Black-Box Optimization from Offline Datasets: Foundations to Recent Advances



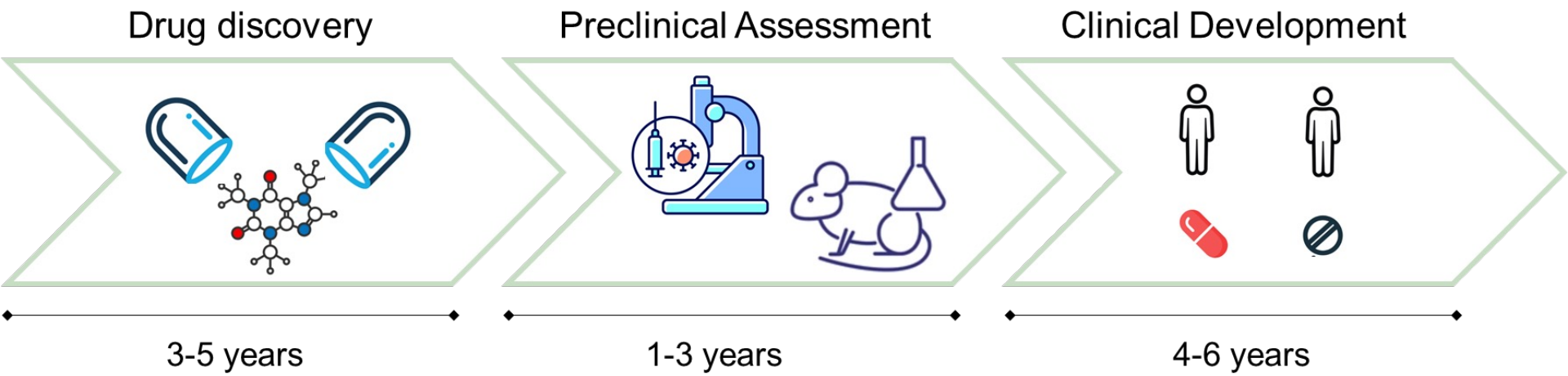


Jan 21 @ 8:30am (Room Opal 107)

Outline of Tutorial

- Introduction and Overview
- Forward Modeling Approaches
- Inverse Modeling Approaches
- Theoretical Analysis
- Offline Optimization in Small Data Setting
- Open Challenges

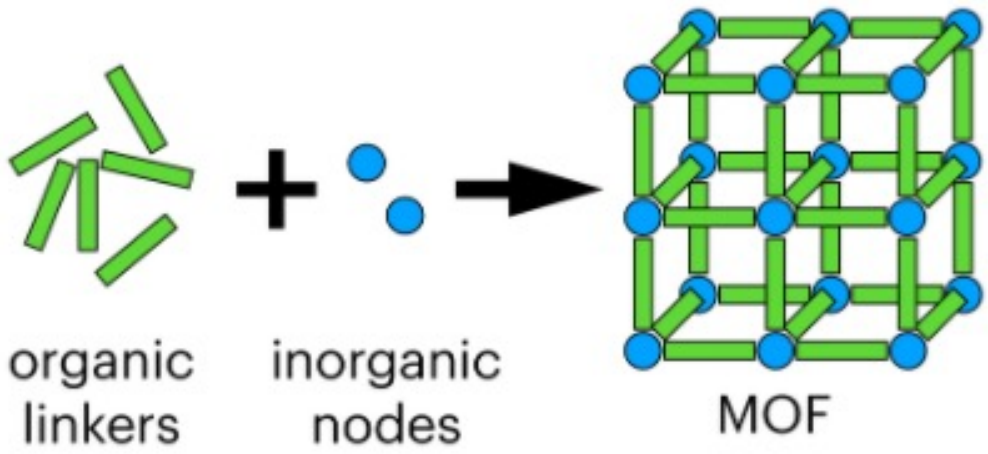
Applications: Drug Discovery



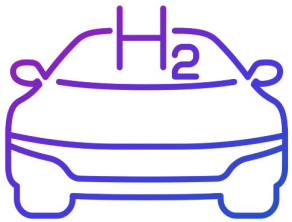
- Lot of costly experiments at every stage

* The Drug Development Process. U.S. Food and Drug Administration (FDA)

Applications: Nanoporous Materials Discovery

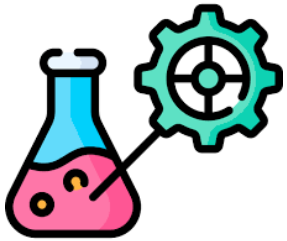


- Costly experiment to make and evaluate each MOF



Storing gases (e.g., hydrogen powered cars)

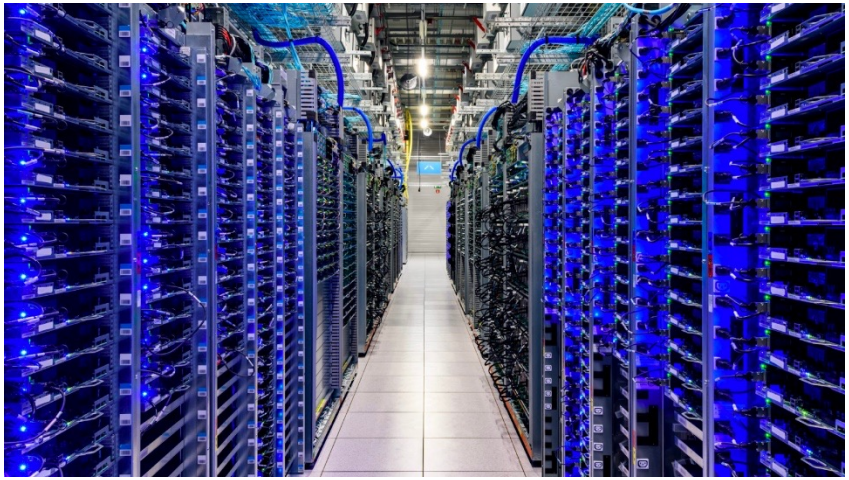
Capture gases (e.g., carbon dioxide power plants)



Catalysis, Drug Delivery etc.

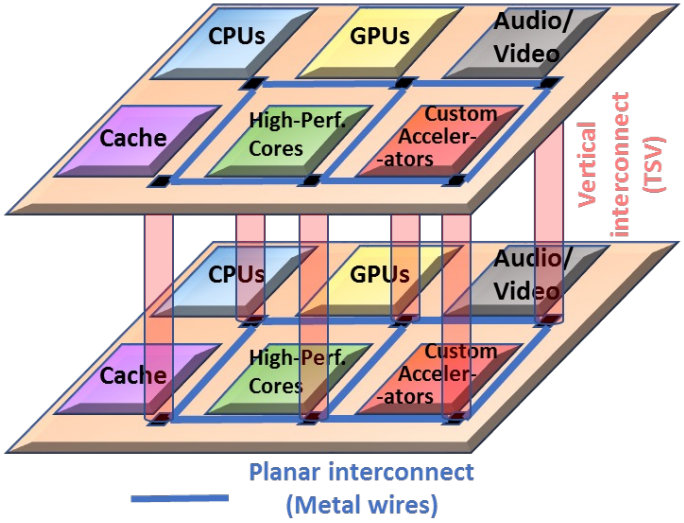
* Yaghi, Omar M. Reticular chemistry in all dimensions. ACS Central Science.

Applications: Next-Generation Hardware Design



America's Data Centers Are Wasting Huge Amounts of Energy*

* Report from Natural Resources Defense Council



Energy-efficient hardware accelerators

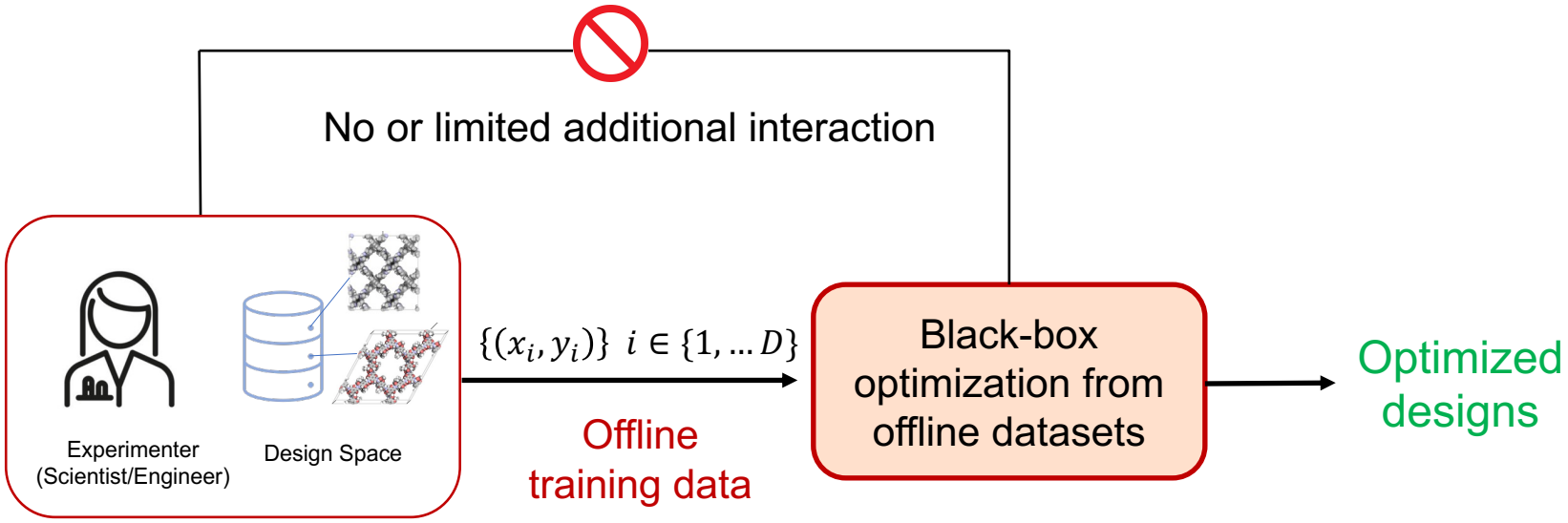
- Expensive computational simulations

Offline Optimization: Problem Definition

- **Given**
  - Black-box objective function
    - e.g., wet-lab experiments, real-world health interventions
  - Design space
    - Focus on high-dimensional or structured design spaces
    - e.g., space of proteins, materials, AutoML configurations
- **Goal:** Find (nearly) optimal designs with one or very few batches of evaluations using the objective function
  - "offline" refers to access to prior logged datasets
  - Sometimes also referred to as "data-driven"

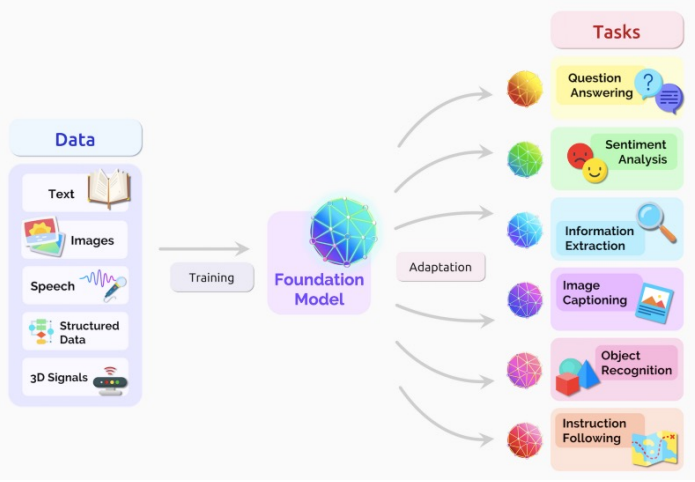
Offline Optimization: Problem Definition

- **Goal:** Find (nearly) optimal designs with one or very few batches of evaluations using the objective function
  - "offline" refers to access to prior logged datasets
  - Sometimes also referred to as "data-driven"



Motivation: AutoML for Foundation Models

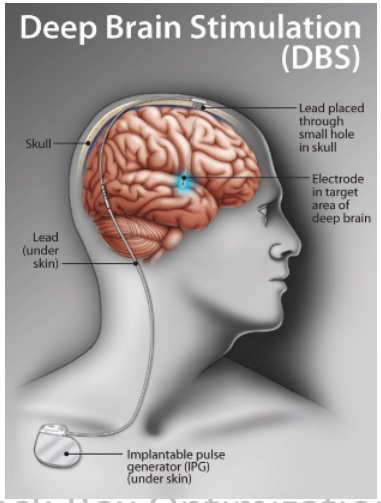
- Foundation models require significant resources for pretraining and finetuning
  - e.g., months of training and millions of dollars for pre-training
- **Goal:** find optimal configurations (e.g., hyper-parameters, data pipelines, data selection) while minimizing resources
  - Leverage "offline" data from prior configurations of in-house and/or open-source models



Credit: https://www.labellerr.com/blog/beginners-guide-using-foundation-models-in-ml-projects/

