Model Interpolation



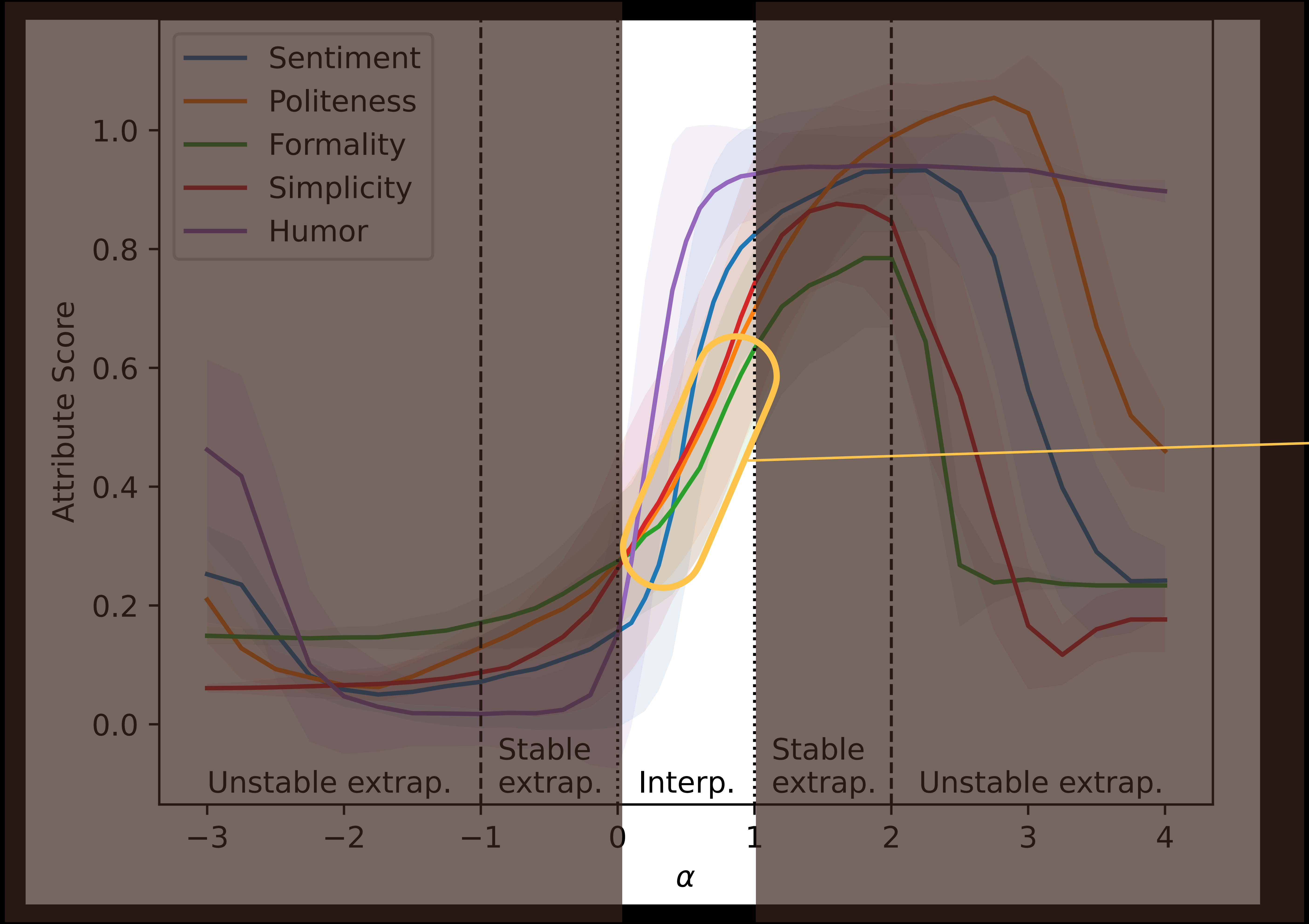
interpolated model parameters

$$\theta^*_{\alpha,\lambda} = \sum_{i=1}^{5} \lambda_i \theta_{\alpha_i}$$

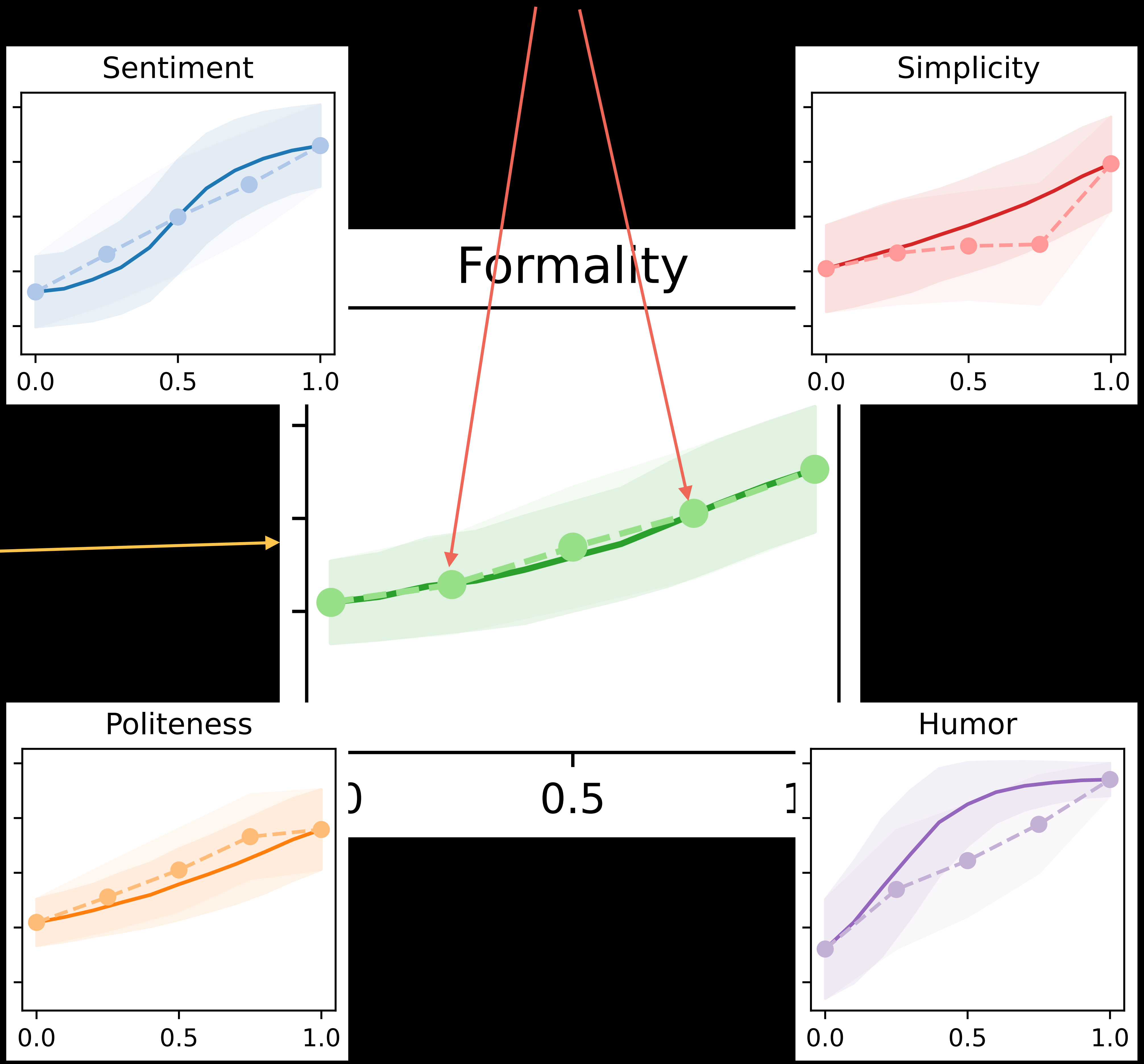
$(\lambda_1, \ldots, \lambda_5)$

How faithful is the interpolation?

Single-Attribute Inter/Extra-polation



models **fine-tuned** on α-data mix

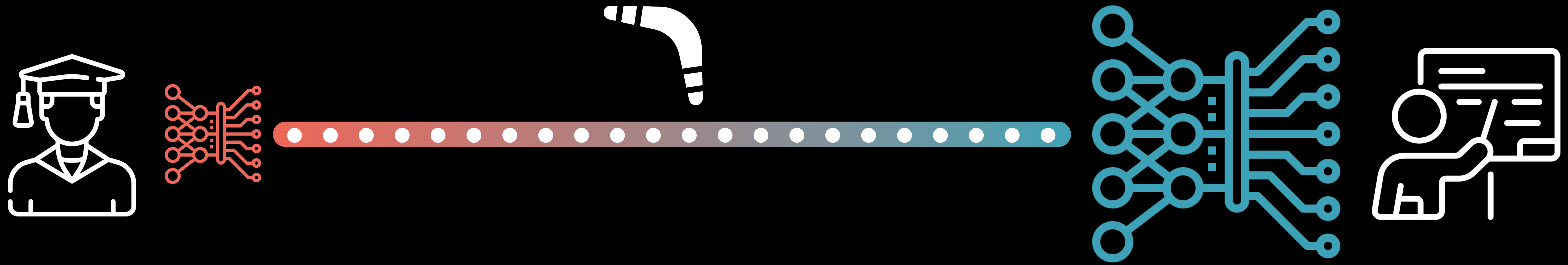


synthetic models **closely track** fine-tuned ones!

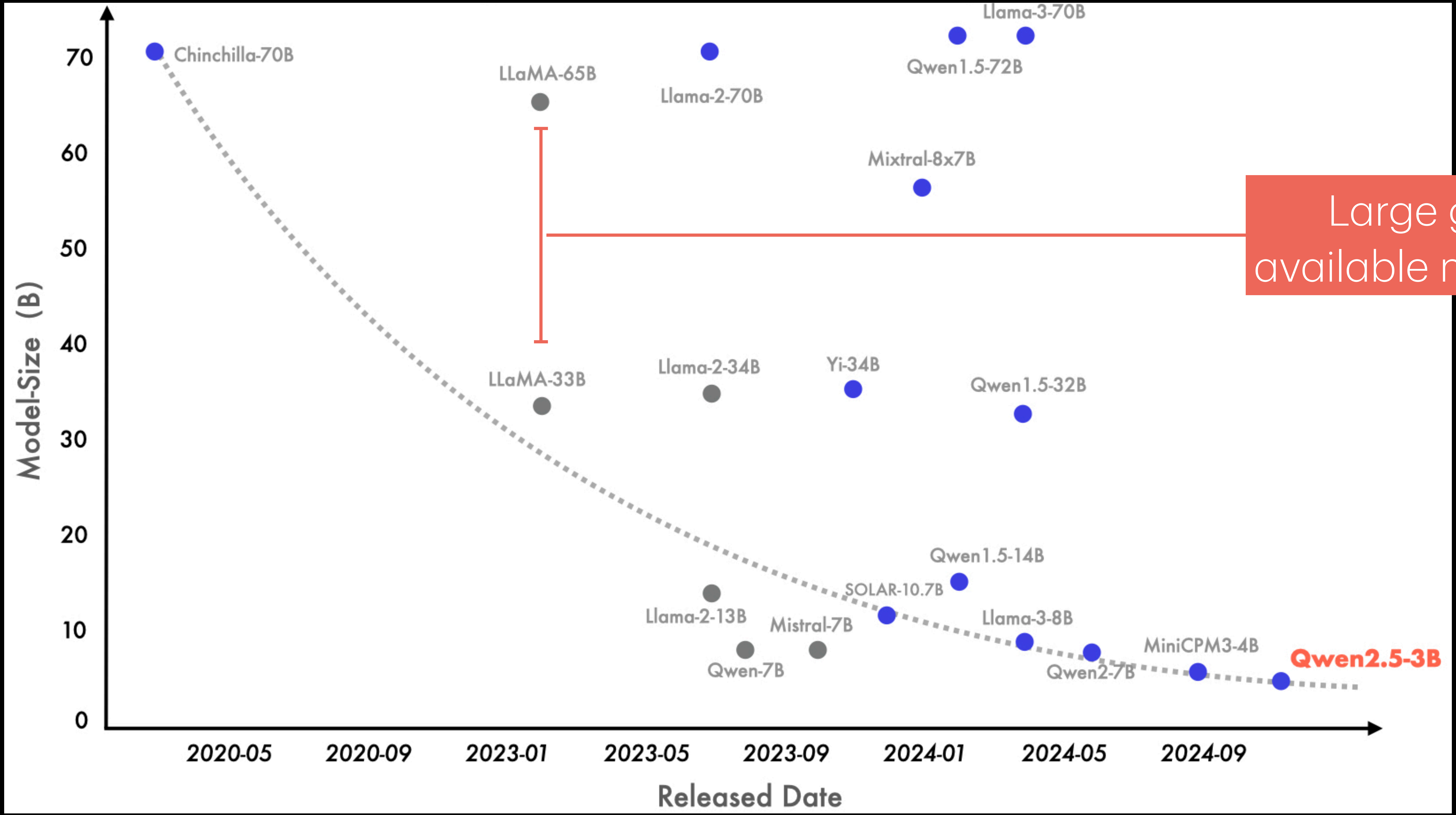
Sara Kangaslahti

Boomerang Distillation Enables Zero-Shot Model Size Interpolation

Kangaslahti, Geuter*, Nayak*, Fumero, Locatello, AM; ICLR 2026



Current landscape: pretrained model families



Large gaps in available model sizes

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!