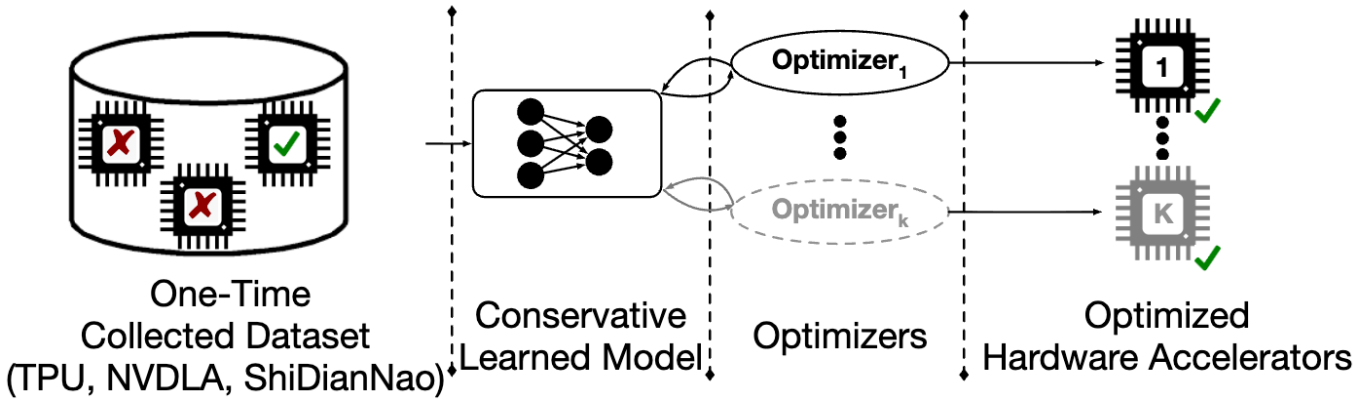
Motivation: Safe Healthcare Strategies

- Discover optimal parameters for healthcare devices
  - e.g., brain-modulation device
- Safety/regulation requirements prohibit online exploration
- **Goal:** find optimal parameters for continued home usage
  - Leverage "offline" data from logs of prior in-clinic usage under careful attention of experts/doctors



Motivation: Hardware Accelerators

- Growing need of application-specific hardware accelerators in shorter timescales
  - Potentially, avoid the use of time-consuming simulations
- **Goal:** find optimal application-specific hardware accelerators
  - Leverage "offline" data of previously tested accelerator designs



Kumar et al., Offline Optimization for Architecting Hardware Accelerators, ICLR 2022

Motivation: Climate-Smart Agriculture

- Growing need for adoption of climate smart strategies
- Excessively long timeframes for farmers to see agricultural yield (objective) based on chosen strategy (search space)
- **Goal:** find optimal climate-smart designs from previously tested strategies

Many Science and Engineering Applications

- Biological sequence design
  - Protein sequence design
    - Takizawa et al., Safe model-based optimization balancing exploration and reliability for protein sequence design
  - Cell-type promoter sequence design
    - Reddy et al., Designing Cell-Type-Specific Promoter Sequences Using Conservative Model-Based Optimization
- Antibody design
- Nanoporous materials design
- …

Offline Optimization vs. Online Optimization

- Offline Optimization: Find (nearly) optimal designs with one or very few batches of evaluations via objective function
  - "offline" refers to access to prior logged datasets
- Online Optimization: has access to ***large*** iterative rounds of access to true objective function
- Relevance of Offline setting to Online setting
  - Selection of initial batch (e.g., aSCR and LEON)
  - Last round of online optimization
  - Problem setting with large batches per evaluation round

*aSCR: Generative Adversarial Model-Based Optimization via Source Critic Regularization, NeurIPS 2024

*LEON: Knowledgeable Language Models as Black-Box Optimizers for Personalized Medicine, arXiv:2509.20975

Bayesian Approaches for Black-box Optimization

- Bayesian approaches (BO) are SOTA for online optimization
- Bayesian decision theory allows optimal decision making!
  - One round optimal decision-making is “Expected Improvement”
- These approaches rely on Bayesian inference or principled uncertainty quantification
  - Active research area in high-dimensional design spaces
- Key BO references relevant to this tutorial:
  - TurBO: Scalable Global Optimization via Local Bayesian Optimization (NeurIPS-19)
  - FocalBO: Scalable Bayesian Optimization via Focalized Sparse GPs (NeurIPS-24)

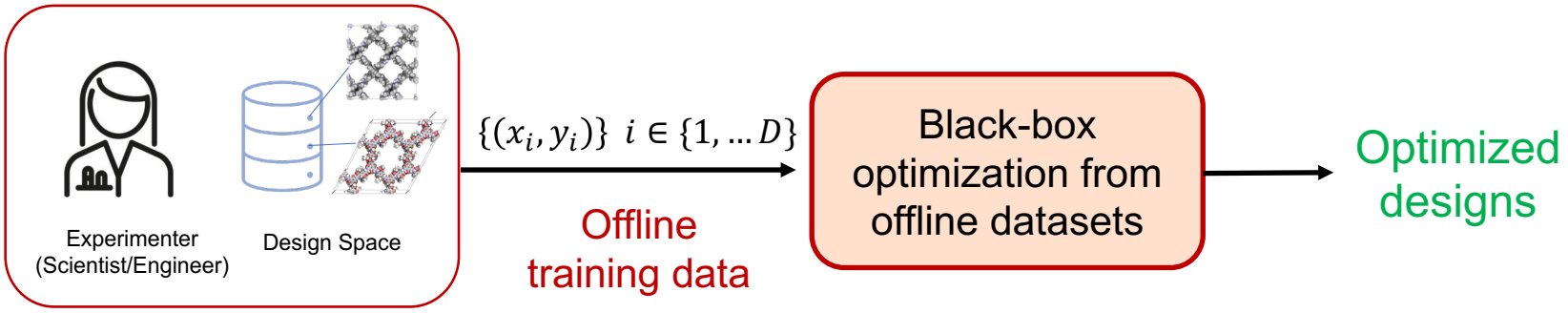
Nuances of Online Optimization

- (Physics/Knowledge-based) Digital Simulators
  - Expensive in terms of computational resources
    - e.g., physics simulations, hardware simulator, crop Simulator
  - **Pros:** Can be evaluated online for multiple rounds
  - **Cons:** Not always available and still a proxy (i.e., applicable within knowledge-domain)
- Physical Real-world Experiments
  - Significantly more expensive in multitude of resources (e.g., computational, human, monetary, regulatory)
    - e.g., wet-lab experimentation (weeks/months), making a chip in a fab (months), real-world agriculture practices (months/years), Foundation models (months)
  - Prior data from human trial-and-error approaches is available

Outline of Tutorial

- Introduction and Overview
- Forward Modeling Approaches
- Inverse Modeling Approaches
- Theoretical Analysis
- Offline Optimization in Small Data Setting
- Open Challenges

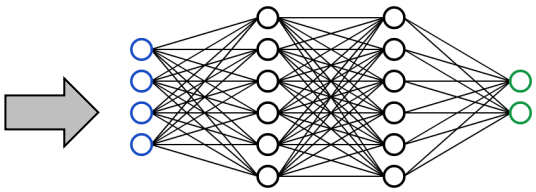
Forward Modeling: Big Picture



- **Step 1:** Training a surrogate model

$$\{(x_i, y_i)\}\ i \in \{1, \ldots D\}$$

Offline training data



- **Step 2:** Gradient based search from best training designs

Forward Modeling: Big Picture

- Focus on (1) modeling surrogate:

$$g(\text{design}) = \text{performance}$$
$$\mathbf{x} \qquad\qquad z$$

and/or (2) search procedure:

$$\Pi(\text{design}; g) = \text{design update}$$
$$\mathbf{x} \qquad\qquad \Delta\mathbf{x}$$

Challenge: $g(\mathbf{x})$ is erratic when $\mathbf{x}$ is far from offline data

Forward Modeling: Big Picture

- Focus on (1) modeling surrogate:

$$g(\text{design}) = \text{performance}$$
$$\mathbf{x} \qquad\qquad z$$

and/or (2) search procedure:

$$\Pi(\text{design}; g) = \text{design update}$$
$$\mathbf{x} \qquad\qquad \Delta\mathbf{x}$$

$\Pi(\mathbf{x}; g)$: estimate the trust region where $g(\mathbf{x})$ is reliable

and avoid venturing outside it

Forward Modeling Approaches: Summary

1. **Conservative Surrogate.** Fit a surrogate to offline data while decreasing its prediction for OOD (out of distribution) designs

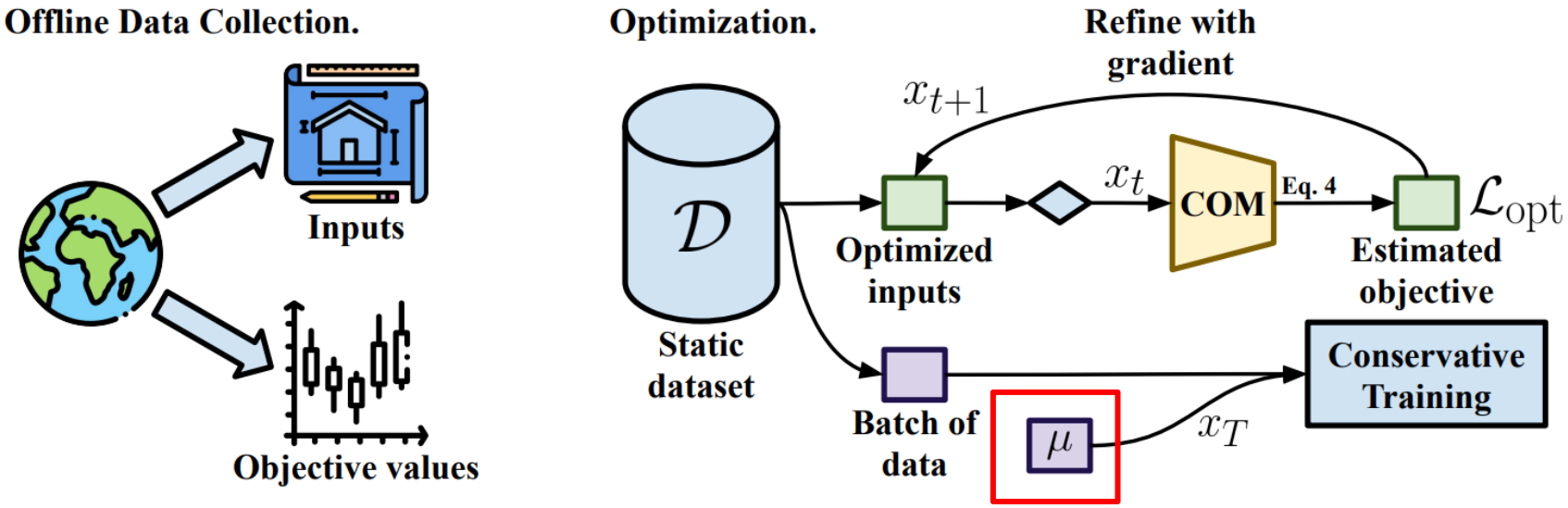
2. **Smoothness-aware Surrogate Ensemble.** Fit a neighborhood of surrogates to offline data & perform search at each step with one that is smoothest at the current design

3. **Minimizing Surrogate Sensitivity.** Fitting a surrogate while minimizing output sensitivity under random model perturbations

4. **Using In-Distribution Critic.** Fitting a surrogate and finding its optima where the critic determines in-distribution with high confidence

5. **Gradient Matching.** Learning gradient curvature instead of matching objective function output (relative ordering is all that matters!)

COMS: Conservative Surrogate [Trabucco et al., 2021]



OOD set collected via early gradient ascent on g(x; θ)

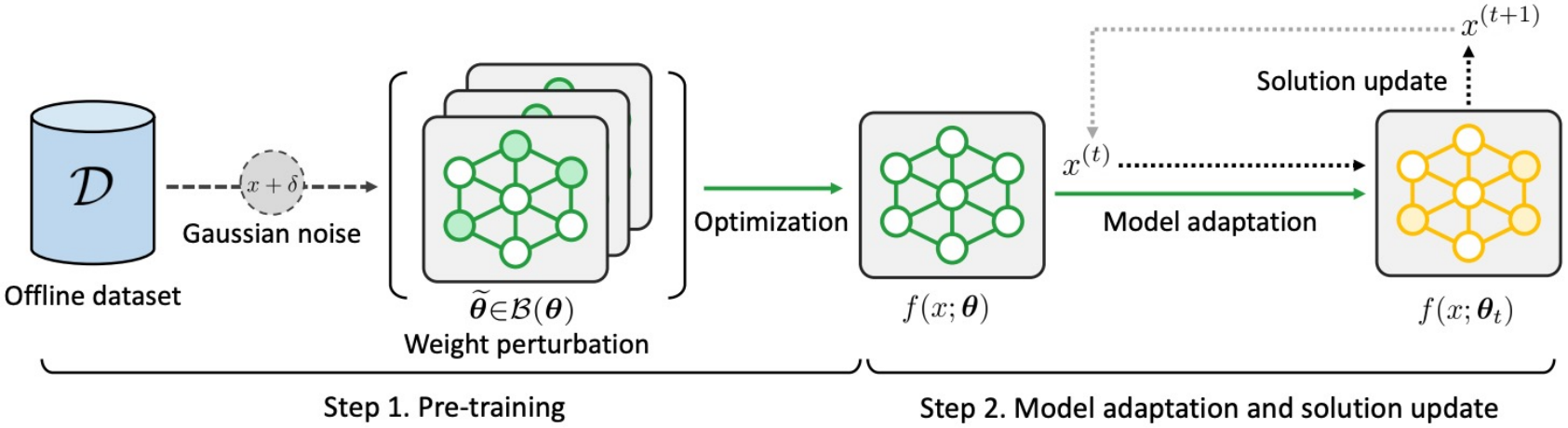
standard MSE

$$\theta = \text{argmin } E_{(\mathbf{x}, z) \sim D}[\,(g(\mathbf{x}; \theta) - z)^2\,] +$$

$$2\alpha(E_{\mathbf{x}\sim\mu}[g(\mathbf{x}; \theta)] - E_{\mathbf{x}\sim D}[g(\mathbf{x}; \theta)])$$

OOD average is discouraged to be larger than in-distribution average

ROMA: Smoothness-Aware Surrogate Ensemble



Ensemble loss

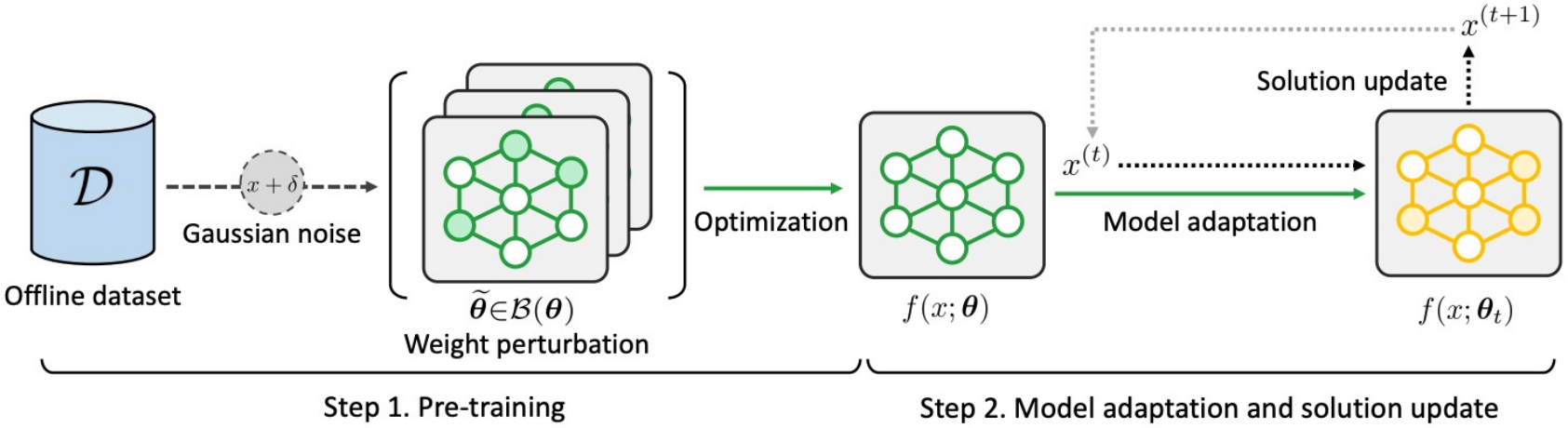
(prediction loss of the worst surrogate in the neighborhood)

$$L(\theta) = \max_{\theta' \in \mathbf{B}(\theta)} \; E_{(\mathbf{x},\, z)\sim D}\, E_{\mathbf{u}\sim N(0,\, \sigma \mathbf{I})} \left[(g(\mathbf{x} + \mathbf{u};\, \theta') - z)^2 \right]$$

Neighborhood: $\theta' \in \mathbf{B}(\theta)$ means $\| \theta - \theta' \| \leq \varepsilon \| \theta \|$

ROMA (Yu et al., 2021)

Smoothness-Aware Surrogate Ensemble



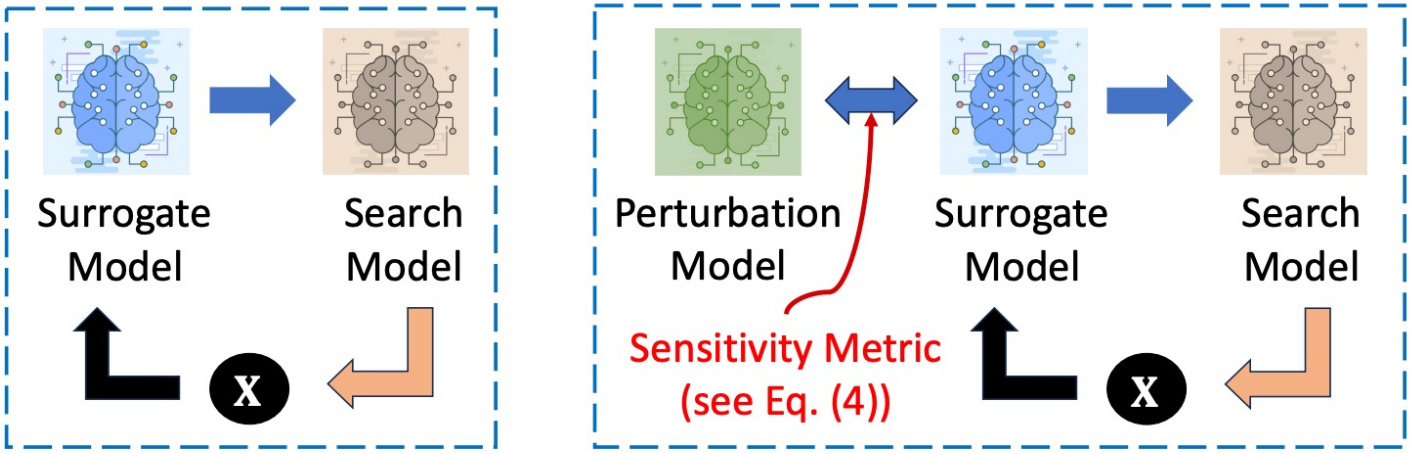
Test-Time Adaption: Choose the surrogate with maximum local smoothness to guide the next search step

$\theta_{t+1} = \min \| \nabla g(\mathbf{x}; \theta) \| + \alpha(g(\mathbf{x}; \theta) - g(\mathbf{x}; \theta_t))$ => smoothest model

$\mathbf{x} = \mathbf{x} + \varepsilon \nabla g(\mathbf{x}; \theta_{t+1})$ => step-wise update with selected model

ROMA (Yu et al., 2021)

Minimizing Surrogate Sensitivity



(a) Existing Offline Optimizer

minimize *loss*(,)

(b) Boosting Offline Optimizers with Surrogate Sensitivity

minimize [*loss*(,) + **max** *regularizer*(,)]

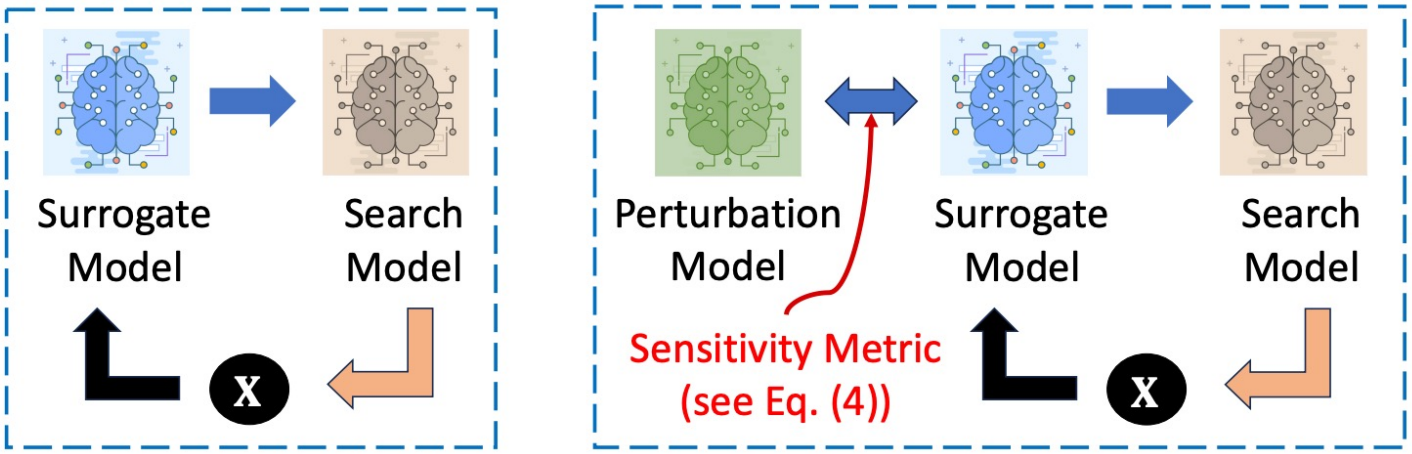
BOSS (Dao et al., 2024a), IGNITE (Dao et al., 2024b)

Surrogate Sensitivity: The chance that the expected output of the model changes significantly under random model perturbation

$$\mathbf{S}(\alpha, \omega) = \mathbf{P}_{\mathbf{u}}(A(\theta, \mathbf{u}) \geq \alpha) \quad \text{where } A(\theta, \mathbf{u}) = |E[g(\mathbf{x}; \theta + \mathbf{u})] - E[g(\mathbf{x}; \theta)]|$$

$$\text{and } \mathbf{u} \sim N(0, \omega \mathbf{I})$$

Minimizing Surrogate Sensitivity



(a) Existing Offline Optimizer

minimize *loss*(,)

(b) Boosting Offline Optimizers with Surrogate Sensitivity

minimize [*loss*(,) + **max** *regularizer*(,)]

BOSS (Dao et al., 2024a), IGNITE (Dao et al., 2024b)

Surrogate Sensitivity: The chance that the expected output of the model changes significantly under random model perturbation

$$\theta = \text{argmin } \mathrm{L}(\theta) + \lambda\, \mathbf{S}(\alpha, \omega)$$

GABO: Using In-Distribution Critics

In-Distribution Critics: The chance that the expected output of the model changes significantly under random model perturbation

$$c^*(.) = \text{argmax } E_{\mathbf{x}' \sim P}[c(\mathbf{x}')] - E_{\mathbf{x} \sim Q}[c(\mathbf{x})]$$

Constrained Optimization: Minimize $g(\mathbf{x}; \theta)$ where $c(\mathbf{x})$ often determines in-distribution with high confidence:

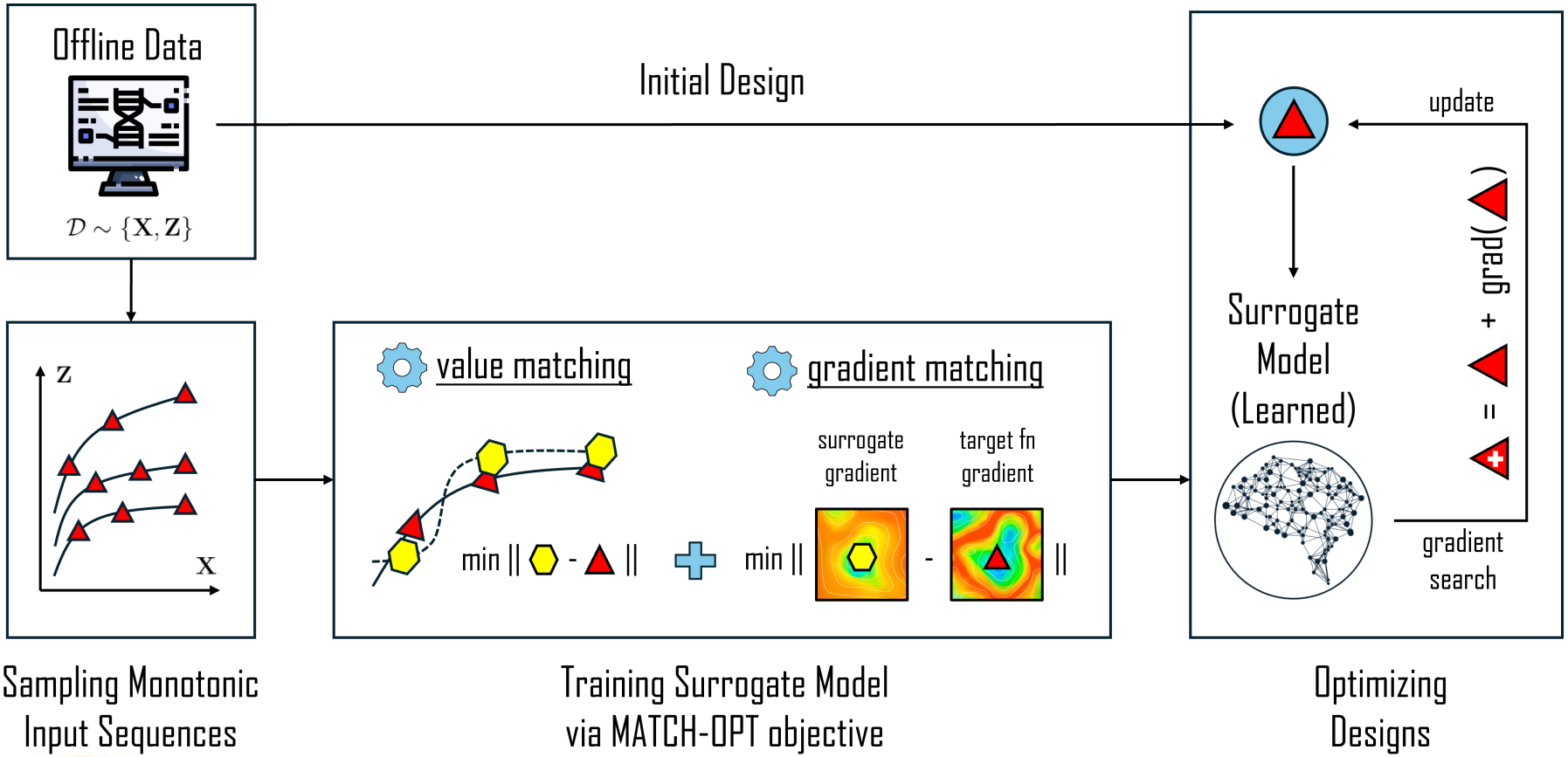
$$\text{minimize } -g(\mathbf{x}; \theta)$$

$$\text{subject to } E_{\mathbf{x}' \sim P}[c^*(\mathbf{x}')] - c(\mathbf{x}) \leq 0$$