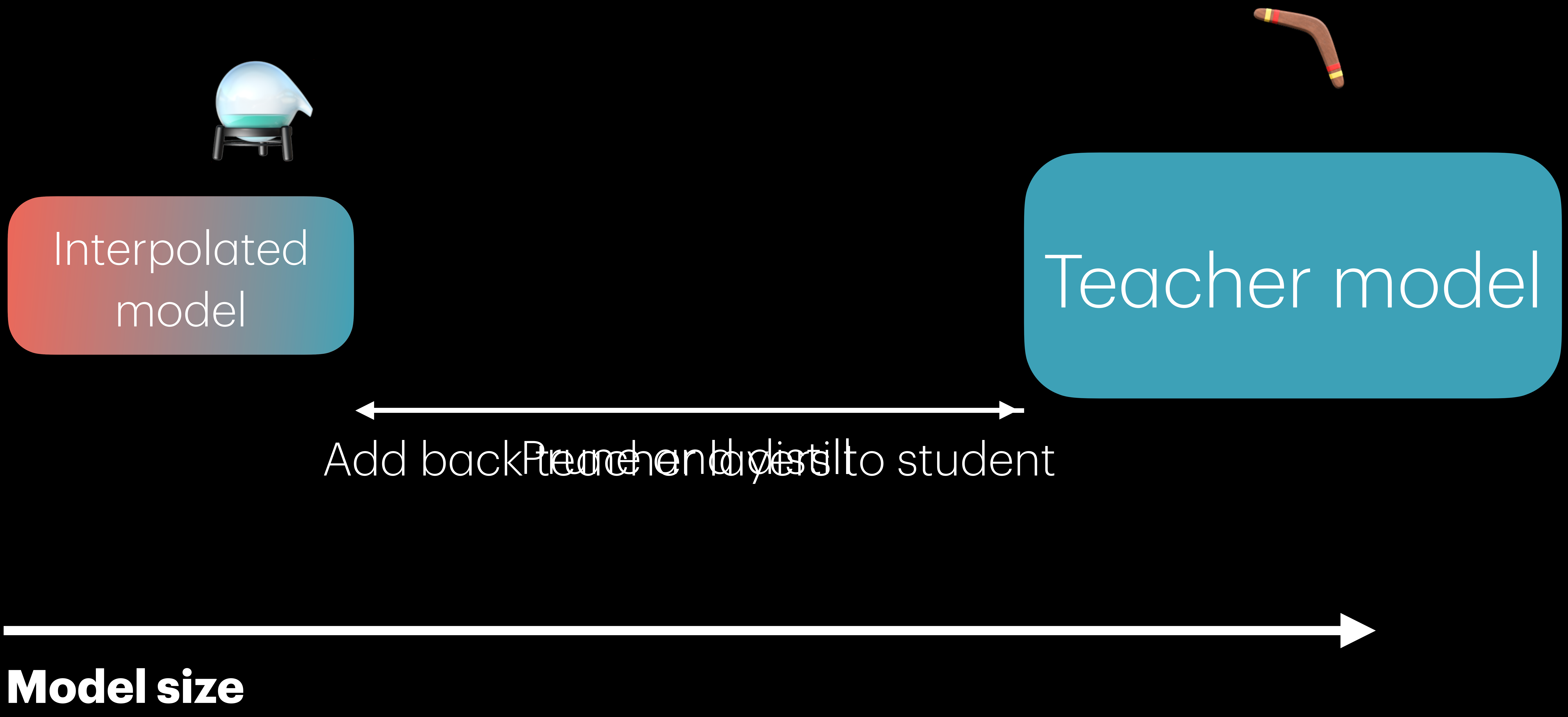
- What about distillation?

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

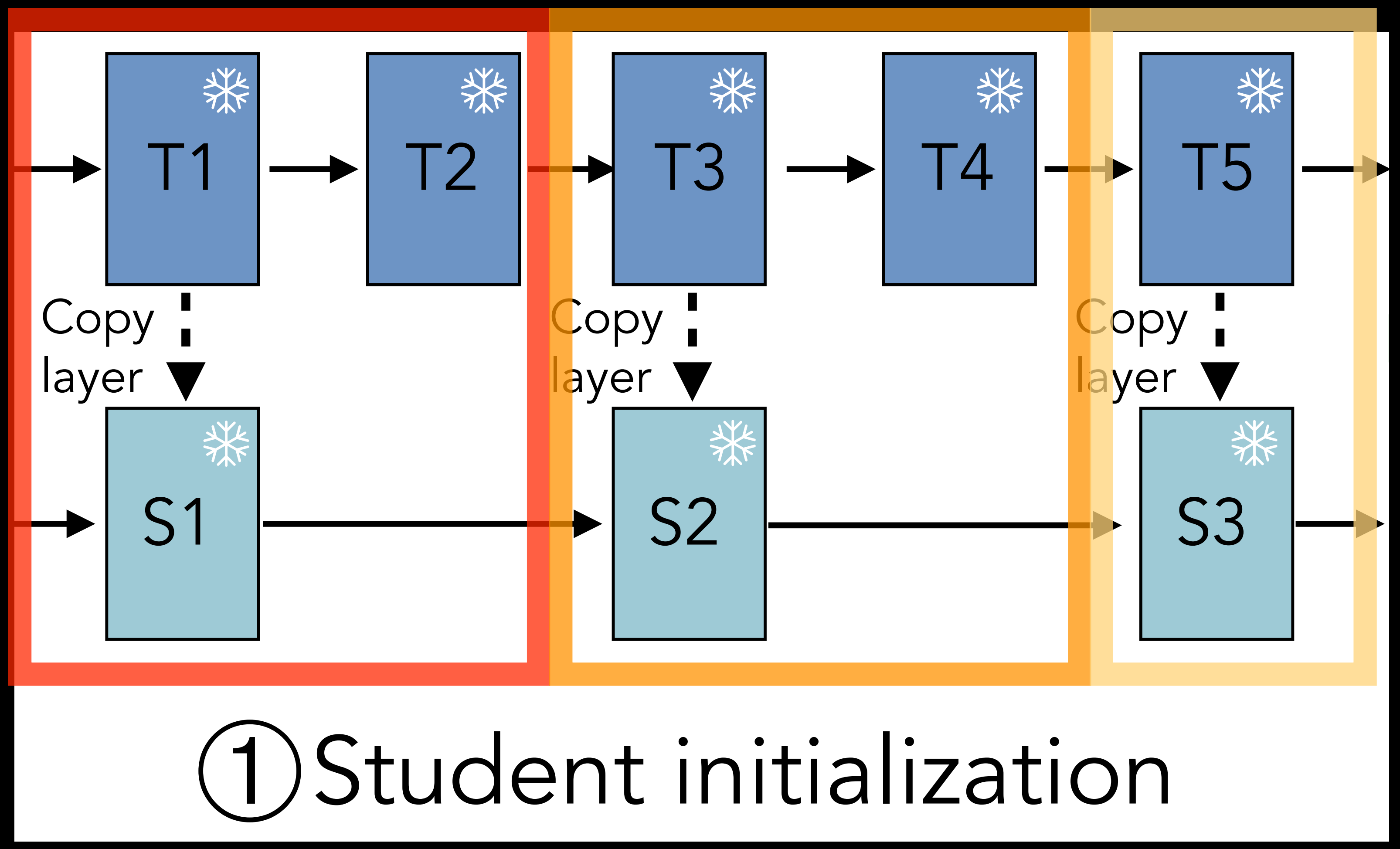
Teacher model

Add back teacher layers to student

Prune and distill

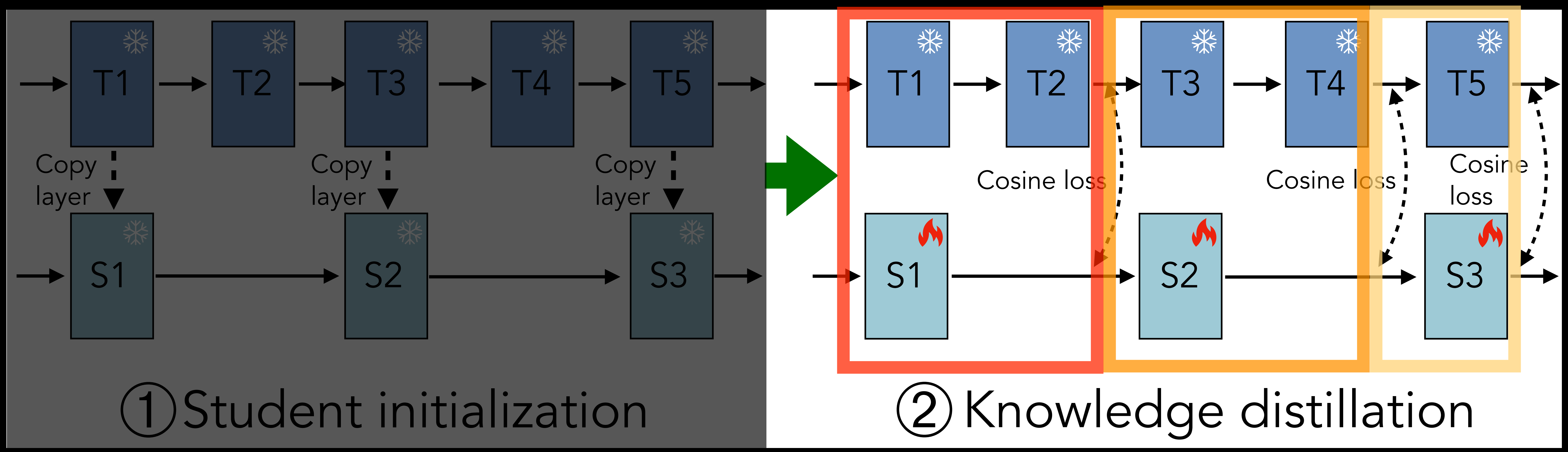
Model size

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 ⚗️

T1
T2
T3
T4
T5
Copy layer
Copy layer
Copy layer
S1
S2
S3
①Student initialization
T1
T2
T3
T4
T5
Cosine loss
Cosine loss
Cosine loss
S1
S2
S3
②Knowledge distillation
T1
T2
T3
T4
T5
Replace layer
Replace layer
S1
T3
T4
S3
③Student patching

Interpolate by patching any student layers with their corresponding teacher blocks

Boomerang 🪃 distillation ⚗️

T1
T2
T3
T4
T5
Copy layer
Copy layer
Copy layer
S1
S2
S3
T1
T2
T3
T4
T5
Cosine loss
Cosine loss
Cosine loss
S1
S2
S3
T1
T2
T3
T4
T5
Replace layer
Replace layer
S1
T3
T4
S3

①Student initialization ②Knowledge distillation ③Student patching

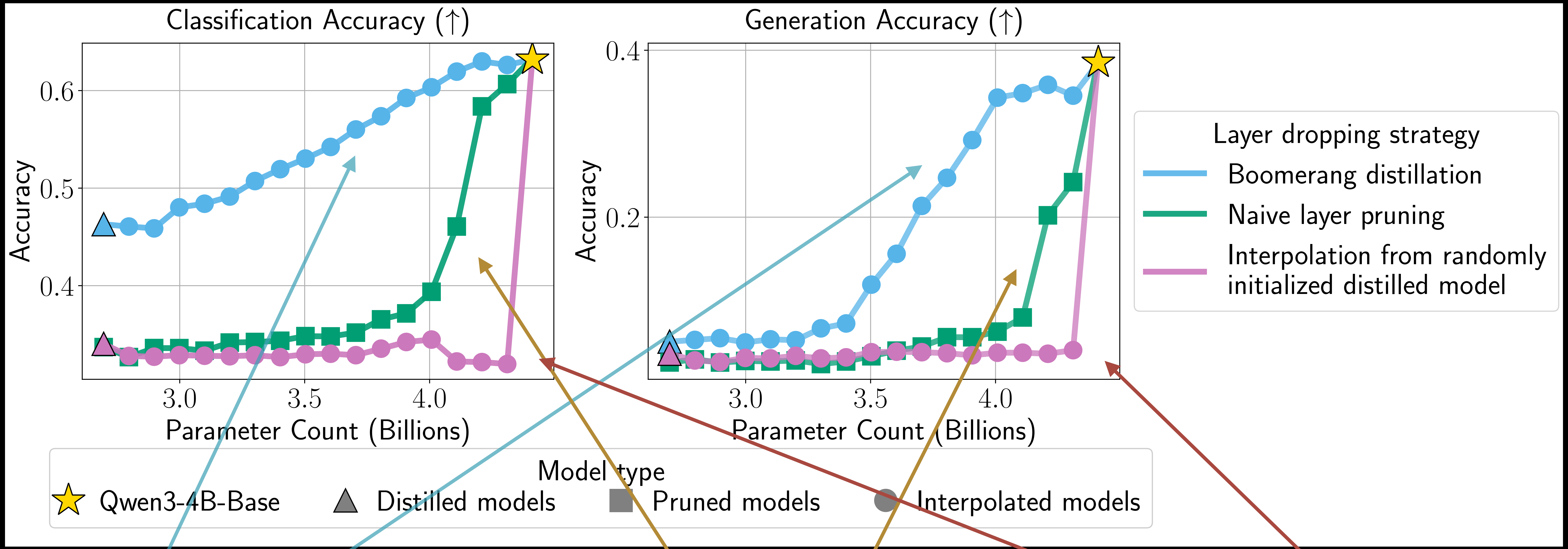
Note: Only first (smallest) model is trained with distillation objective, all other intermediate models are materialized **without additional training!**