GABO (Yao et al., 2024)

MATCH-OPT: Gradient Matching

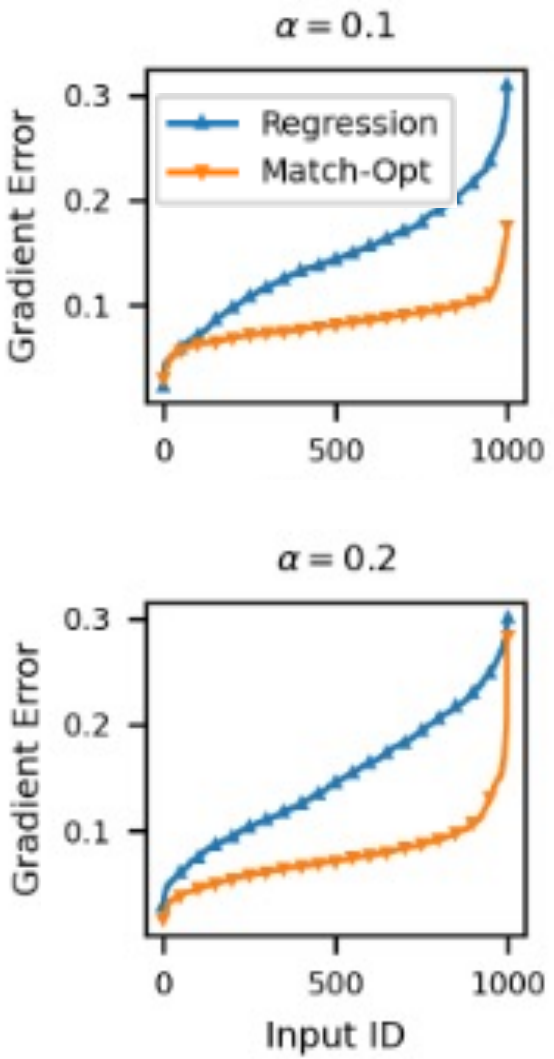
Gradient Matching: Learning gradient curvature instead of matching objective function output values (relative ordering is all that matters!)



MATCH-OPT (Hoang et al., 2024)

Gradient Matching

Intuition:
(1) gradient error leads to compounding search errors
(2) gradient error can be subsided via explicit gradient matching



Empirical verification on 4D Shekel function

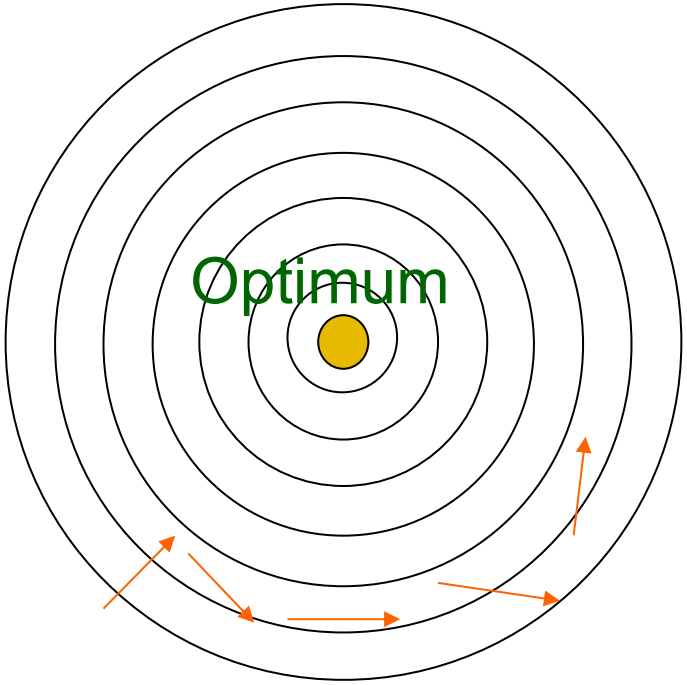
(1) train $\sim N(0, \mathbf{I})$

(2) test $\sim N(0, \alpha\mathbf{I})$ (i.e., OOD testing inputs)

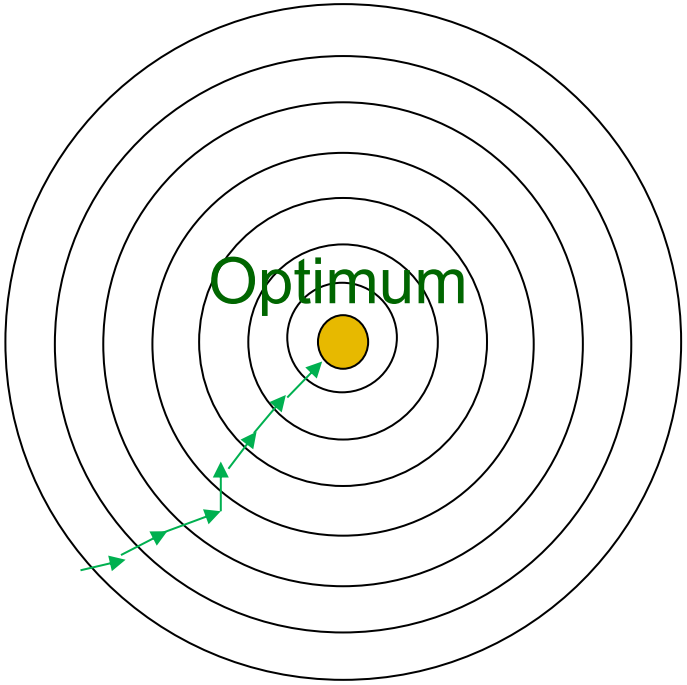
Observation: Gap widens with optimization steps
→ less error compounding in OOD regions.

Gradient Matching: Intuition

Key Idea: Learn the *dynamics* of optimization, not just function values



Gradients do not align with optimization path



Gradients align with optimization path

→ Learned gradients with MATCH-OPT

Gradient Matching Approach: The Recipe

A. Sample monotonic trajectories (low → high performance)
→ Learn optimization-aligned gradients, not just function values

B. Match integrated gradients with gradient-alignment loss
→ Generalizes better in high-value regions

C. Combined MSE + gradient loss
→ Closes the prediction-optimization gap of standard surrogates

Gradient Matching via Line Integration Theorem

Gradient Match Loss

Use line integration theorem,

$$\Delta z = z' - z \quad = \quad (\boldsymbol{x}' - \boldsymbol{x})^{\mathrm{T}} \int_{\mathbf{0}}^{\mathbf{1}} \nabla_{\boldsymbol{x}}\, g(t\boldsymbol{x}' + (1-t)\boldsymbol{x})\, dt$$

$$\approx \quad (\boldsymbol{x}' - \boldsymbol{x})^{\mathrm{T}} \int_{\mathbf{0}}^{\mathbf{1}} \nabla_{\boldsymbol{x}}\, g(t\boldsymbol{x}' + (1-t)\boldsymbol{x};\ \phi)\, dt$$

Gradient Match Loss:

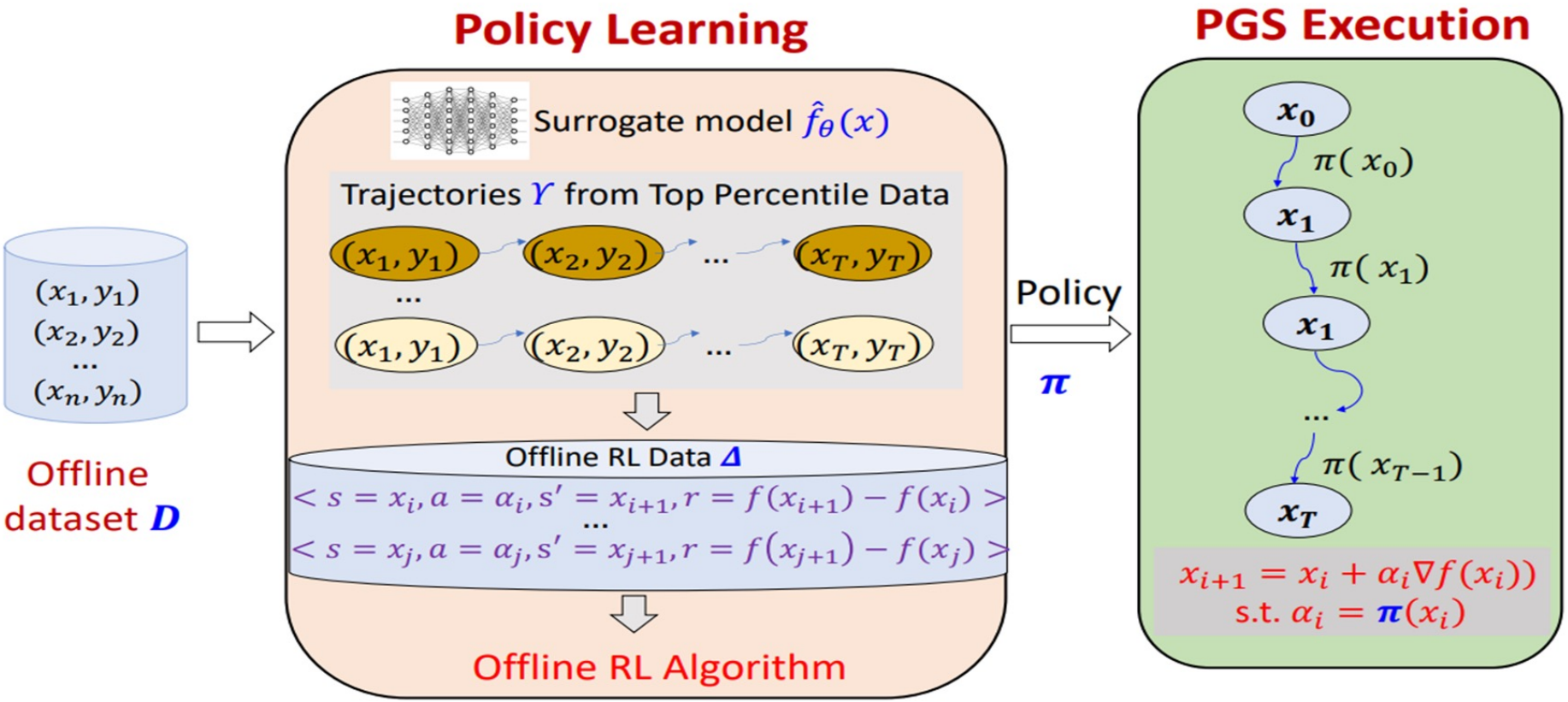
(1) sampling a no. of offline training pairs ($\boldsymbol{x}$, z) and ($\boldsymbol{x}'$, z')

(2) minimize the (averaged) gap between **(A)** and **(B)**

MATCH-OPT (Hoang et al., 2024)

PGS: Policy-Guided Search via Offline RL

Key Idea: Learn a policy using offline RL to correct gradient errors



PGS (Chemingui et al., 2024)

Experimental Evaluation: Forward Modeling

Benchmark Optimization Tasks:

Dataset	Size	Dimensions	Categories	Type
Ant Morphology	25009	60	N/A	Continuous
D'Kitty Morphology	25009	56	N/A	Continuous
TF Bind 8	32898	8	4	Discrete
TF Bind 10	50000	10	4	Discrete
RNA-Binding	5000	14	4	Discrete

Trabucco et al, *"Design-Bench: Benchmarks for Data-Driven Offline Model-Based Optimization"*, ICML 2022
Lorenz et al, *"ViennaRNA Package 2.0", Algorithms for Molecular Biology 2011*

Experimental Evaluation: Forward Modeling

Normalized performance of at 100-percentile

METHOD	ANT	DKITTY	HOPPER	SCON	TF8	TF10	MNR
GA	0.271	0.895	0.780	0.699	0.954	0.966	0.600
ENS-MEAN	0.517	0.899	1.524	0.716	0.926	**0.968**	0.500
ENS-MIN	0.536	0.908	1.420	0.734	0.959	0.959	0.467
CMA-ES	**0.974**	0.722	0.620	**0.757**	**0.978**	0.966	0.367
MINS	0.910	0.939	0.150	0.690	0.900	0.759	0.700
CBAS	0.842	0.879	0.150	0.659	0.916	0.928	0.733
ROMA	0.832	0.880	2.026	0.704	0.664	0.820	0.667
BONET	0.927	0.954	0.395	0.500	0.911	0.756	0.683
COMS	0.885	0.953	**2.270**	0.565	0.968	0.873	0.467
MATCH-OPT	0.931 (2)	**0.957 (1)**	1.572 (3)	0.732 (3)	0.977 (2)	0.924 (6)	**0.283 (1)**