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

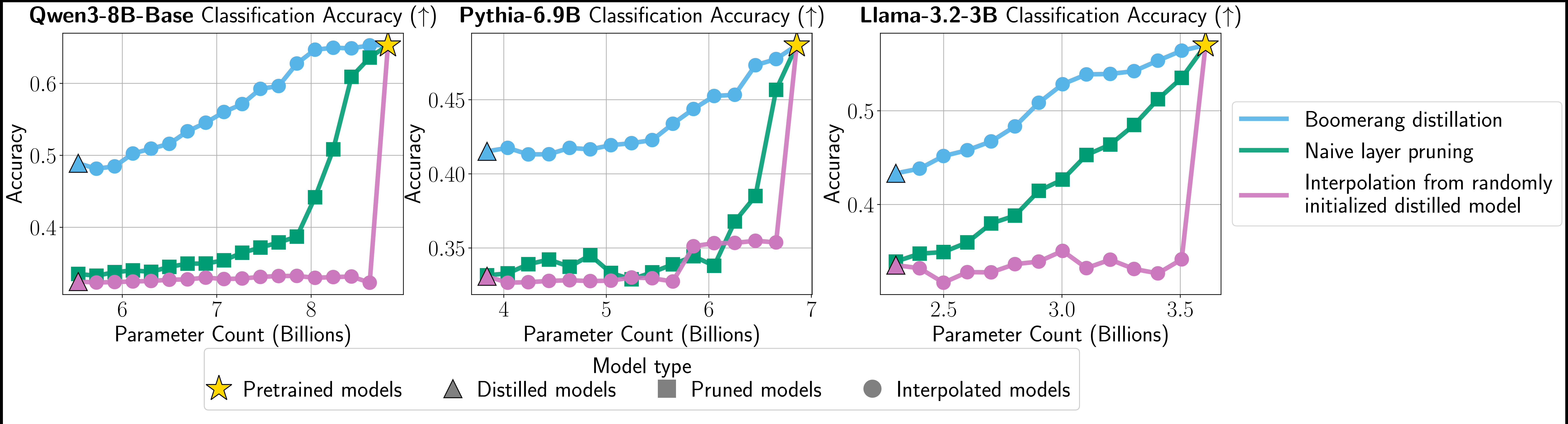


Boomerang distillation yields smooth performance interpolation behavior

Naive pruning (without distillation) fails at interpolation

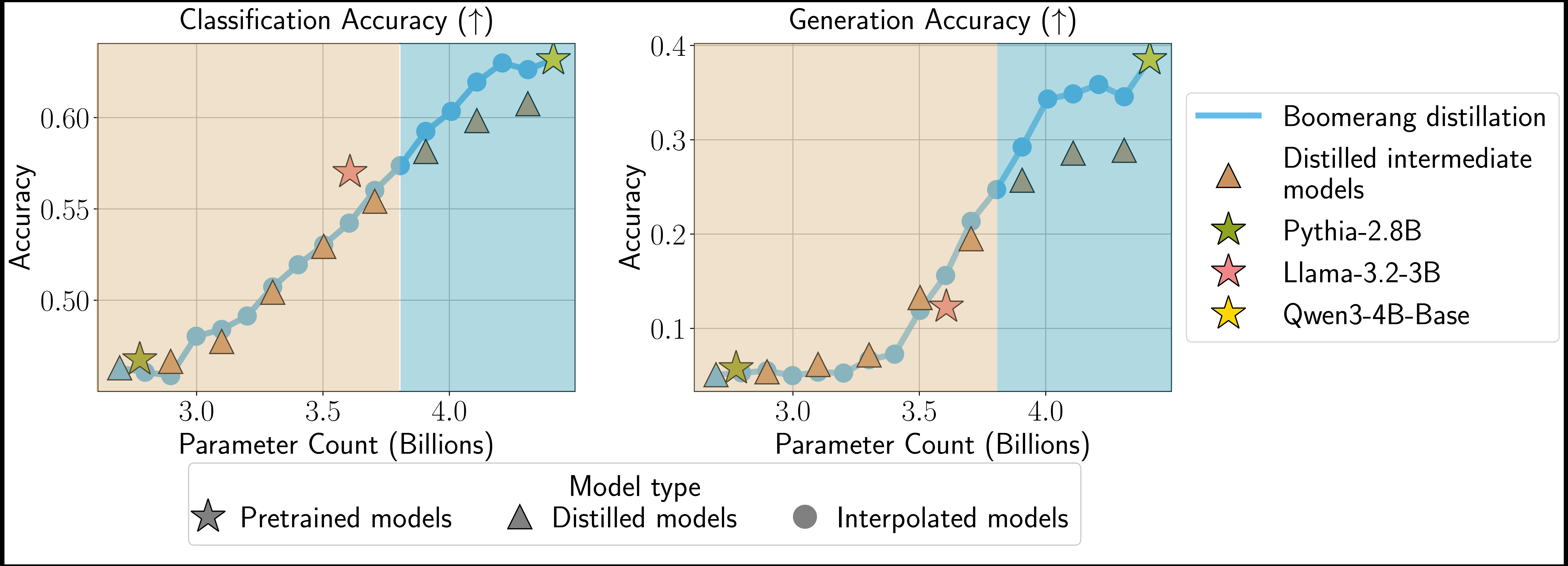
Distillation alone (without teacher initialization) cannot interpolate either

🪃 ⚗️ occurs across model families and sizes



🪃 ⚗️ matches intermediate distilled + pertained models

Setup: comparing boomerang distillation to standard distillation and pretrained models

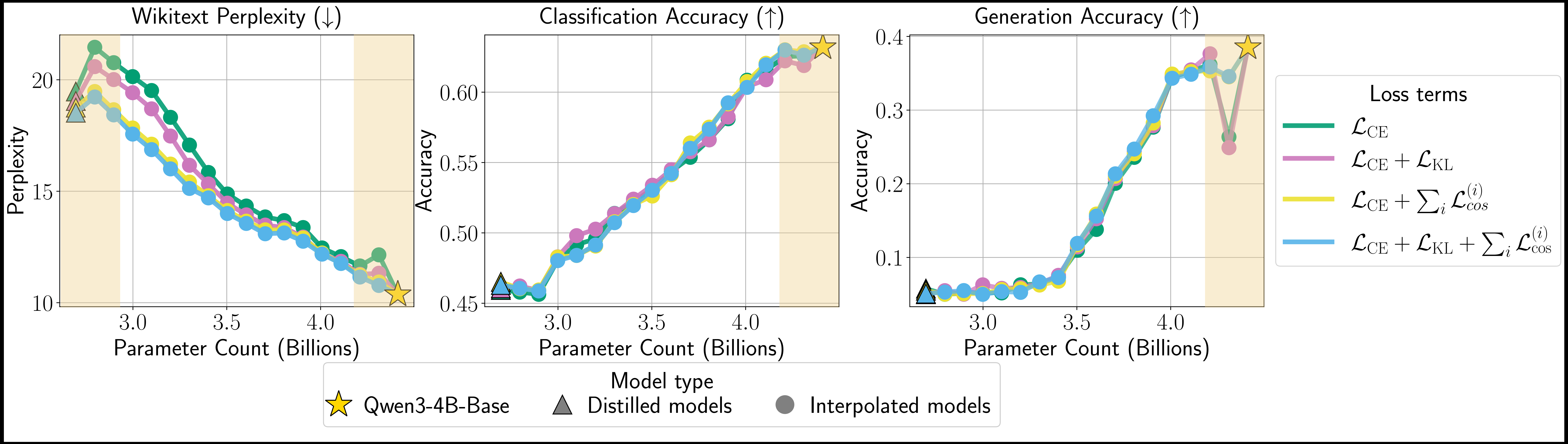


Interpolated models have similar performance to distilled models at small sizes

Interpolated models outperform larger distilled models

Layer alignment loss is crucial for stability

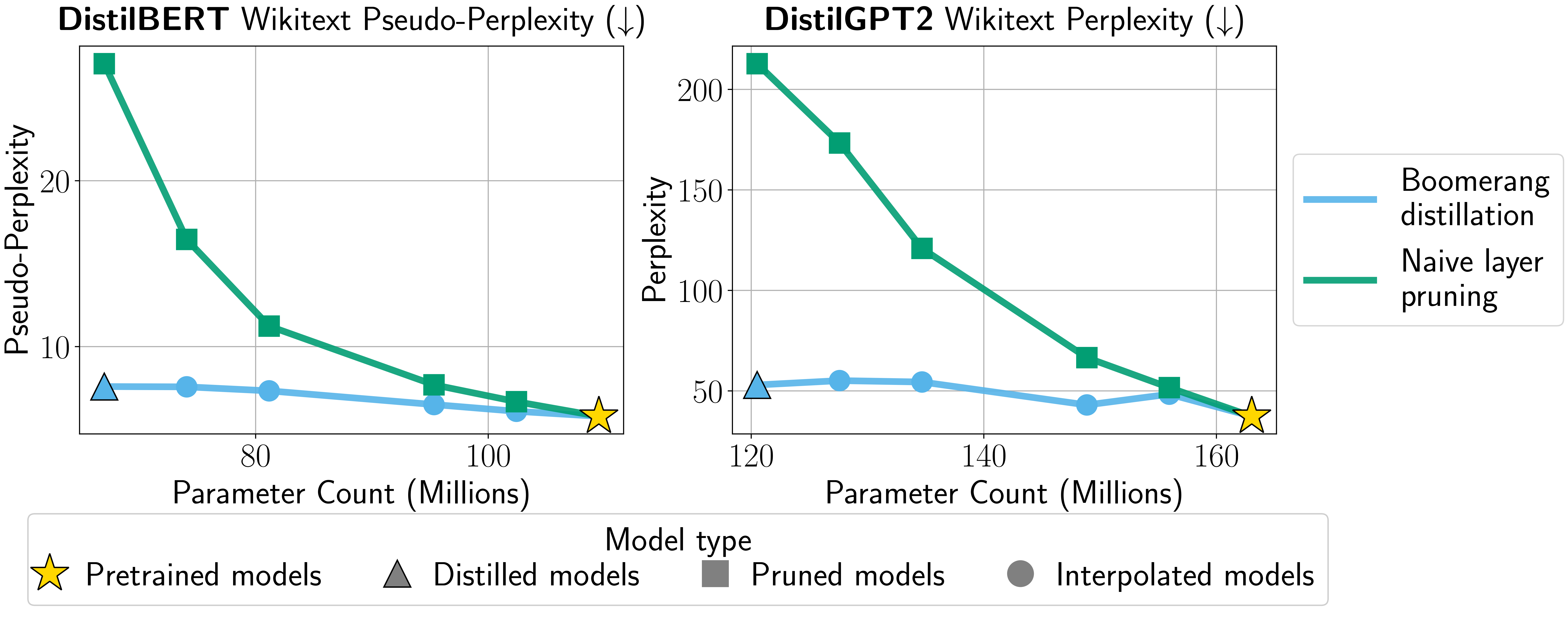
Setup: testing different combinations of loss terms during distillation



Alignment loss provides stability at interpolation extremes

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

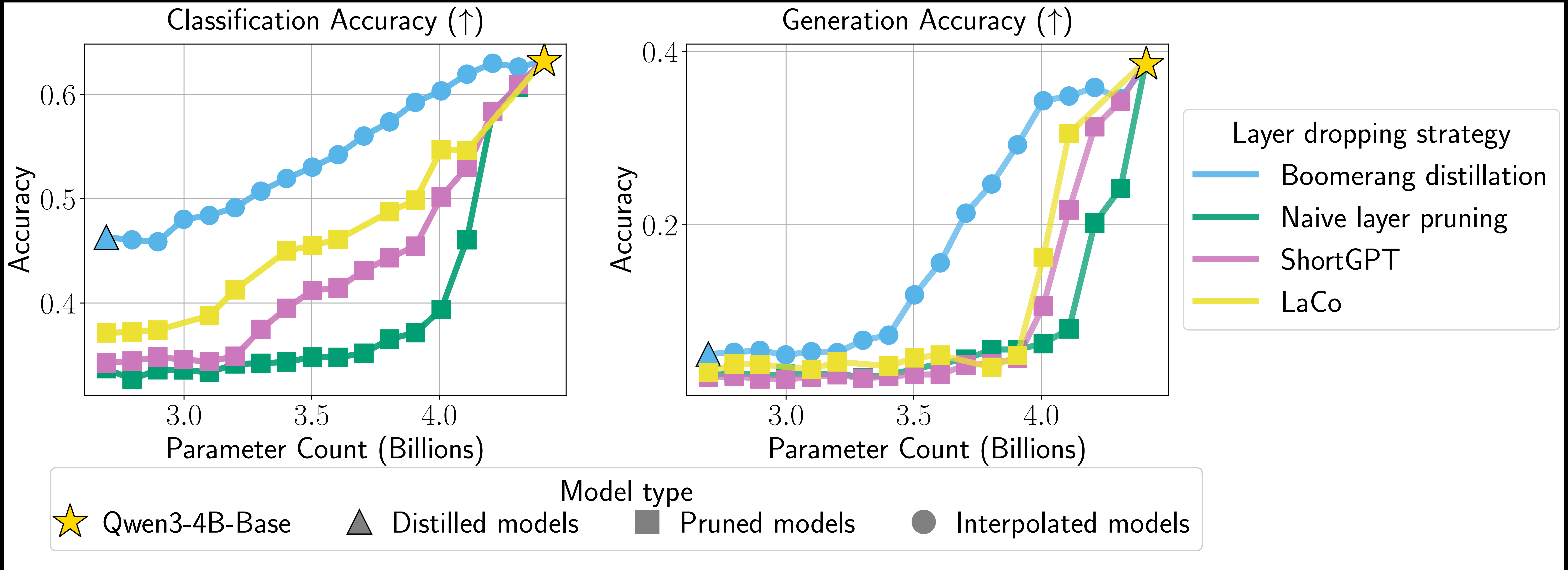
Setup: interpolation between DistilBERT and BERT and DistilGPT2 and GPT2



Boomerang distillation works here too!