Observations:

(1) no method is best in more than 2 task domains

(2) MATCH-OPT is in top-3 in 4/6 tasks; more reliable

(3) MATCH-OPT achieves best mean normalized rank

MATCH-OPT (Hoang et al., 2024)

Outline of Tutorial

- Introduction and Overview
- Forward Modeling Approaches
- Inverse Modeling Approaches
- Theoretical Analysis
- Offline Optimization in Small Data Setting
- Open Challenges

Inverse Modeling: Big Picture

- Focus on modeling p(design | performance)
  $\mathbf{x}$ $\quad z \geq \lambda$

- How to **redistribute mass** from the data distribution **toward** regions associated with **high performance**?

p($\mathbf{x}$) $\xrightarrow{\ ?\ }$ p($\mathbf{x}$ | $z \geq \lambda$)

Forward vs. Inverse Modeling

- How to **redistribute mass** from the data distribution **toward** regions associated with **high performance**?

$$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{\quad ? \quad} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$$

- (Forward) Geometric vs. (Inverse) Distributional view

local gradient $\nabla g(\mathbf{x})$
edit policy: $x \rightarrow \Delta\mathbf{x}$

neural surrogate $z = g(\mathbf{x})$
reinforcement learning

data distribution $p(\mathbf{x})$
predictive uncertainty $p(z \mid \mathbf{x})$

generative modeling
probabilistic prediction

Inverse Modeling: A Unified View

$$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{\ ?\ } p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$$

- Unified view (w/ different instantiations):

$$p(\mathbf{x} \mid z \geq \lambda) = p(z \geq \lambda \mid \mathbf{x})\, p(\mathbf{x}) / p(z \geq \lambda)$$

$$\propto p(z \geq \lambda \mid \mathbf{x})\, p(\mathbf{x}) \quad \text{[Bayes Theorem]}$$

where $p(\mathbf{x})$ can be learned via deep generative modeling

$$p(z \geq \lambda \mid \mathbf{x}) = \mathrm{E}\,[\mathrm{I}(z \geq \lambda) \mid z \sim p_D(z \mid \mathbf{x})]$$

probabilistic prediction
(learned from offline data)

Inverse Modeling: A Unified View

$$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{\;?\;} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$$

- Unified view (w/ different instantiations):

$$p(\mathbf{x} \mid z \geq \lambda) = p(z \geq \lambda \mid \mathbf{x})\, p(\mathbf{x}) / p(z \geq \lambda)$$

$$\propto p(z \geq \lambda \mid \mathbf{x})\, p(\mathbf{x}) \quad \text{[Bayes Theorem]}$$

$p(\mathbf{x})$ and $p(z \geq \lambda \mid \mathbf{x})$ can be estimated but exact computation of $p(\mathbf{x} \mid z \geq \lambda)$ remains intractable

Inverse Modeling: A Unified View

$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{?} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$

- Unified view (w/ different instantiations):

$$p(\mathbf{x} \mid z \geq \lambda) = p(z \geq \lambda \mid \mathbf{x})\, p(\mathbf{x}) / p(z \geq \lambda)$$

$$\propto p(z \geq \lambda \mid \mathbf{x})\, p(\mathbf{x}) \quad \text{[Bayes Theorem]}$$

$p(\mathbf{x})$ and $p(z \geq \lambda \mid \mathbf{x})$ can be estimated but exact computation of $p(\mathbf{x} \mid z \geq \lambda)$ remains intractable

Inverse modeling: Different approaches to approximate $p(\mathbf{x} \mid z \geq \lambda)$ with a parametric distribution

DBAS: Design by Adaptive Sampling

$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{?} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \boldsymbol{\lambda}}{\text{performance}})$

- Design by Adaptive Sampling (DBAS) [Brookes & Listgarten, 2018]

$$\theta = \text{argmin KL}(p(\mathbf{x} \mid z \geq \boldsymbol{\lambda}) \,\|\, p(\mathbf{x}; \theta))$$

$$= \text{argmin } E[-\log p(\mathbf{x}; \theta) * p(z \geq \boldsymbol{\lambda} \mid \mathbf{x}) \mid \mathbf{x} \sim p(\mathbf{x})]$$

$$\simeq \text{argmin } E[-\log p(\mathbf{x}; \theta) * p(z \geq \boldsymbol{\lambda} \mid \mathbf{x}) \mid \mathbf{x} \sim D(\mathbf{x})]$$

empirical data distribution

DBAS: Design by Adaptive Sampling

$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{?} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$

- Design by Adaptive Sampling (DBAS) [Brookes & Listgarten, 2018]

$$\theta = \text{argmin KL}(p(\mathbf{x} \mid z \geq \lambda) \,\|\, p(\mathbf{x}; \theta))$$

$$= \text{argmin } E[-\log p(\mathbf{x}; \theta) * p(z \geq \lambda \mid \mathbf{x}) \mid \mathbf{x} \sim p(\mathbf{x})]$$

$$\simeq \text{argmin } E[-\log p(\mathbf{x}; \theta) * \boxed{p(z \geq \lambda \mid \mathbf{x})} \mid \mathbf{x} \sim D(\mathbf{x})]$$

too close to zero when λ is large
and $\mathbf{x} \sim p(\mathbf{x}) \cong D(\mathbf{x})$ – learning collapses!

DBAS: Design by Adaptive Sampling

$p(\mathbf{x}_{\text{design}}) \xrightarrow{?} p(\mathbf{x}_{\text{design}} \mid z \geq \lambda_{\text{performance}})$

- Design by Adaptive Sampling (DBAS) [Brookes & Listgarten, 2018]

Annealing (raising λ slowly) to avoid learning collapse

$$\theta_{t+1} = \underset{\theta}{\text{argmin}}\, E[-\log \boxed{p(\mathbf{x}; \theta)} * p(z \geq \lambda_{t+1} \mid \mathbf{x}) \mid \mathbf{x} \sim \boxed{p(\mathbf{x}; \theta_t)}]$$

(learnable) present generative model

(frozen) generative model from previous step

DBAS: Design by Adaptive Sampling

$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{?} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$

- Design by Adaptive Sampling (DBAS) [Brookes & Listgarten, 2018]

Generative model representation:
must be differentiable and can be sampled from

For example, variational auto-encoder (VAE):

$$\log p(\mathbf{x}; \theta) \geq E_{q(z \mid \mathbf{x}; \gamma)}[\log \underbrace{p(\mathbf{x} \mid z; \theta)}_{\text{denoising}}] - \mathrm{KL}(\underbrace{q(z \mid \mathbf{x}; \gamma)}_{\text{noising}} \,\|\, \underbrace{p(z)}_{\text{noise}})$$

DBAS: Design by Adaptive Sampling

$p(\mathbf{x})$ $\xrightarrow{?}$ $p(\mathbf{x} \mid z \geq \lambda)$

sample can be drawn via (1) sampling (Gaussian) noise;
(2) simulating denoising

For example, variational auto-encoder (VAE):

$$\log p(\mathbf{x}; \theta) \geq E_{q(z \mid \mathbf{x}; \gamma)}[\log p(\mathbf{x} \mid z; \theta)] - KL(q(z \mid \mathbf{x}; \gamma) \,\|\, p(z))$$

denoising noising noise

CBAS: Conditional by Adaptive Sampling

p(design) $\mathbf{x}$ —?→ p(design | performance) $\mathbf{x}$ $z \geq \lambda$

- Conditional by Adaptive Sampling for Robust Design (CBAS) [Brookes and Listgarten, 2019]

Issue with annealing DBAS:

θ_{t+1} = argmin E[-log p($\mathbf{x}$; θ) * p($z \geq \lambda_{t+1} \mid \mathbf{x}$) | $\mathbf{x} \sim$ p($\mathbf{x}$; θ_t)]

= argmin KL(p($\mathbf{x} \mid z \geq \lambda_{t+1}$) ‖ p($\mathbf{x}$;$\theta_t$))

(eroding the anchor to D(x) – error might be amplified)

CBAS: Conditional by Adaptive Sampling

$p(\underset{\mathbf{x}}{\text{design}})$ $\xrightarrow{?}$ $p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$

- Conditional by Adaptive Sampling for Robust Design (CBAS) [Brookes and Listgarten, 2019]

Anchoring to D(x) (data distribution) via simple re-parameterization:

$$\theta_{t+1} = \text{argmin } E[-\log p(\mathbf{x}; \theta) * p(z \geq \lambda_{t+1} \mid \mathbf{x}) \mid \mathbf{x} \sim p(\mathbf{x}; \theta_t)]$$

$$= \text{argmin } E_D[-\log p(\mathbf{x}; \theta) * (p(z \geq \lambda_{t+1} \mid \mathbf{x}) / D(x))]$$

Limitations of DBAS/CBAS

$$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{\ ?\ } p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$$

- Fundamental limitations of DBAS/CBAS:

$$p(\mathbf{x} \mid z \geq \lambda) \propto p(z \geq \lambda \mid \mathbf{x})\, p(\mathbf{x})$$

what happens when a high-value regime D_h
is not well-supported, i.e., $p(\mathbf{x}) \simeq 0$ when $\mathbf{x}$ belongs to D_h?

DBAS/CBAS only reshape *within* existing support
(i.e., cannot expand support)

BONET: Generative Pretraining

p(design) **x** —?→ p(design | condition) **x** **c**

- Generative Pre-Training for Offline Optimization (BONET)
  [Mashkaria et al., 2023]

Alternatively, BONET characterizes condition as a generative path, i.e., condition = (r(1), **x**(1), …, r(p), **x**(p), r(p + 1))

where r(i) = (f(**x**$_*$) – f(**x**(i)) + … + (f(**x**$_*$) – f(**x**(τ)) -- regret

$$p(\mathbf{x}(p+1) \mid r(p+1); \theta) = E[p(\mathbf{x}(p+1), \mathbf{c} \mid r(p+1); \theta) \mid \mathbf{c}]$$

$$= E_{\mathbf{c}}\left[\prod_{t=1}^{p} \boxed{p(\mathbf{x}(p+1) \mid \mathbf{r}(1{:}\,p+1), \mathbf{x}(1{:}\,p); \theta)}\right]$$

auto-regressive model (i.e., transformer)

BONET: Generative Pretraining

p(design) **x** —?→ p(design | condition) **x** **c**

- Generative Pre-Training for Offline Optimization (BONET)
  [Mashkaria et al., 2023]

Alternatively, BONET characterizes condition as a generative path, i.e., condition = (r(1), **x**(1), …, r(p), **x**(p), r(p + 1))

where r(i) = (f(**x**$_*$) – f(**x**(i)) + … + (f(**x**$_*$) – f(**x**(τ)) -- regret

optimization steps

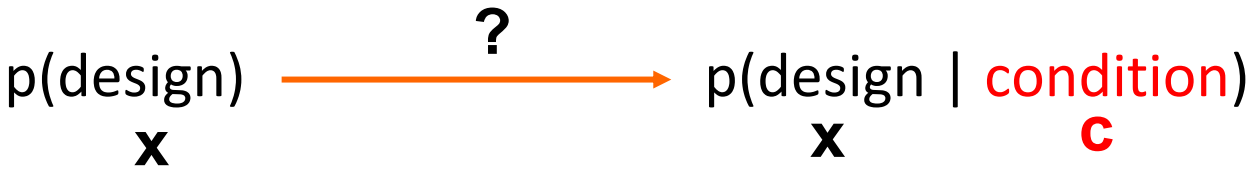
Learn: maximizing log p(**x**(p + 1) | r(p + 1); θ) via

IV

maximizing $E_{\mathbf{c}}\left[\log \prod_{t=1}^{p} \mathbf{p(x(p+1) \mid r(1:p+1), x(1:p); \theta)}\right]$

The condition **c** can be sampled from offline data

BONET: Generative Pretraining



- Generative Pre-Training for Offline Optimization (BONET)
  [Mashkaria et al., 2023]

Alternatively, BONET characterizes condition as a generative path, i.e., condition = (r(1), **x**(1), …, r(p), **x**(p), r(p + 1))

where r(i) = (f(**x**$_*$) – f(**x**(i)) + … + (f(**x**$_*$) – f(**x**(τ)) -- regret

optimization steps

Generate: 1. sample **c**
2. set r(p + 1) = r (small value)
3. generate **x**(p + 1) via **p(x(p + 1) | r(1: p + 1), x(1: p) ; θ)**
4. repeat step 2 with p++ and same r

Limitation of BONET

p(design) $\xrightarrow{?}$ p(design | condition)

$\mathbf{x}$ $\mathbf{x}$ $\mathbf{c}$

- Fundamental limitation of BONET:

p(design | condition) only generalizes well along paths that are similar to observed condition in offline data

i.e., learning is ill-conditioned in regimes of condition w/o support from the data distribution

Limitations of DBAS/CBAS

$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{?} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$

- Fundamental limitations of DBAS/CBAS (recap):

$$p(\mathbf{x} \mid z \geq \lambda) \propto p(z \geq \lambda \mid \mathbf{x})\, p(\mathbf{x})$$