🪃⚗️ significantly outperforms layer-dropping methods

Setup: comparison between interpolation and layer pruning methods

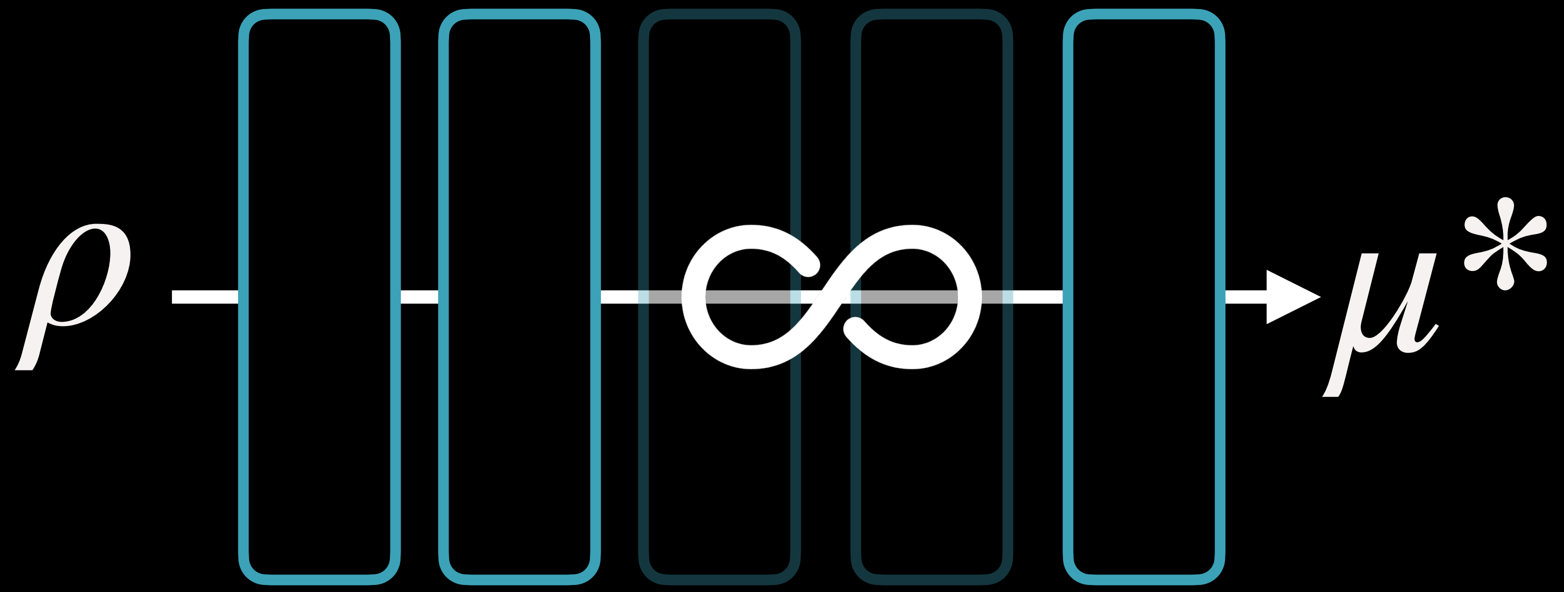


Boomerang distillation significantly outperforms baseline layer pruning methods, **especially in the high-compression regime**