DBAS/CBAS only reshape *within* existing support
i.e., cannot expand support

Motivation for using diffusion:
A diffusion model guarantees support almost everywhere

Limitations of DBAS/CBAS

$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{?} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$

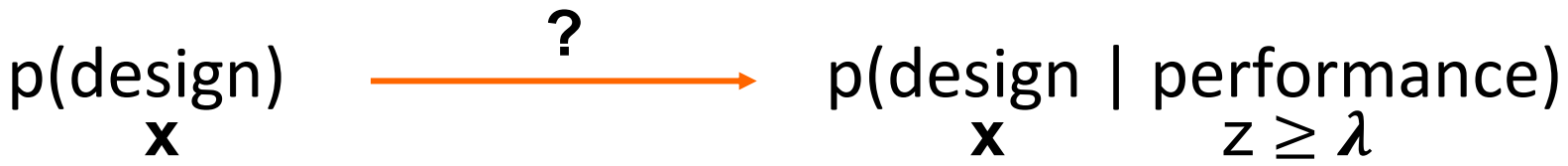
- Fundamental limitations of DBAS/CBAS:

$$p(\mathbf{x} \mid z \geq \lambda) \propto p(z \geq \lambda \mid \mathbf{x})\, p(\mathbf{x})$$

Even with diffusion base $p(\mathbf{x})$, DBAS/CBAS still *reweights on rare event* as *$p(z \geq \lambda \mid \mathbf{x})$ is applied post generation*

i.e., less mass gets shifted to high-value regimes that have originally low support
(even when $p(z \geq \lambda \mid \mathbf{x})$ suggests high potential)

DDOM: Denoising Diffusion Optimization Model



- Guided Diffusion: Denoising Diffusion Optimization Model (DDOM) [Krishnamoorthy et al., 2023]

1. Can expand support flexibly
(mass can be moved to previously unsupported regions)

2. Incorporate reweighting within generation process
(guided generation to accelerate mass relocation)

DBAS/CBAS vs. DDOM

$p(\mathbf{x})$ $\xrightarrow{?}$ $p(\mathbf{x} \mid z \geq \lambda)$

- DBAS/CBAS represents (noising, denoising) via separately parameterized encoder & decoder (do not guarantee $p(\mathbf{x})$ having support almost everywhere)

- DDOM represents (noising, denoising) via SDE and its induced reverse-time SDE

⇩

the relationship is provable rather than approximated via learning (compared to VAE)

DDOM: Denoising Diffusion Optimization Model

$$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{\;?\;} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$$

- Guided Diffusion: Denoising Diffusion Optimization Model (DDOM) [Krishnamoorthy et al., 2023]

Noising via forward SDE from $\mathbf{x}(0) \sim D(x)$

$d\mathbf{x}(t) = u(\mathbf{x}(t), t)\, dt + g(t)\, d\mathbf{w}(t)$ where $d\mathbf{w}(t) \sim N(0, dt)$

Brownian motion

DDOM: Denoising Diffusion Optimization Model

$$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{\ ?\ } p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$$

- Guided Diffusion: Denoising Diffusion Optimization Model (DDOM) [Krishnamoorthy et al., 2023]

  Noising via forward SDE from $\mathbf{x}(0) \sim D(x)$

  $d\mathbf{x}(t) = u(\mathbf{x}(t), t)\, dt + g(t)\, d\mathbf{w}(t)$ where $d\mathbf{w}(t) \sim N(0, dt)$

  Brownian motion

  -1/2 g(t) positive, monotonic

  Appropriate choices of $u(\mathbf{x}(t), t)$ and $g(t)$ guarantee $\mathbf{x}(t)$ is distributed by $N(0, \mathbf{I})$ when t is sufficiently large

DDOM: Denoising Diffusion Optimization Model

$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{?} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$

- Guided Diffusion: Denoising Diffusion Optimization Model (DDOM) [Krishnamoorthy et al., 2023]

  Noising via forward SDE from $\mathbf{x}(0) \sim D(\mathbf{x})$

  $d\mathbf{x}(t) = u(\mathbf{x}(t), t)\, dt + g(t)\, d\mathbf{w}(t)$ where $d\mathbf{w}(t) \sim N(0, dt)$

  Denoising via reverse-time SDE from $\mathbf{x}(T) \sim D(\mathbf{x})$

  $d\mathbf{x}(t) = (u(\mathbf{x}(t), t) - g(t)^2 \nabla \log p_t(\mathbf{x}(t)))\, dt + g(t)\, d\underline{\mathbf{w}}(t)$

  where $d\underline{\mathbf{w}}(t)$ is reverse Brownian

  $p_t(\mathbf{x}(t))$ is marginal of $\mathbf{x}(t)$ in forward SDE

DDOM: Denoising Diffusion Optimization Model

$$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{\;?\;} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$$

- Idea #1: Classifier-guided diffusion (not yet what DDOM does)
  [Dhariwal and Nichol, 2021]

1. Given $u(\mathbf{x}(t), t)$ and $g(t)$, learn $\nabla \log p_t(\mathbf{x}(t))$ via score matching

2. Augmenting with $p(z \geq \lambda \mid \mathbf{x}(t))$

$$\nabla \log p_t(\mathbf{x} \mid z \geq \lambda) = \nabla \log p_t(\mathbf{x}(t)) + \nabla \log p(z \geq \lambda \mid \mathbf{x}(t))$$

DDOM: Denoising Diffusion Optimization Model

$p(\mathbf{x}) \xrightarrow{?} p(\mathbf{x} \mid z \geq \lambda)$

- Idea #1: Classifier-guided diffusion

3. sample high-value designs via simulating:

$$d\mathbf{x}(t) = (u(\mathbf{x}(t), t) - g(t)^2 \nabla \log p_t(\mathbf{x}(t) \mid z \geq \lambda))\, dt + g(t)\, d\underline{\mathbf{w}}(t)$$

Boils down to estimate score $\nabla \log p_t(\mathbf{x}(t))$
and augment it with $p(z \geq \lambda \mid \mathbf{x}(t))$ – the rest is fixed

DDOM: Denoising Diffusion Optimization Model

$p(\mathbf{x})$ $\xrightarrow{?}$ $p(\mathbf{x} \mid z \geq \lambda)$

- Issue with idea #1:

Dependent on learning $p(z \mid \mathbf{x})$ accurately

1. $p(z \mid \mathbf{x})$ might not generalize well along the diffusion path (due to limited data)

2. $p(z \mid \mathbf{x})$ requires accurate *uncertainty quantification* (less reliable with large model)

DDOM: Denoising Diffusion Optimization Model

$$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{\ ?\ } p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \boldsymbol{\lambda}}{\text{performance}})$$

- Guided Diffusion: Denoising Diffusion Optimization Model (DDOM) [Krishnamoorthy et al., 2023]

Idea #2: Classifier-free diffusion (what DDOM does)
[Ho and Salimans, 2022]

1. Given $u(\mathbf{x}(t), t)$ and $g(t)$, learn $\nabla \log p_t(\mathbf{x}(t))$ via score matching

2. Aim to sample from $p(\mathbf{x}(t) \mid z(0) = \boldsymbol{\lambda})$
rather than $p(\mathbf{x}(t) \mid z(t) \geq \boldsymbol{\lambda})$

DDOM: Denoising Diffusion Optimization Model

$$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{\ ?\ } p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$$

- Idea #2: Classifier-free diffusion (what DDOM does)
  [Ho and Salimans, 2022]

2. Note that

conditional score matching

$$\nabla \log p(\mathbf{x}(t) \mid z(0)) = \mathbb{E}[\nabla \log p(\mathbf{x}(t) \mid \mathbf{x}(0)) \mid \mathbf{x}(t), \mathbf{x}(0), z(0)]$$

where $(\mathbf{x}(0), z(0)) \sim D$

$\mathbf{x}(t)$ is sampled via simulating SDE from $\mathbf{x}(0)$

DDOM: Denoising Diffusion Optimization Model

$$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{\ ?\ } p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$$

- Idea #2: Classifier-free diffusion (what DDOM does)

  [Ho and Salimans, 2022]

2. Note that

   does not involve $p(z \mid \mathbf{x})$ ☺

$$\nabla \log p(\mathbf{x}(t) \mid z(0)) = \mathbb{E}[\nabla \log p(\mathbf{x}(t) \mid \mathbf{x}(0)) \mid \mathbf{x}(t), \mathbf{x}(0), z(0)]$$

where $(\mathbf{x}(0), z(0)) \sim D$

$\mathbf{x}(t)$ is sampled via simulating SDE from $\mathbf{x}(0)$

DDOM: Denoising Diffusion Optimization Model

p(design) $\mathbf{x}$ $\xrightarrow{?}$ p(design | performance) $\mathbf{x}$ $z \geq \lambda$

- Idea #2: Classifier-free diffusion (what DDOM does)
  [Ho and Salimans, 2022]

3. sample high-value designs for a performance level $z(0) = \lambda$ via simulating:

$$d\mathbf{x}(t) = (u(\mathbf{x}(t), t) - g(t)^2\, s(\boldsymbol{x}(t), t, \boldsymbol{\lambda}))\, dt + g(t)\, d\underline{\mathbf{w}}(t)$$

where $s(\boldsymbol{x}(t), t, \boldsymbol{\lambda}) = \alpha\, \nabla \log p(\mathbf{x}(t) \mid \mathbf{x}(0)) + (1 - \alpha)\, \nabla \log p(\mathbf{x}(t) \mid z(0) = \boldsymbol{\lambda})$

DDOM: Denoising Diffusion Optimization Model

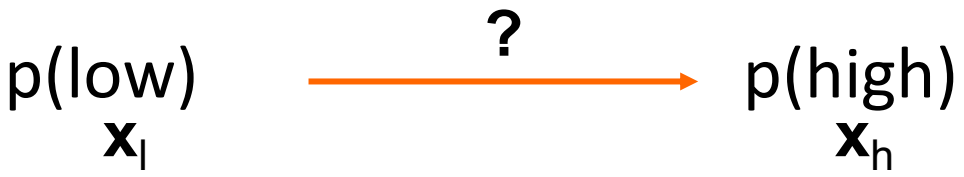
$p(\underset{\mathbf{x}}{\text{design}}) \xrightarrow{?} p(\underset{\mathbf{x}}{\text{design}} \mid \underset{z \geq \lambda}{\text{performance}})$

- Issue with idea #2 (DDOM)

Due to limited offline data, the score network $s(\boldsymbol{x}(t), t, \boldsymbol{\lambda})$ might only see low value of $\boldsymbol{\lambda}$ = z(0) and cannot generalize well during sampling when we want to condition on high $\boldsymbol{\lambda}$

We need more explicit mechanism to mitigate data scarcity

Low-to-High Value Design Translation



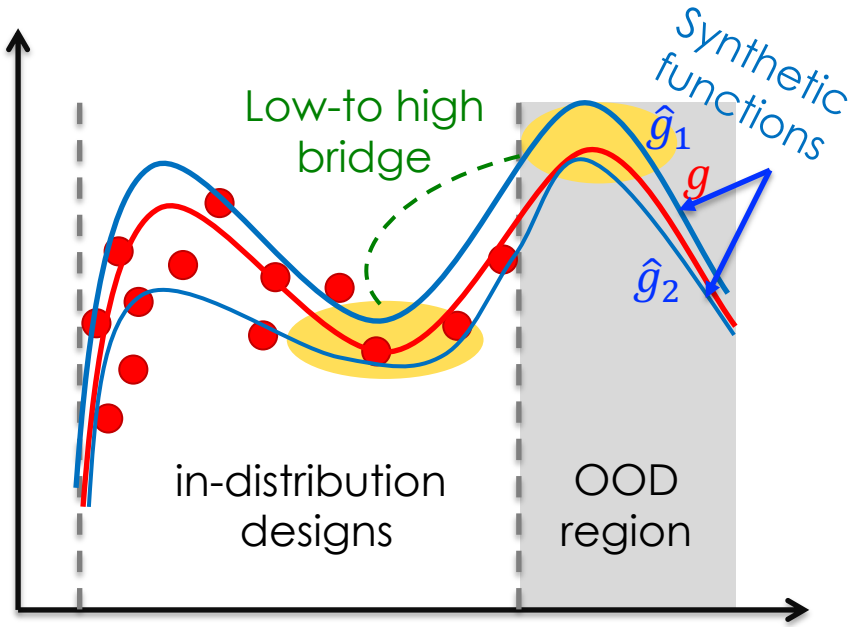
- New Perspective: Generative Meta Optimization

1. Generate similar synthetic functions and their corresponding low- and high-value regimes (to mitigate data scarcity)

2. Learn direct, meta-transport between the unionized low- and high-value design regimes of all functions (sidestep guided diffusion, de-amplify generative errors)

Distributional Translation via Probabilistic Bridge

p(low) $\mathbf{x}_l$? p(high) $\mathbf{x}_h$



Offline Optimization:
Learning Low-to-High Meta Transport

Learning **low- to high-value mapping** across **related functions near the oracle**:

$$\mathbf{N}(g) = \{\bar{g}(\boldsymbol{x}) \mid \bar{g}(\boldsymbol{x}) \cong g(\boldsymbol{x})\ \forall \boldsymbol{x} \in D\}$$