🪃⚗️: 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

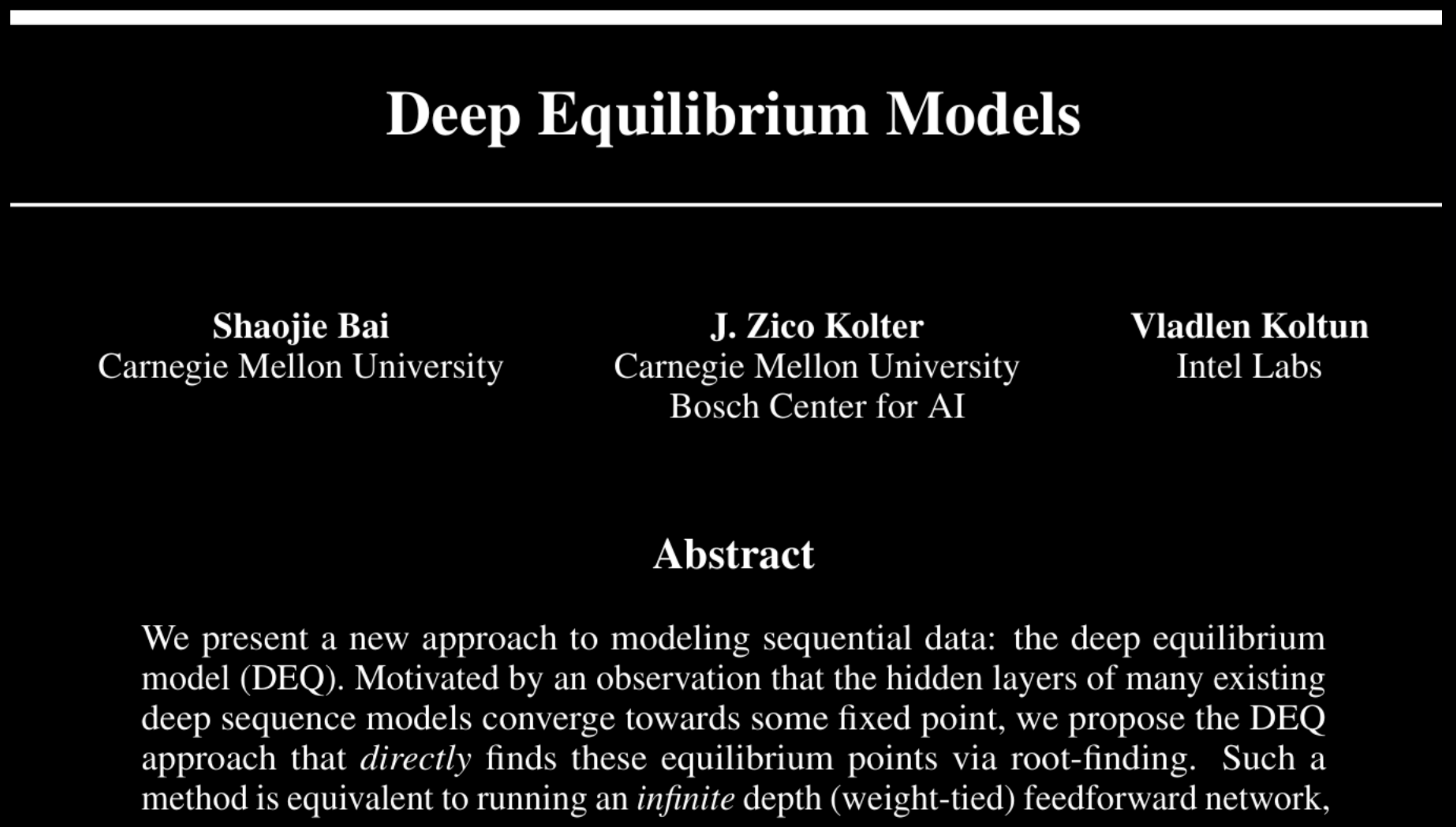
AISTATS 2025

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

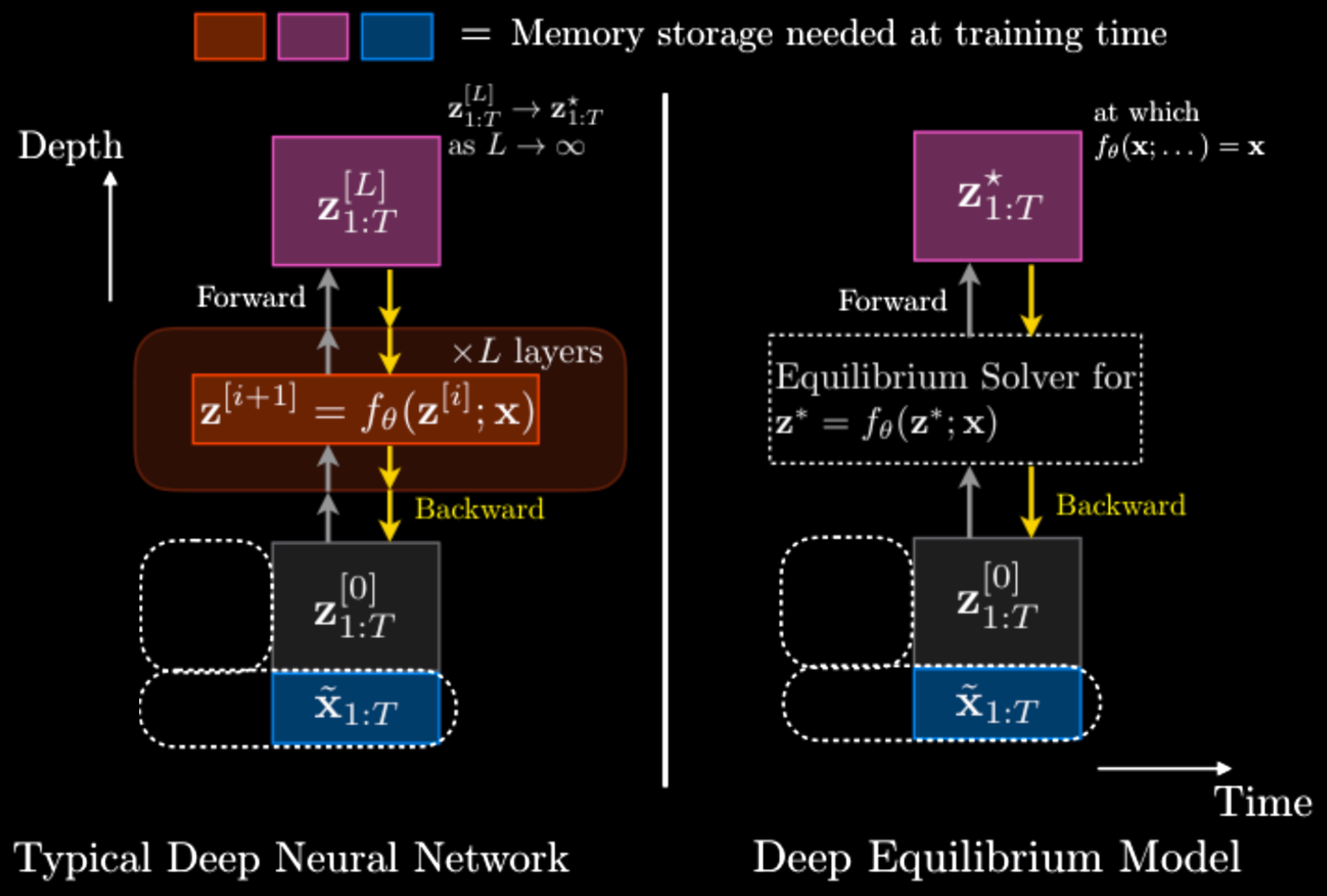
Shaojie Bai
Carnegie Mellon University

J. Zico Kolter
Carnegie Mellon University
Bosch Center for AI

Vladlen Koltun
Intel Labs

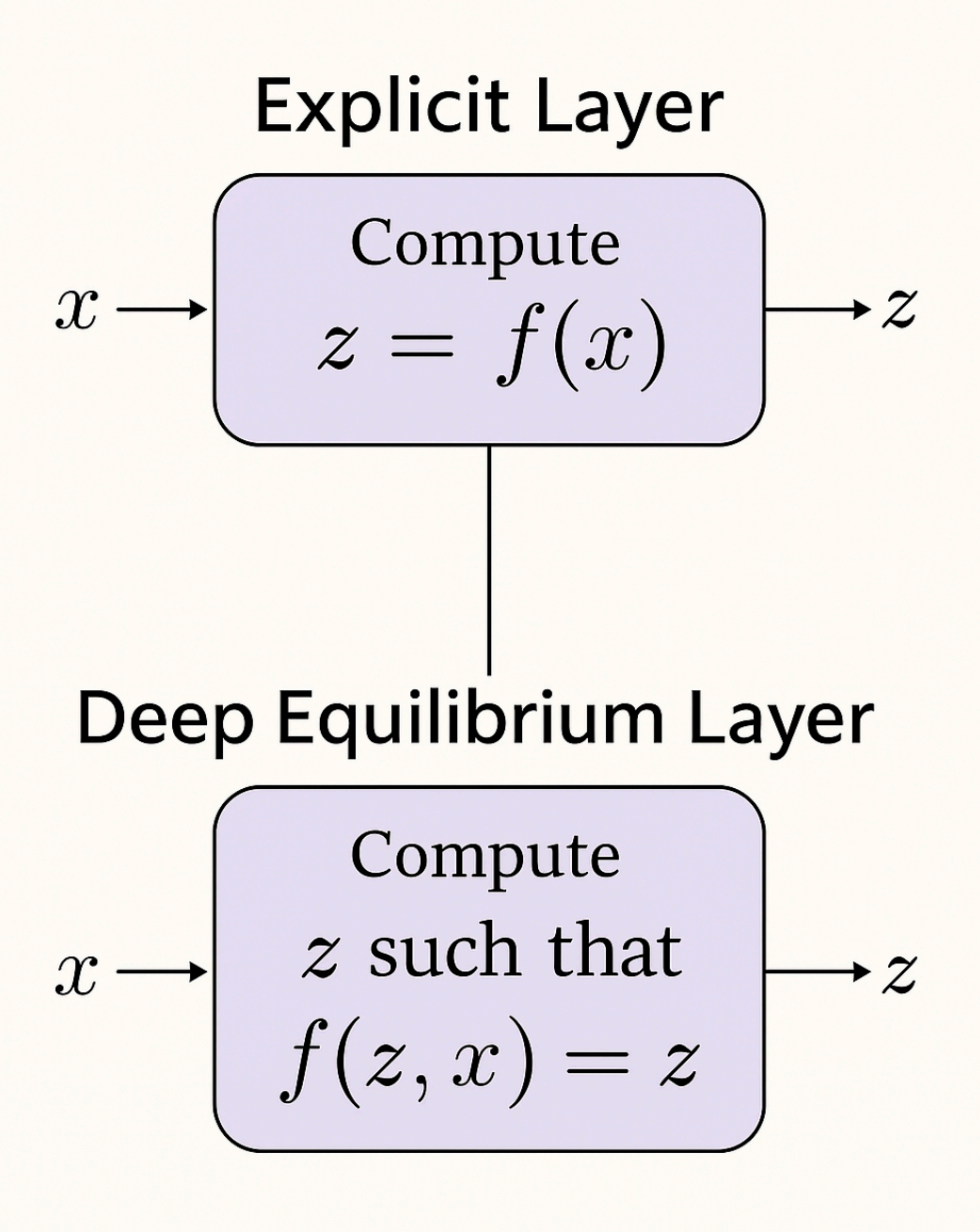
Abstract

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,


🟥 🟪 🟦 = Memory storage needed at training time
Depth
$\mathbf{z}^{[L]}_{1:T} \to \mathbf{z}^{\star}_{1:T}$
as $L \to \infty$
$\mathbf{z}^{[L]}_{1:T}$
Forward
$\times L$ layers
$\mathbf{z}^{[i+1]} = f_\theta(\mathbf{z}^{[i]}; \mathbf{x})$
Backward
$\mathbf{z}^{[0]}_{1:T}$
$\tilde{\mathbf{x}}_{1:T}$
Typical Deep Neural Network
at which
$f_\theta(\mathbf{x}; \dots) = \mathbf{x}$
$\mathbf{z}^{\star}_{1:T}$
Forward
Equilibrium Solver for
$\mathbf{z}^* = f_\theta(\mathbf{z}^*; \mathbf{x})$
Backward
$\mathbf{z}^{[0]}_{1:T}$
$\tilde{\mathbf{x}}_{1:T}$
Time
Deep Equilibrium Model

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**

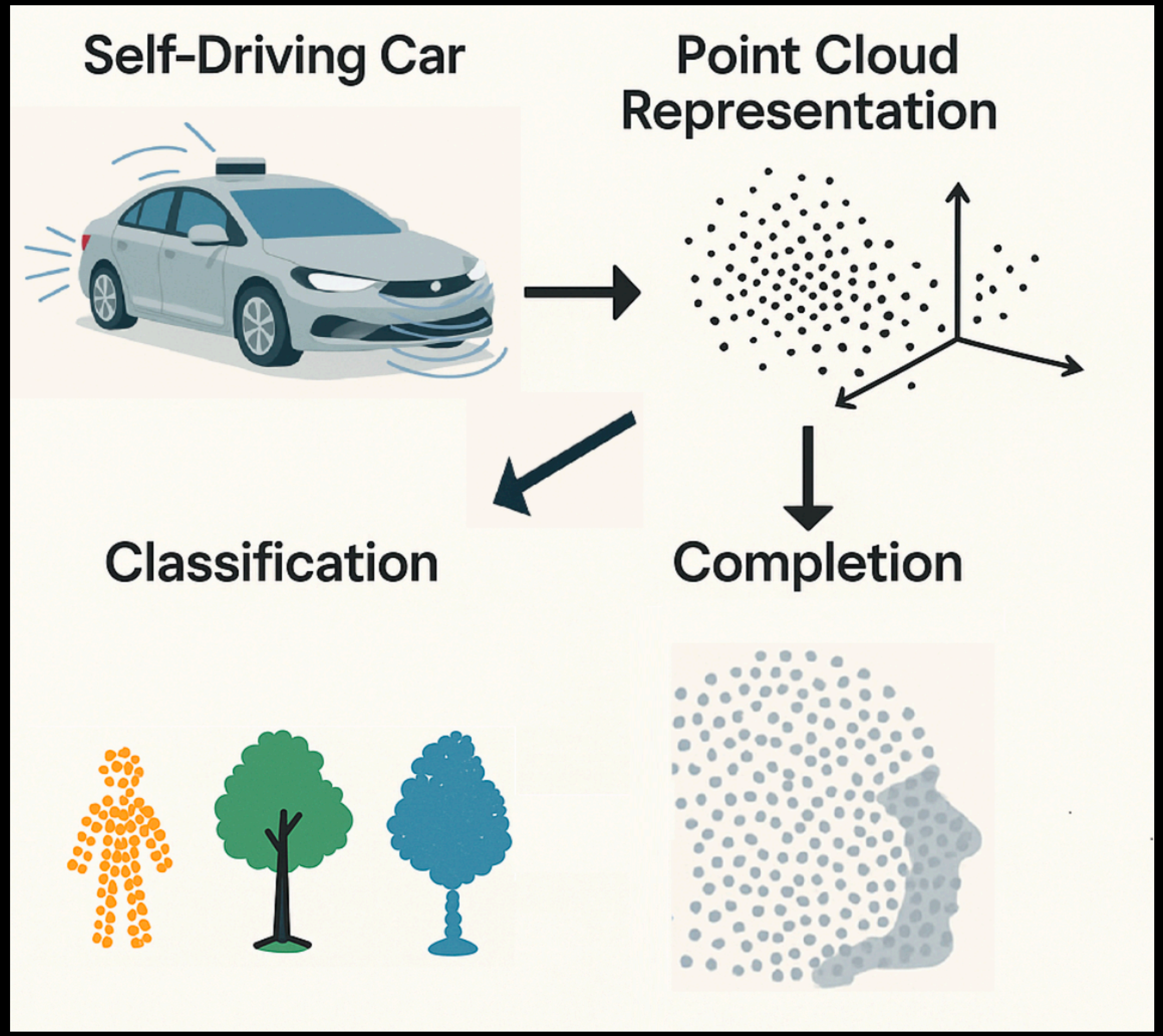
- Training/Inference:
  - Forward pass = find the fixed point (via root-finding algo, to desired tolerance)
  - Backward pass = one implicit gradient step (uses implicit f'n theorem), **no need to store intermediate activations**!

- 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

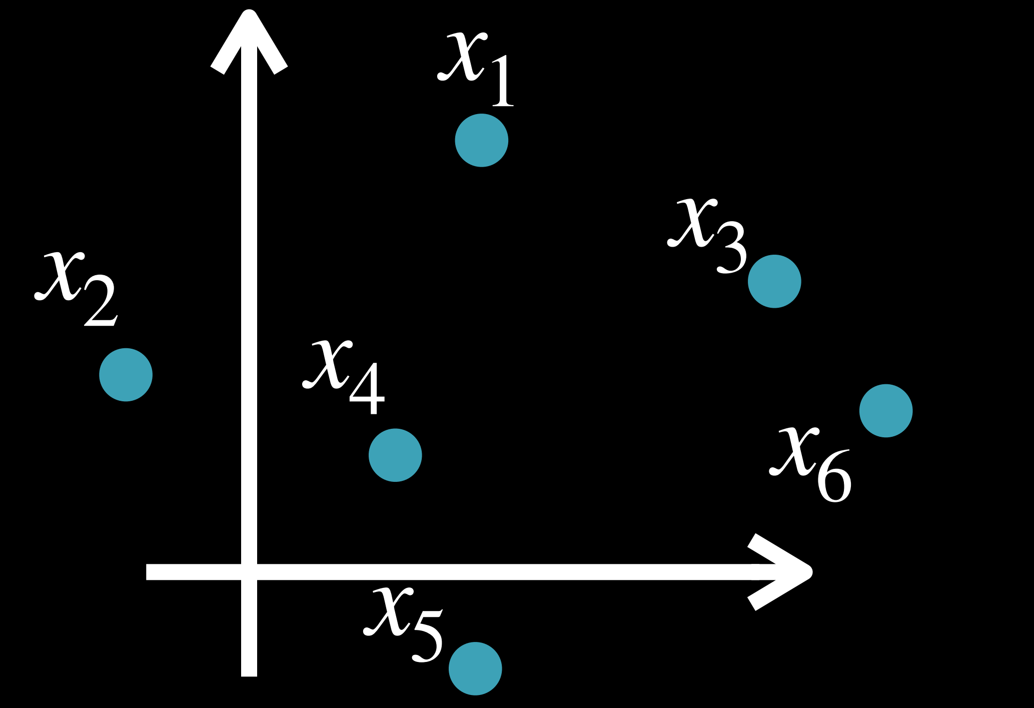
- Standard DEQs assume vector/matrix inputs/outputs of **fixed shape/dimensionality**
- But many real-world inputs are sets or distributions
  - e.g., point clouds, graphs, multi-particle systems
  - **variable size, permutation invariant**
- Need a model that operates **directly on sets/distributions**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



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

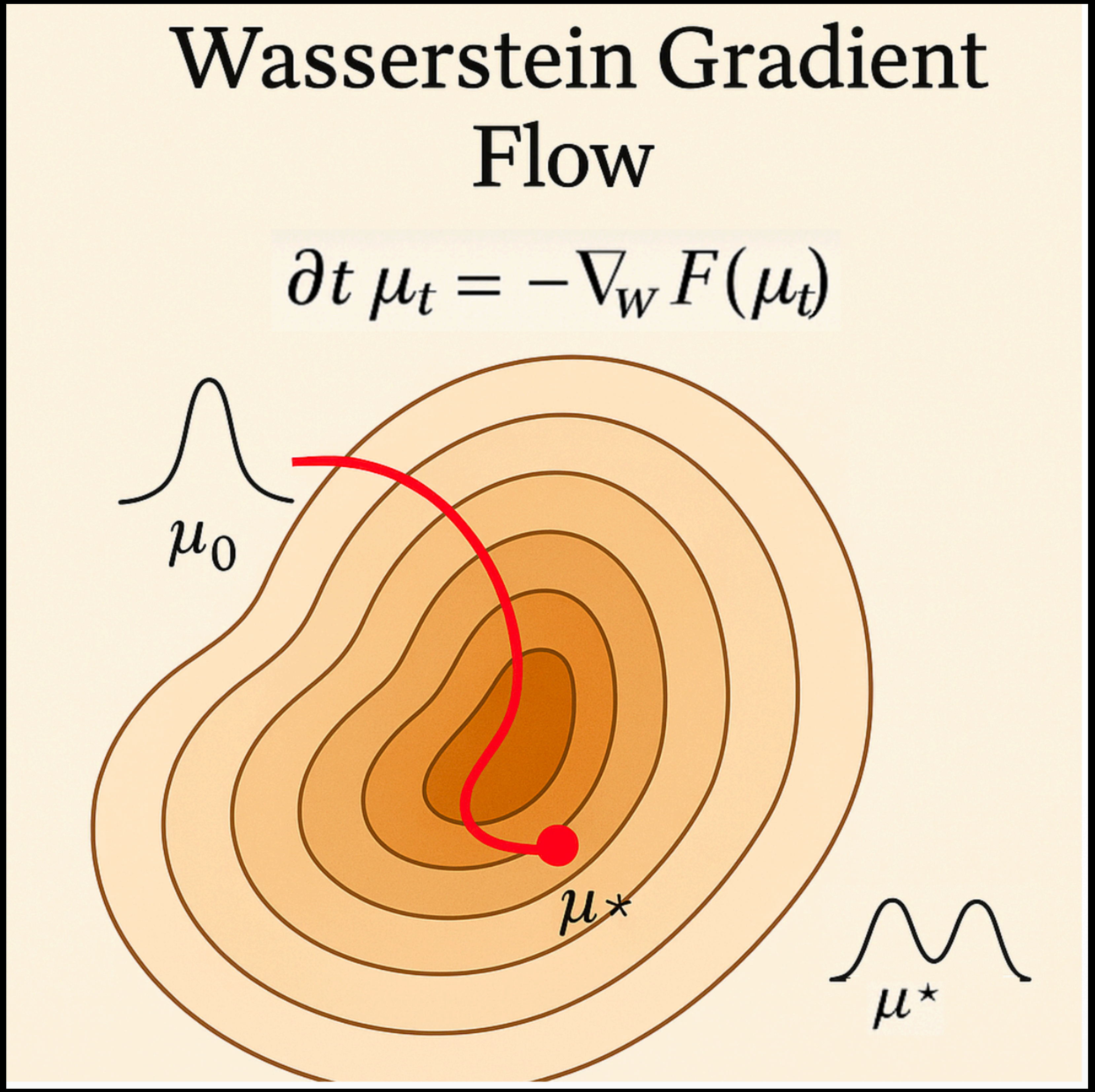
- Ingredients:
  - Data: $(\rho, y) \in \mathscr{P}_2(\mathbb{R}^d) \times \mathscr{Y} \sim P_{\text{data}}$
  - Latent measure: $\mu \in \mathscr{P}_2(\mathbb{R}^p)$
  - DDEQ layer: $F_\theta(\,\cdot\,, \rho) : \mathscr{P}_2(\mathbb{R}^p) \to \mathscr{P}_2(\mathbb{R}^p)$
  - “Post-processor”: $h_\theta : \mathscr{P}_2(\mathbb{R}^p) \to \mathscr{Y}$

- Given fixed point $\mu^*_{\theta,\rho} = F_\theta(\mu^*_{\theta,\rho}, \rho)$, define loss as: $\mathscr{L}(\theta) := \mathbb{E}_{\rho,y\sim P_{\text{data}}}\left[\ell\left(h_\theta(\mu^*_{\theta,\rho}), y\right)\right]$

- **How do we find the fixed point?**
  - Wasserstein Gradient Flow!

- **How to compute gradient**?
  - Implicit Gradient Theorem!