Low Region: $D_- = \{\boldsymbol{x} = \text{argmax}\ \bar{g}(\boldsymbol{x}) \sim \mathbf{N}(g)\}$

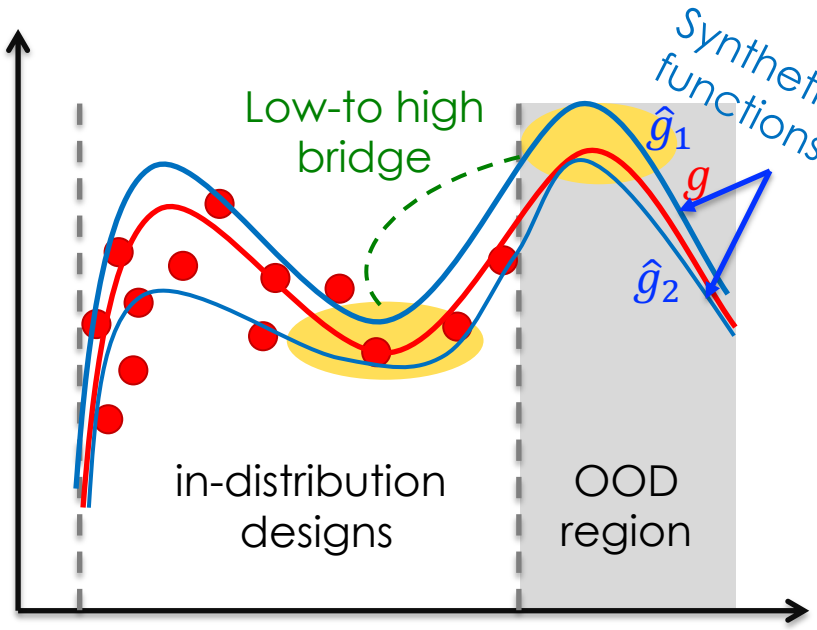
High Region: $D_+ = \{\boldsymbol{x} = \text{argmax}\ \bar{g}(\boldsymbol{x}) \sim \mathbf{N}(g)\}$

How to learn θ: $D_- \rightarrow D_+$

(Cuong Dao, Hung Tran, Le Nguyen, Nguyen Truong, Nghia Hoang, 2025)

Distributional Translation via Probabilistic Bridge

p(low) $\mathbf{x}_l$ —?→ p(high) $\mathbf{x}_h$



Offline Optimization:
Learning Low-to-High Meta Transport

Why sampling multiple functions?

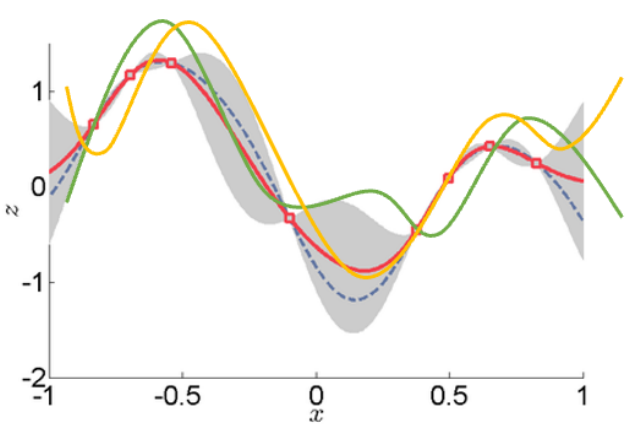
$$\mathbf{N}(g) = \{\bar{g}(\boldsymbol{x}) \mid \bar{g}(\boldsymbol{x}) \cong g(\boldsymbol{x})\ \forall \boldsymbol{x} \in D\}$$

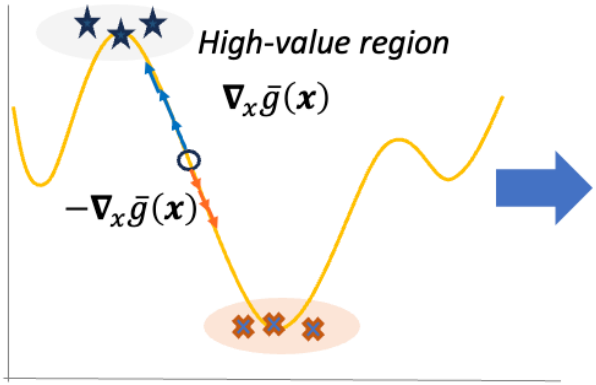
Population's wisdom: Independent models rarely agree on false optima

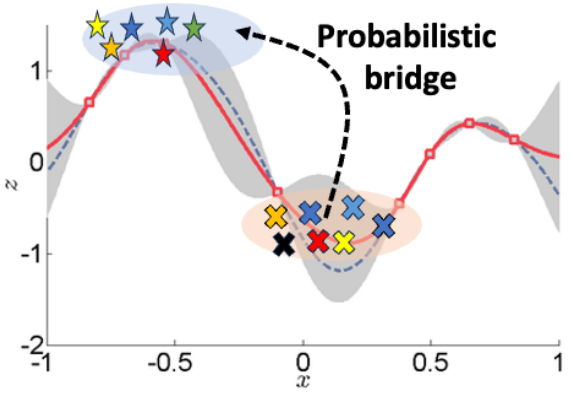
(Cuong Dao, Hung Tran, Le Nguyen, Nguyen Truong, Nghia Hoang, 2025)

Distributional Translation via Probabilistic Bridge

p(low) $\mathbf{x}_l$ —?→ p(high) $\mathbf{x}_h$







Step 1: Fit multiple GPs using offline data

Step 2: Take GA, GD on each GP posterior mean

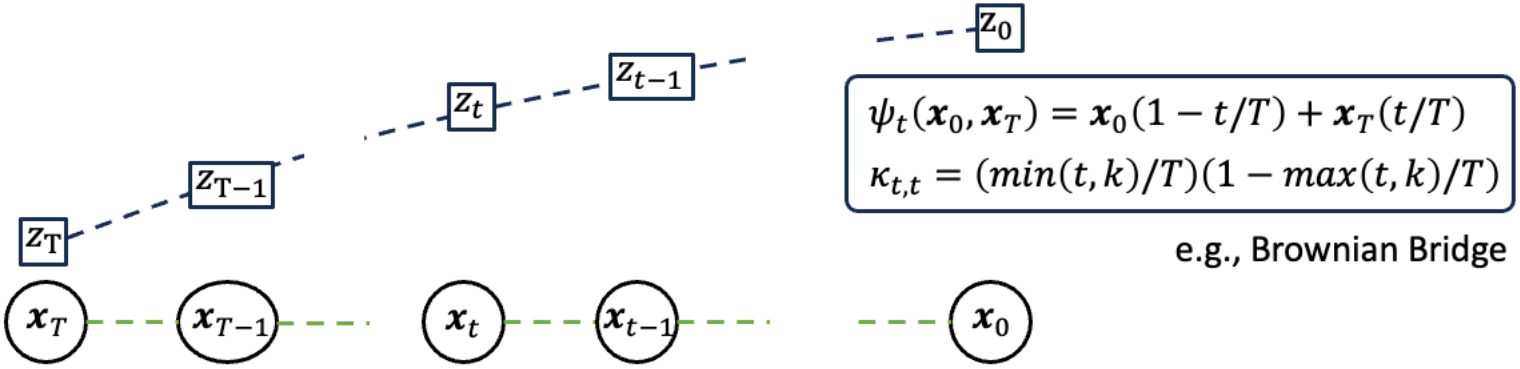
Step 3: Learning low-to-high probabilistic transport (bridge)

(Cuong Dao, Hung Tran, Le Nguyen, Nguyen Truong, Nghia Hoang, 2025)

Distributional Translation via Probabilistic Bridge

p(low) $\mathbf{x}_l$ —?→ p(high) $\mathbf{x}_h$

Bridge: Gaussian process (GP) conditioned on two endpoints



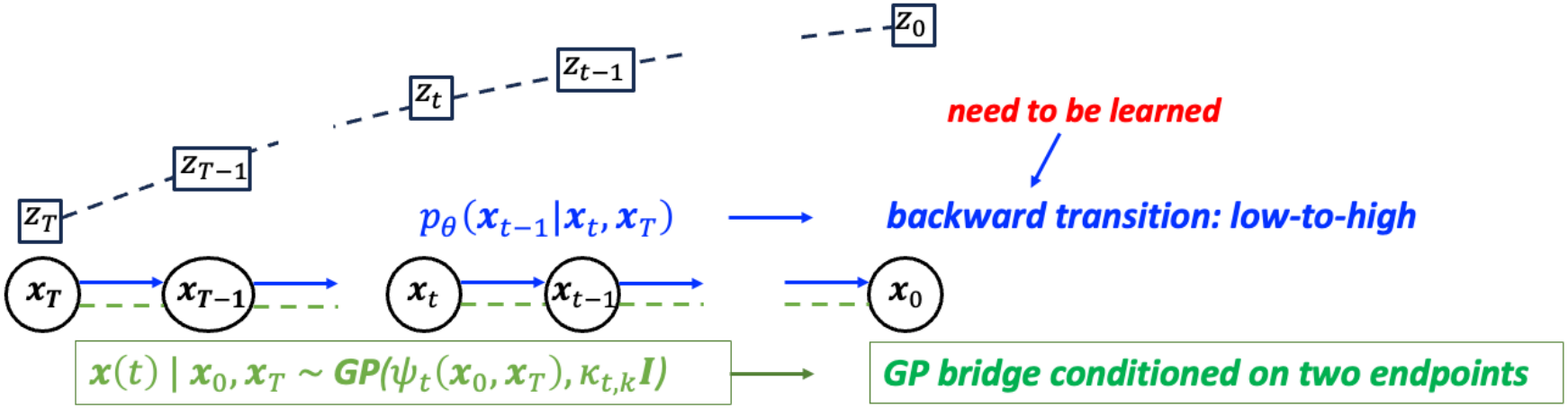
$\boldsymbol{x}(t) \mid \boldsymbol{x}_0, \boldsymbol{x}_T \sim GP(\psi_t(\boldsymbol{x}_0, \boldsymbol{x}_T), \kappa_{t,k}\boldsymbol{I})$ – *a bridge conditioned on two endpoints*

(Cuong Dao, Hung Tran, Le Nguyen, Nguyen Truong, Nghia Hoang, 2025)

Distributional Translation via Probabilistic Bridge

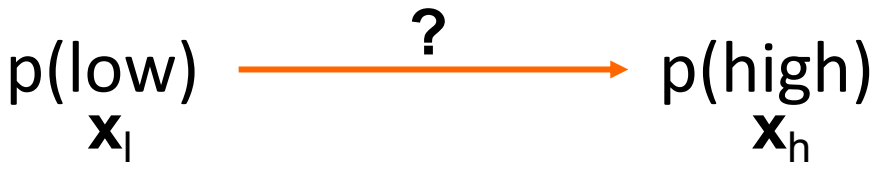
p(low) $\mathbf{x}_l$ —?→ p(high) $\mathbf{x}_h$

Bridge: Gaussian process (GP) conditioned on two endpoints



(Cuong Dao, Hung Tran, Le Nguyen, Nguyen Truong, Nghia Hoang, 2025)

Distributional Translation via Probabilistic Bridge



Bridge: Gaussian process (GP) conditioned on two endpoints

Parameterize target-agnostic transition kernel $p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_T)$
(not knowing $\boldsymbol{x}_0$)

closed-form derived from GP bridge

Learning θ: distilling $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0, \boldsymbol{x}_T)$ into $p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_T)$
(amortizing over $\boldsymbol{x}_0$)

$$\theta_{PB} = \arg\min_\theta \mathbb{E}_{(\boldsymbol{x}_0, \boldsymbol{x}_T, t)}\left[D_{KL}\left(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0, \boldsymbol{x}_T) \,||\, p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_T)\right)\right]$$

Experimental Evaluation: Inverse Modeling

Benchmark Optimization Tasks:

Dataset	Size	Dimensions	Categories	Type
Ant Morphology	25009	60	N/A	Continuous
D'Kitty Morphology	25009	56	N/A	Continuous
TF Bind 8	32898	8	4	Discrete
TF Bind 10	50000	10	4	Discrete
RNA-Binding	5000	14	4	Discrete

Trabucco et al, *"Design-Bench: Benchmarks for Data-Driven Offline Model-Based Optimization"*, ICML 2022
Lorenz et al, *"ViennaRNA Package 2.0", Algorithms for Molecular Biology 2011*