Wasserstein Gradient Flows


Wasserstein Gradient Flow
$\partial t\, \mu_t = -\nabla_W F(\mu_t)$
μ_0
$\mu_\star$
$\mu^\star$

- 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:** $\dot{x}(t) = -\nabla f(x(t))$

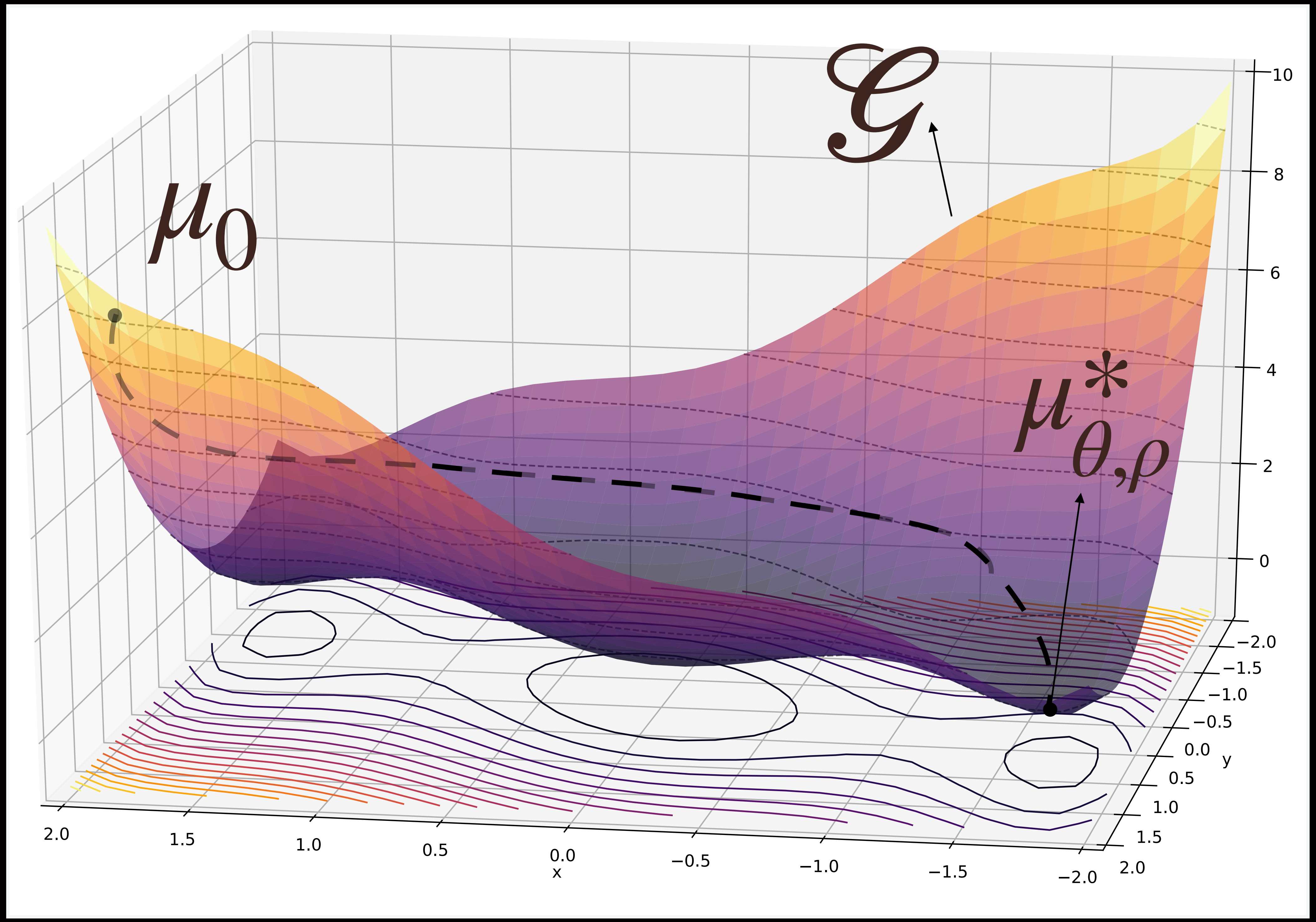
- This is **a curve $x(t)$ in** $\mathbb{R}^d$ that starts at $x(0) = x_0$ and moves at each instant in the direction of steepest descent of f

- Gradient flows can be defined over distributions via the Wasserstein gradient: $\nabla_W \triangleq \nabla \cdot (\rho \nabla \frac{\partial(\cdot)}{\partial \rho})$

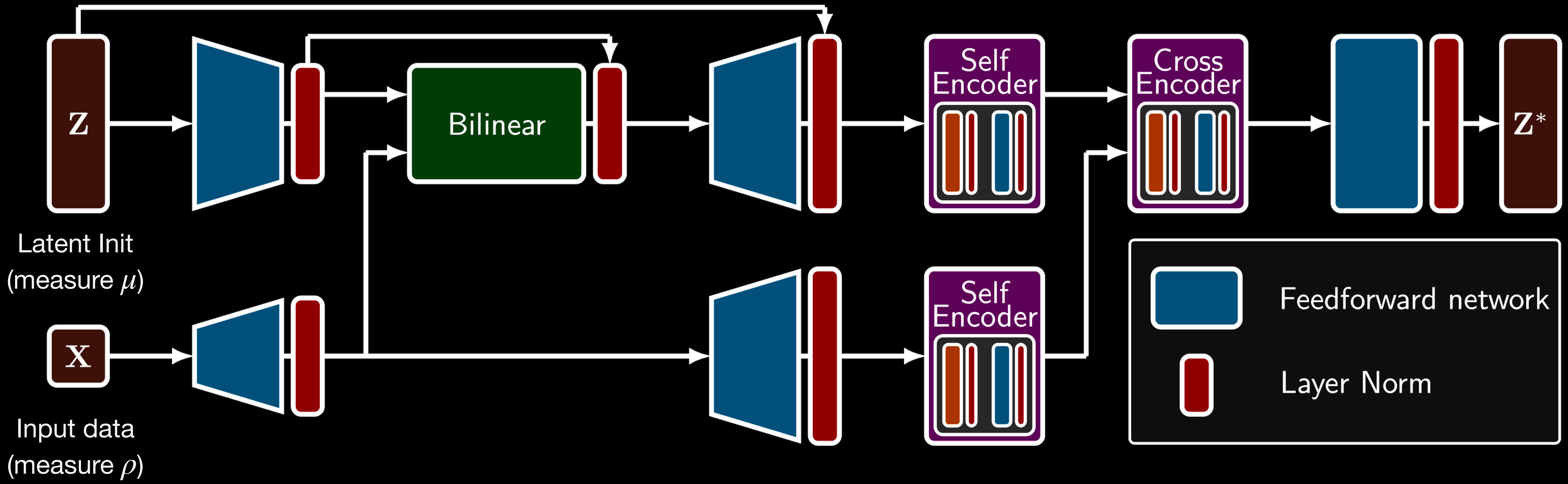
(e.g., Santambrogio 2016; Ambrosio, Gigli, Savare 2008)

Distributional Fixed Points

- Goal: Find the fixed point $\mu^*_{\theta,\rho} = F_\theta(\mu^*_{\theta,\rho}, \rho)$
- Challenge: $\mu^*_{\theta,\rho} - F_\theta(\mu^*_{\theta,\rho}, \rho)$ not well defined!
- Instead, can operationalize this as: $\mathrm{MMD}(\mu, F_\theta(\mu, \rho)) = 0$
- Let $\mathscr{G}(\mu) = \frac{1}{2}\mathrm{MMD}^2(\mu, F_\theta(\mu, \rho))$
- Finding fixed point of $F_\theta \Rightarrow$ following (time-discretized) Wasserstein gradient flow of $\mathscr{G}$

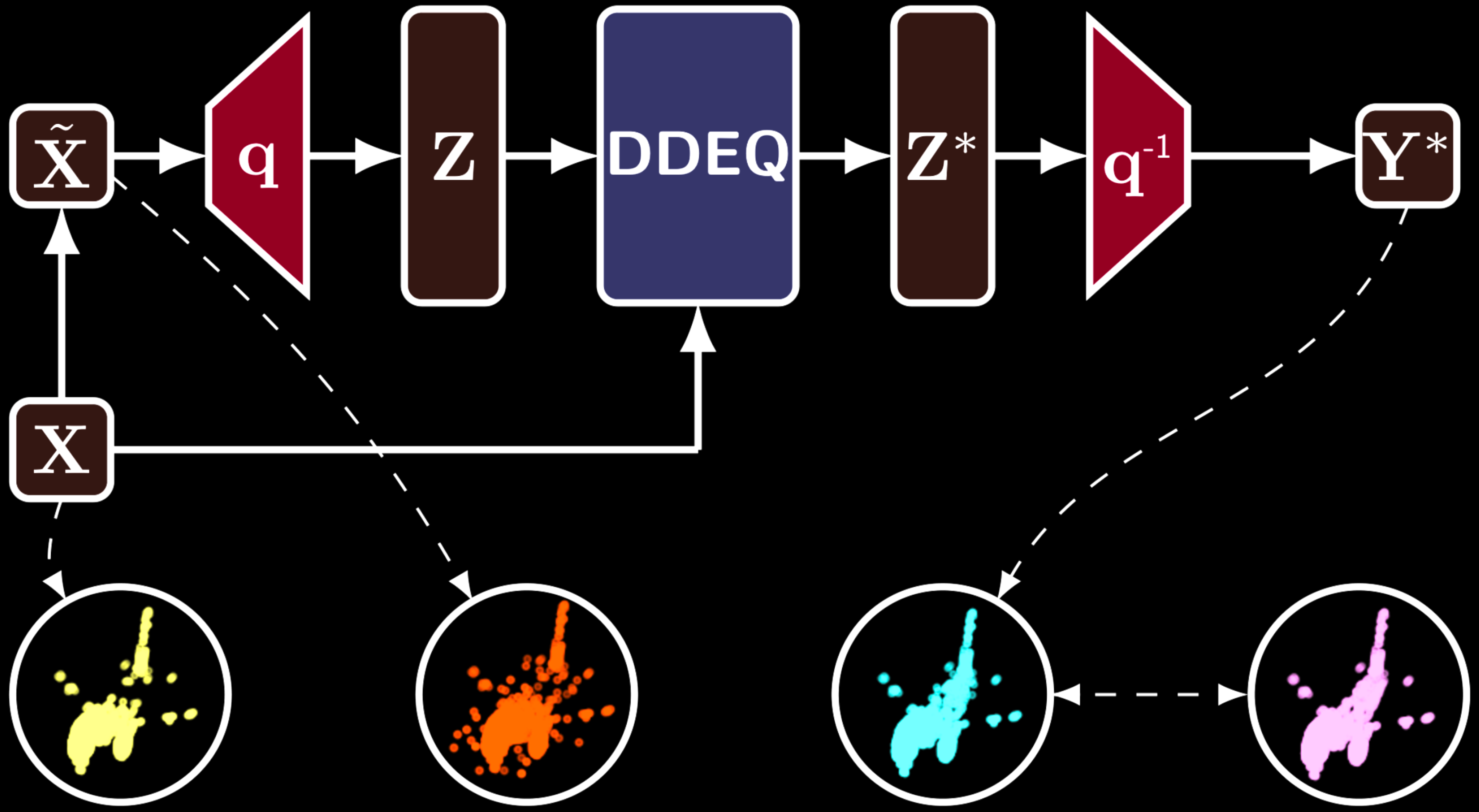


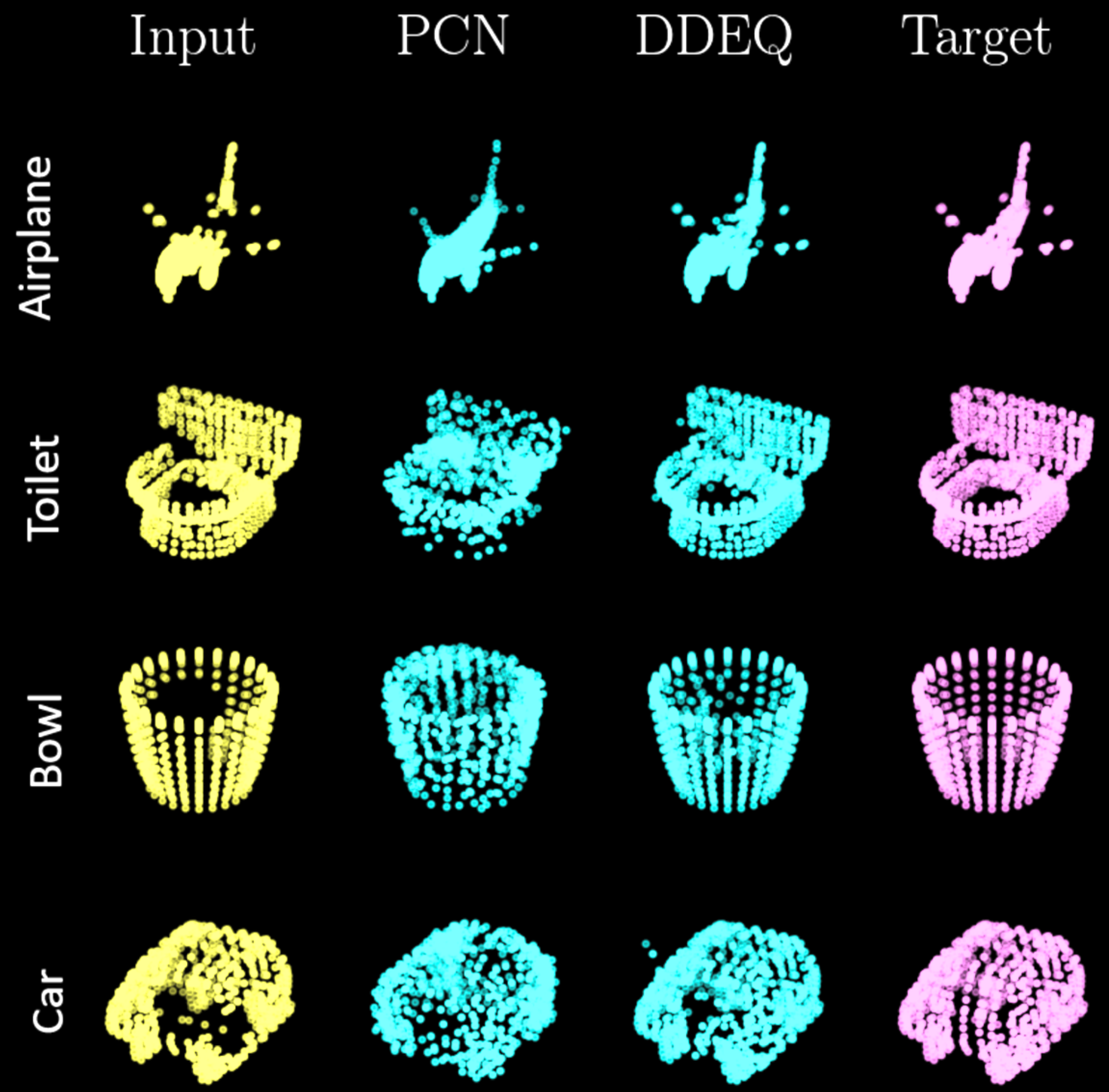
Distributional DEQ Architecture



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

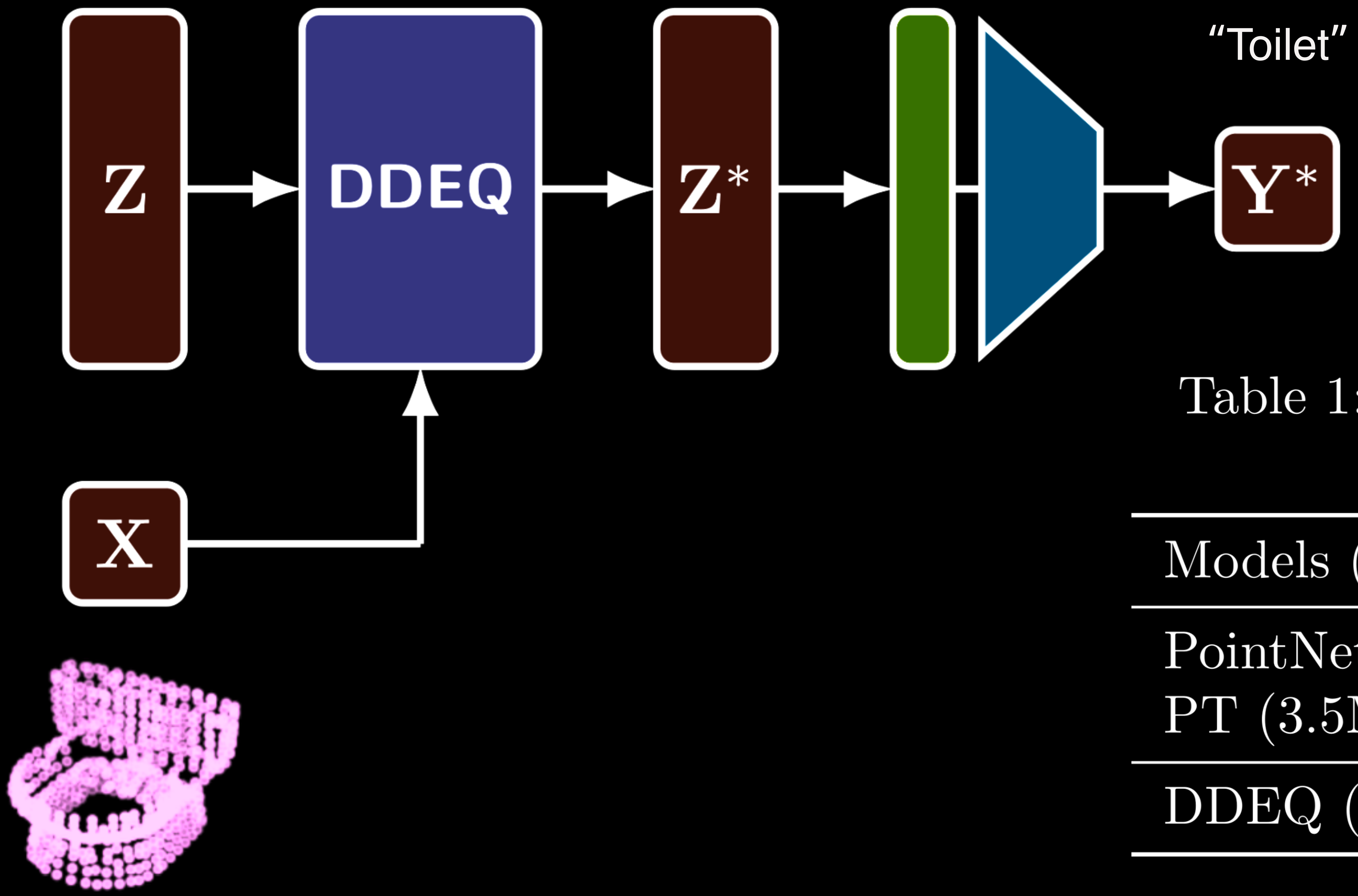


Table 1: Accuracies on Point Cloud Classification

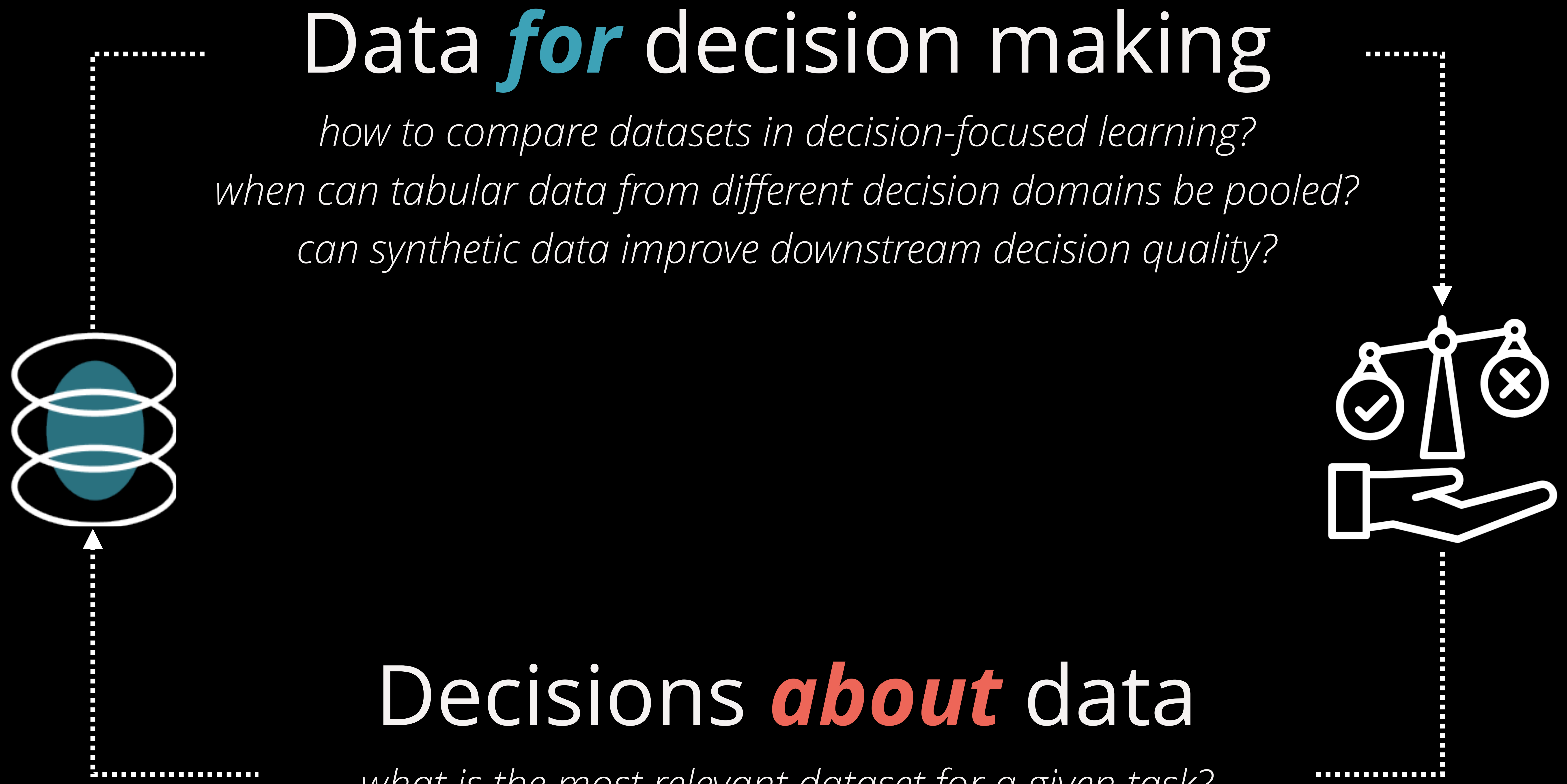
Models (size)	**MNIST-pc**	**ModelNet40-s**
PointNet (1.6M)	97.5	77.3
PT (3.5M)	98.6	79.2
DDEQ (776k/1.2M)	98.1	78.2

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)

Data-Centric Decision Making

Data Drives Decision Making



Data ***for*** decision making

how to compare datasets in decision-focused learning?
when can tabular data from different decision domains be pooled?
can synthetic data improve downstream decision quality?

Decisions ***about*** data

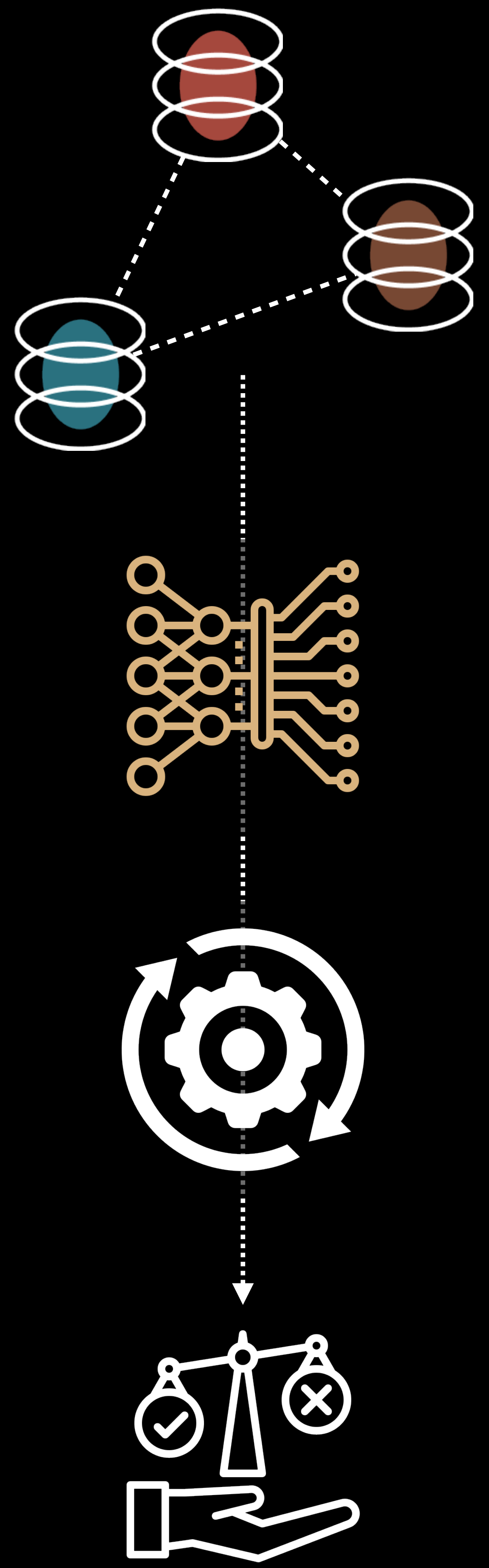
what is the most relevant dataset for a given task?
how much labeled data to collect?
train on the entire dataset or just a subset?

A Lesson From Interpolation

- The same principle (discrete choices to continuum limits) can help make data decision making principled, tractable, and reliable

- Two concrete instantiations our recent work:

1. Decision-aware dataset distance for predict-then-optimize transfer
2. Targeted instruction selection as query-aware subset choice



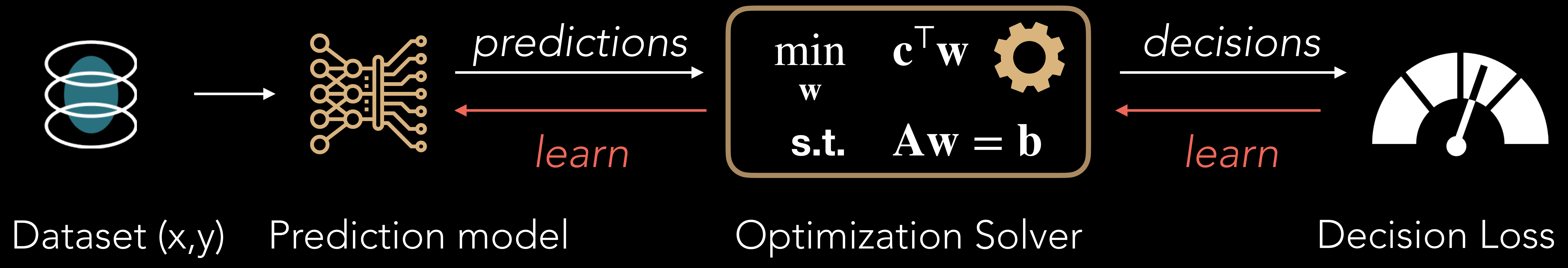
What is the Right Notion of Distance between Predict-then-Optimize Tasks?

Rodriguez-Diaz et al., UAI 2025

Paula Rodriguez Diaz

Decision-aware Dataset Distances

- **Setting**: Predict-then-optimize (PtO)



predictions

learn

$$\min_{\mathbf{w}} \quad \mathbf{c}^\top \mathbf{w} \qquad \text{s.t.} \quad \mathbf{A}\mathbf{w} = \mathbf{b}$$

decisions

learn

Dataset (x,y) Prediction model Optimization Solver Decision Loss

- **Goal**: transfer prediction. Given model trained on task A, how will it perform on task B?
- OT-based dataset distances are highly predictive of transferability — purely based on data geometry, model agnostic (AM & Fusi, ’20)
- But PtO tasks live in feature × label × decision space; how should we **incorporate decisions into similarity** computation?

Decision-aware Dataset Distances

- Our solution: inject decision information into Optimal Transport problem
- Ground cost for OT: pointwise distance between feature-label-decision triplets,

$$d((x,y,z),(x',y',z')) = \alpha_x d(x,x') + \alpha_y d(y,y') + \alpha_z d(z,z')$$

- Where $d(z,z')$ = *decision quality disparity*: difference in objective value achieved by two decisions when evaluated on the same true labels.
- Highly predictive of transferability, much more than decision-agnostic OT

Decision -agnostic

Warcraft Shortest Path

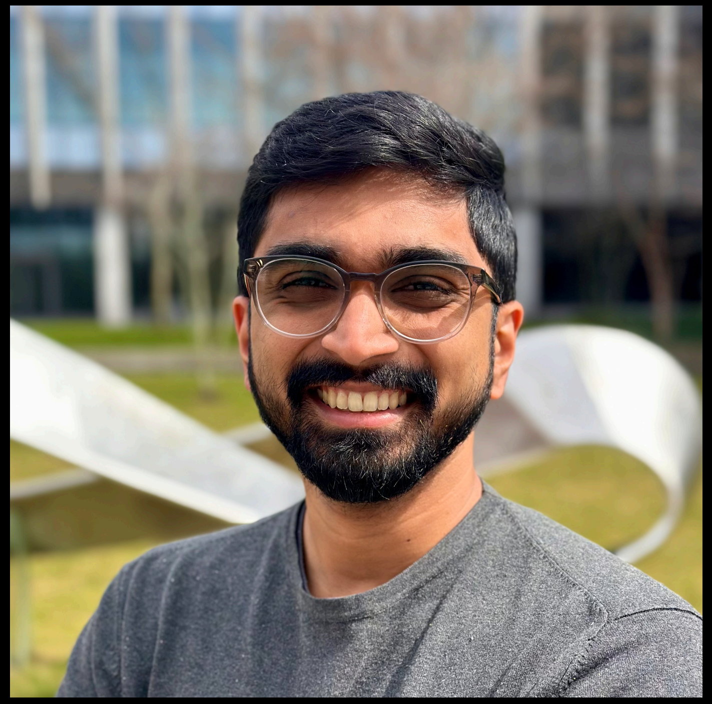
Inventory Stock

Linear TopK

Decision-aware

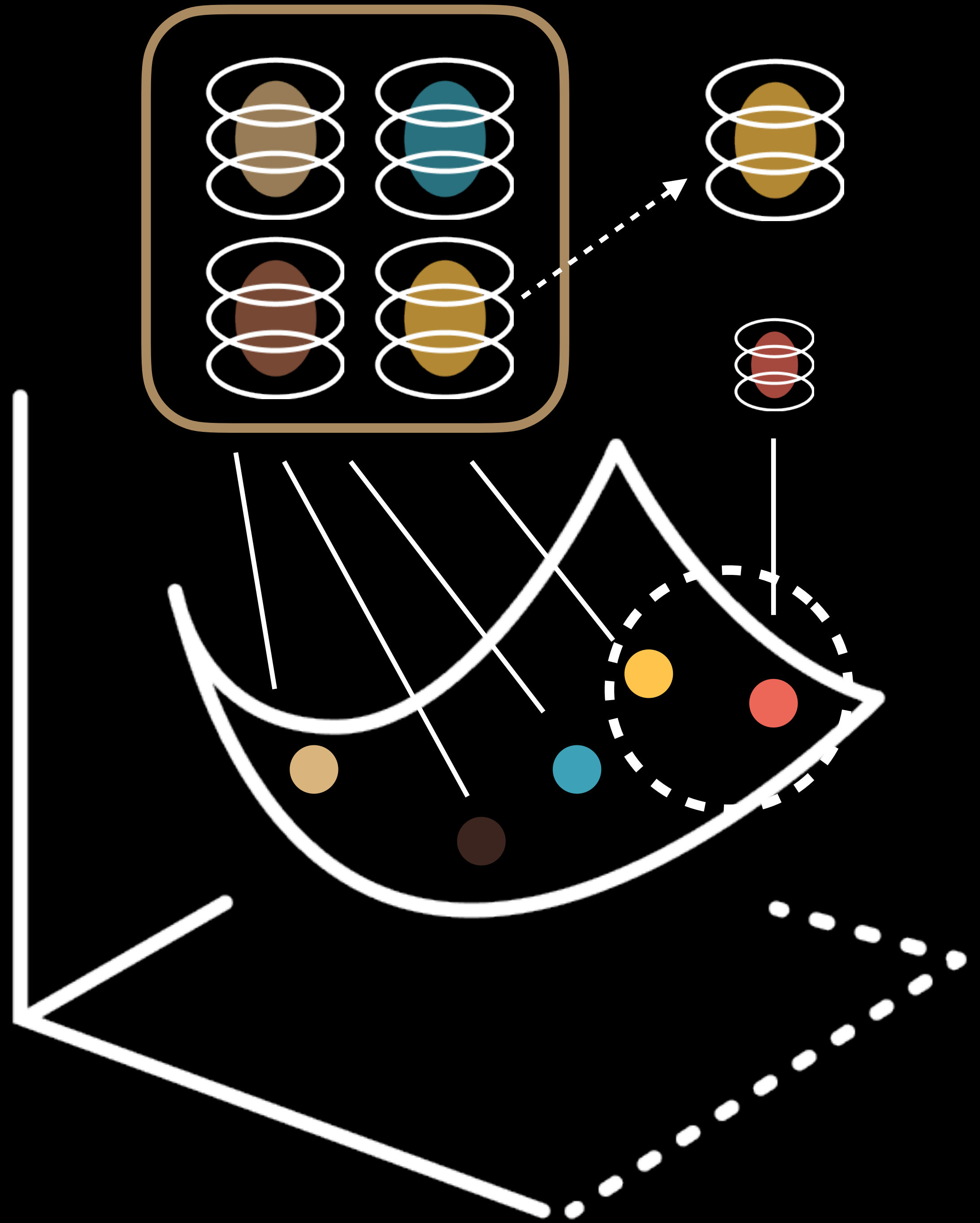
Transfer Error
200
150
100
50
0
0.70
0.75
0.80
R-squared: 0.42
0.64
0.66
0.68
R-squared: 0.78
0.20
0.15
0.10
0.05
0.00
0.55
0.60
R-squared: 0.77
0.45
0.50
R-squared: 0.85
6
4
2
0
−2
1.00
1.25
1.50
1.75
R-squared: 0.09
1.0
1.1
R-squared: 0.84

- **Key idea**: comparing datasets in the **geometry induced by downstream decisions**



Nihal Nayak

Paula Rodriguez Diaz



A Critical Look at Targeted Instruction Selection:

Disentangling What Matters (and What Doesn’t)

Nayak, Rodriguez-Diaz, Hulkund, Beery, AM, ArXiv 2026

Data Selection For Instruction Fine-Tuning

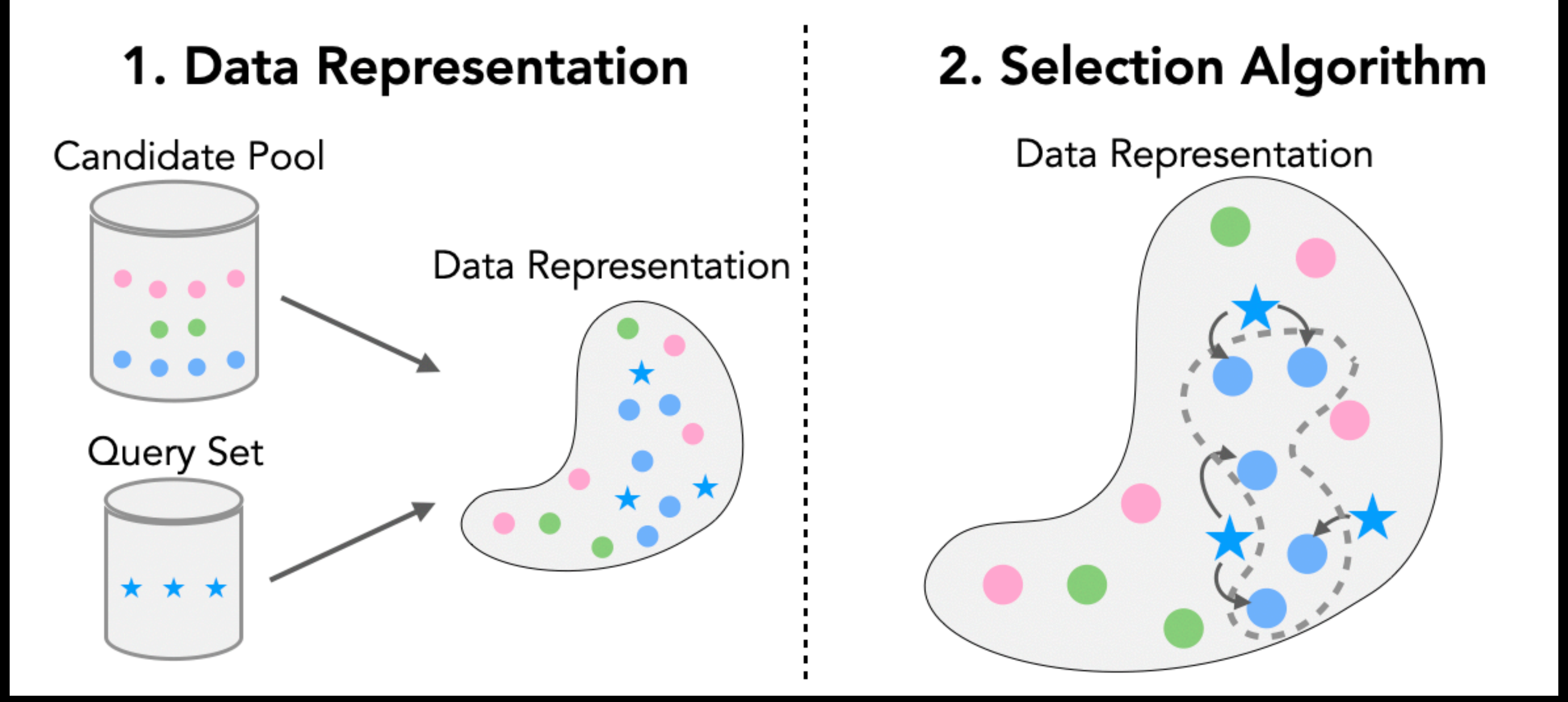
- LLM Fine-tuning is a data-selection problem: given a large candidate pool, **which examples should we train on** to get the most improvement per token/step?
- **Setup**: choose a subset S of size k from an instruction pool $\mathscr{C}$, guided by a small query set Q that represents the target use cases.
- **Decision objective**: maximize downstream performance on Q under a training data budget
- **Basic principle**: prioritize instructions that are most relevant to Q based on some notion of similarity in a latent representation space
- But the literature is fragmented, hard to compare—methods mix representation + selection choices, evaluations are inconsistent, and basic baselines missing

Data Selection For Instruction Fine-Tuning

- We disentangle the main steps of subset selection:

  i. **representation** of data points

  ii. **selection algorithm** over representations.



- This allows us to study effects in isolation, and make apples-to-apples comparisons
- **Findings**:
  - Most methods use criteria that are **very weakly (or not) correlated** with performance!
  - Only gradient-based representations correlate well, but even those yield inconsistent gains
  - Randomly sampled subsets often match or outperform many popular selection methods
  - Our proposed Optimal-Transport based method yields (modest) gains at large budgets
- A unifying perspective: We show many methods are forms of **approximate distributional distance minimization** between selected subset and query set → generalization bounds

Data-Centric Decision-Making Lessons

- **One unifying principle**: distributional dataset distances that turns discrete data choices into a controllable, near-continuous optimization problem.
  - Dataset choice for decisions (PtO): compare datasets in the **geometry induced by downstream decisions**/regret—use distance to pick sources and predict transfer.
  - Subset choice for adaptation (LLMs): targeted instruction selection can be viewed as **budget-constrained dataset distance minimization**.
- **Budgets become 'sliders'**: trade off data volume vs compute vs robustness; these methods can reveal optimal regimes, e.g., where extra data helps most.
- Enables more reliable transfer/deployment by **choosing what to trust/train on** (and when to adapt) using decision-aware similarity rather than heuristics.
- **Next steps**: dataset fusion for tabular data, synthetic data for decision making.

Overall 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. Data drives decision making.

5. Interpolating >> Choosing