Experimental Evaluation: Inverse Modeling

Results on Design-bench

Method	Benchmarks				Mean Rank
	Ant	D'Kitty	TFBind8	TFBind10	
D_o (best)	0.565	0.884	0.439	0.467	-
BO-qEI	0.812 ± 0.000	0.896 ± 0.000	0.825 ± 0.091	0.627 ± 0.033	16.75 / 22
CMA-ES	1.561 ± 0.896	0.724 ± 0.001	0.939 ± 0.039	0.664 ± 0.034	8.00 / 22
REINFORCE	0.263 ± 0.026	0.573 ± 0.204	0.961 ± 0.034	0.618 ± 0.011	17.00 / 22
GA	0.293 ± 0.029	0.860 ± 0.021	0.985 ± 0.011	0.638 ± 0.032	12.75 / 22
COMs	0.882 ± 0.044	0.932 ± 0.006	0.940 ± 0.027	0.621 ± 0.033	13.25 / 22
CbAS	0.846 ± 0.033	0.895 ± 0.016	0.903 ± 0.028	0.649 ± 0.055	12.50 / 22
MINs	0.894 ± 0.022	0.939 ± 0.004	0.908 ± 0.063	0.630 ± 0.019	12.50 / 22
GA on GP	0.948 ± 0.013	0.946 ± 0.001	0.770 ± 0.087	0.654 ± 0.038	9.25 / 22
RoMA	0.593 ± 0.066	0.829 ± 0.020	0.665 ± 0.000	0.553 ± 0.000	20.00 / 22
ICT	0.911 ± 0.030	0.945 ± 0.011	0.888 ± 0.047	0.624 ± 0.033	13.50 / 22
Tri-mentoring	0.944 ± 0.033	0.950 ± 0.015	0.899 ± 0.045	0.647 ± 0.039	9.00 / 22
MATCH-OPT	0.931 ± 0.011	0.957 ± 0.014	0.977 ± 0.004	0.543 ± 0.002	9.50 / 22
PGS	0.949 ± 0.017	0.966 ± 0.013	0.981 ± 0.015	0.532 ± 0.000	7.75 / 22
LTR	0.907 ± 0.032	0.960 ± 0.014	0.973 ± 0.000	0.652 ± 0.039	6.25 / 22
DDOM	0.930 ± 0.029	0.925 ± 0.008	0.885 ± 0.061	0.634 ± 0.015	13.75 / 22
GTG	0.865 ± 0.040	0.935 ± 0.010	0.901 ± 0.039	0.639 ± 0.016	12.50 / 22
BDI	0.964 ± 0.000	0.941 ± 0.000	0.973 ± 0.000	0.636 ± 0.020	7.50 / 22
RGD	0.922 ± 0.020	0.883 ± 0.014	0.889 ± 0.068	0.644 ± 0.048	13.00 / 22
BONET	0.948 ± 0.025	0.957 ± 0.008	0.894 ± 0.086	0.606 ± 0.024	10.75 / 22
GABO	0.224 ± 0.051	0.719 ± 0.001	0.939 ± 0.038	0.639 ± 0.033	15.25 / 22
DEMO	0.948 ± 0.013	0.956 ± 0.011	0.812 ± 0.054	0.648 ± 0.042	9.25 / 22
ROOT (ours)	0.965 ± 0.014	0.972 ± 0.005	0.986 ± 0.007	0.685 ± 0.053	1.25 / 22

Results on Biological RNA Tasks

Method	Benchmarks			Mean Rank
	RNA-A	RNA-B	RNA-C	
CbAS	0.270 ± 0.098	0.249 ± 0.088	0.261 ± 0.093	6.00 / 8
BO-qEI	0.537 ± 0.106	0.517 ± 0.108	0.481 ± 0.100	3.67 / 8
GA	0.518 ± 0.120	0.499 ± 0.100	0.496 ± 0.091	4.33 / 8
COMs	0.187 ± 0.123	0.144 ± 0.121	0.209 ± 0.100	7.67 / 8
REINFORCE	0.166 ± 0.096	0.149 ± 0.081	0.225 ± 0.075	7.33 / 8
BDI	0.604 ± 0.000	0.505 ± 0.000	0.411 ± 0.000	4.00 / 8
Boot-Gen	0.913 ± 0.064	0.881 ± 0.024	0.786 ± 0.039	2.00 / 8
ROOT (ours)	0.956 ± 0.023	0.955 ± 0.013	0.922 ± 0.013	1.00 / 8

ROOT establishes a new SOTA across a diverse set of benchmark tasks

(Cuong Dao, Hung Tran, Le Nguyen, Nguyen Truong, Nghia Hoang, 2025)

Experimental Evaluation: Inverse Modeling

Results on Design-bench

Method	Benchmarks				Mean Rank
	Ant	D'Kitty	TFBind8	TFBind10	
D_o (best)	0.565	0.884	0.439	0.467	-
BO-qEI	0.812 ± 0.000	0.896 ± 0.000	0.825 ± 0.091	0.627 ± 0.033	16.75 / 22
CMA-ES	1.561 ± 0.896	0.724 ± 0.001	0.939 ± 0.039	0.664 ± 0.034	8.00 / 22
REINFORCE	0.263 ± 0.026	0.573 ± 0.204	0.961 ± 0.034	0.618 ± 0.011	17.00 / 22
GA	0.293 ± 0.029	0.860 ± 0.021	0.985 ± 0.011	0.638 ± 0.032	12.75 / 22
COMs	0.882 ± 0.044	0.932 ± 0.006	0.940 ± 0.027	0.621 ± 0.033	13.25 / 22
CbAS	0.846 ± 0.033	0.895 ± 0.016	0.903 ± 0.028	0.649 ± 0.055	12.50 / 22
MINs	0.894 ± 0.022	0.939 ± 0.004	0.908 ± 0.063	0.630 ± 0.019	12.50 / 22
GA on GP	0.948 ± 0.013	0.946 ± 0.001	0.770 ± 0.087	0.654 ± 0.038	9.25 / 22
RoMA	0.593 ± 0.066	0.829 ± 0.020	0.665 ± 0.000	0.553 ± 0.000	20.00 / 22
ICT	0.911 ± 0.030	0.945 ± 0.011	0.888 ± 0.047	0.624 ± 0.033	13.50 / 22
Tri-mentoring	0.944 ± 0.033	0.950 ± 0.015	0.899 ± 0.045	0.647 ± 0.039	9.00 / 22
MATCH-OPT	0.931 ± 0.011	0.957 ± 0.014	0.977 ± 0.004	0.543 ± 0.002	9.50 / 22
PGS	0.949 ± 0.017	0.966 ± 0.013	0.981 ± 0.015	0.532 ± 0.000	7.75 / 22
LTR	0.907 ± 0.032	0.960 ± 0.014	0.973 ± 0.000	0.652 ± 0.039	6.25 / 22
DDOM	0.930 ± 0.029	0.925 ± 0.008	0.885 ± 0.061	0.634 ± 0.015	13.75 / 22
GTG	0.865 ± 0.040	0.935 ± 0.010	0.901 ± 0.039	0.639 ± 0.016	12.50 / 22
BDI	0.964 ± 0.000	0.941 ± 0.000	0.973 ± 0.000	0.636 ± 0.020	7.50 / 22
RGD	0.922 ± 0.020	0.883 ± 0.014	0.889 ± 0.068	0.644 ± 0.048	13.00 / 22
BONET	0.948 ± 0.025	0.957 ± 0.008	0.894 ± 0.086	0.606 ± 0.024	10.75 / 22
GABO	0.224 ± 0.051	0.719 ± 0.001	0.939 ± 0.038	0.639 ± 0.033	15.25 / 22
DEMO	0.948 ± 0.013	0.956 ± 0.011	0.812 ± 0.054	0.648 ± 0.042	9.25 / 22
ROOT (ours)	0.965 ± 0.014	0.972 ± 0.005	0.986 ± 0.007	0.685 ± 0.053	1.25 / 22

Results on Biological RNA Tasks

Method	Benchmarks			Mean Rank
	RNA-A	RNA-B	RNA-C	
CbAS	0.270 ± 0.098	0.249 ± 0.088	0.261 ± 0.093	6.00 / 8
BO-qEI	0.537 ± 0.106	0.517 ± 0.108	0.481 ± 0.100	3.67 / 8
GA	0.518 ± 0.120	0.499 ± 0.100	0.496 ± 0.091	4.33 / 8
COMs	0.187 ± 0.123	0.144 ± 0.121	0.209 ± 0.100	7.67 / 8
REINFORCE	0.166 ± 0.096	0.149 ± 0.081	0.225 ± 0.075	7.33 / 8
BDI	0.604 ± 0.000	0.505 ± 0.000	0.411 ± 0.000	4.00 / 8
Boot-Gen	0.913 ± 0.064	0.881 ± 0.024	0.786 ± 0.039	2.00 / 8
ROOT (ours)	0.956 ± 0.023	0.955 ± 0.013	0.922 ± 0.013	1.00 / 8

Generating synthetic tasks:

Help identify low- and high-value design distributions more reliably

Offline optimization = Learning transport between 2 distributions

→ Mitigates data scarcity and erratic behavior in OOD

(Cuong Dao, Hung Tran, Le Nguyen, Nguyen Truong, Nghia Hoang, 2025)

Outline of Tutorial

- Introduction and Overview
- Forward Modeling Approaches
- Inverse Modeling Approaches
- Theoretical Analysis
- Offline Optimization in Small Data Setting
- Open Challenges

Theoretical Understanding ⇔ Algorithm Design

- Existing theory uses some structural aspect of the oracle objective function or the offline data distribution
  - to explicitly/implicitly characterize the optimization performance

- **Q1:** Understanding how particular structural aspects influence the (worst-case) optimization performance?
- **Q2:** How such an understanding can be leveraged to develop practical algorithms?

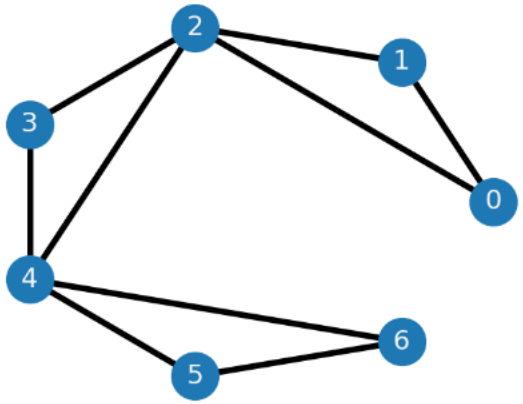
Theoretical Analysis: Forward Modeling

- Kuba et al., Functional Graphical Models: Structure Enables Offline Data-Driven Optimization
- Many forward models rely on pessimism/conservatism
  - However, strict pessimism limits you to "convex hull" of the training data => cannot find anything better than training data

- **Key Idea:** Exploiting ***structure*** helps us find better designs
  - Intuition: if objective function decomposes into smaller, interacting components, we can "stitch together" optimal parts

Theoretical Analysis: Forward Modeling

- **Key assumption:** function decomposes additively over cliques of the input graph



$$f(\mathbf{x}) = f_{0,1,2}(\mathbf{x}_{0,1,2}) + f_{2,3,4}(\mathbf{x}_{2,3,4}) + f_{4,5,6}(\mathbf{x}_{4,5,6})$$

Theoretical Analysis: Forward Modeling

- **Key assumption:** function decomposes additively over cliques of the input graph

- Regret (error) bound depends on $\frac{\pi^*(x)}{p(x)}$ where $\pi^*(x)$ is the optimal distribution and $p(x)$ is the data distribution over the entire input space
  - Naïvely, regret is large because ratio explodes

Using FGM assumption, regret depends only on the coverage of the marginals of the cliques $\max_{\mathbf{c}} \frac{\pi^*(x_C)}{p(x_C)}$

Theoretical Analysis: Gradient-Type Bound

The optimization quality depends on the discrepancy between the surrogate's and oracle's gradient structure

Let $R_s(\mathbf{x})$ & $R_o(\mathbf{x})$ denote the values of the recommended designs via m-step gradient search with step size λ using the learned surrogate and oracle, respectively.

Worst-case performance gap: $G_{m,\lambda} = \max_{\mathbf{x}} | R_s(\mathbf{x}) - R_o(\mathbf{x}) |$

Assume the oracle $g(\mathbf{x})$ is ℓ-Lipschitz and μ-smooth:

$$G_{m,\lambda} \leq m\lambda\ell(1 + \lambda\mu)^{m-1} \max_{\mathbf{x}} \| \nabla g(\mathbf{x}) - \nabla g(\mathbf{x}; \theta) \|$$

gradient gap

Theoretical Analysis: Gradient-Type Bound

The optimization quality depends on the discrepancy between the surrogate's and oracle's gradient structure

Worst-case performance gap: $G_{m,\lambda} = \max_{\mathbf{x}} | R_s(\mathbf{x}) - R_o(\mathbf{x}) |$

Assume the oracle $g(\mathbf{x})$ is ℓ-Lipschitz and μ-smooth:

$$G_{m,\lambda} \leq m\lambda\ell(1 + \lambda\mu)^{m-1} \max_{\mathbf{x}} \| \nabla g(\mathbf{x}) - \nabla g(\mathbf{x}; \theta) \|$$

gradient gap

Key insight: Matching gradient is more robust than matching value (see MATCH-OPT (Hoang et al., 2024))

Theoretical Analysis: Inverse Modeling

- Li et al., Diffusion Model for Data-Driven Black-Box Optimization
- **Assumption:** data lie on low-dimensional (m < d) manifold

$$x = Az$$

- Inverse approach: $x^* \sim p_\theta(x|y = a)$ where p_θ is learned via diffusion model
- Regret depends on three terms:

$$\epsilon_1 + \epsilon_2 + \epsilon_3$$

Estimation error

On-support distribution shift

Off-support Fidelity error

Theoretical Analysis: inverse approach

- Li et al., Diffusion Model for Data-Driven Black-Box Optimization
- Regret depends on three terms:

$$\epsilon_1 + \epsilon_2 + \epsilon_3$$

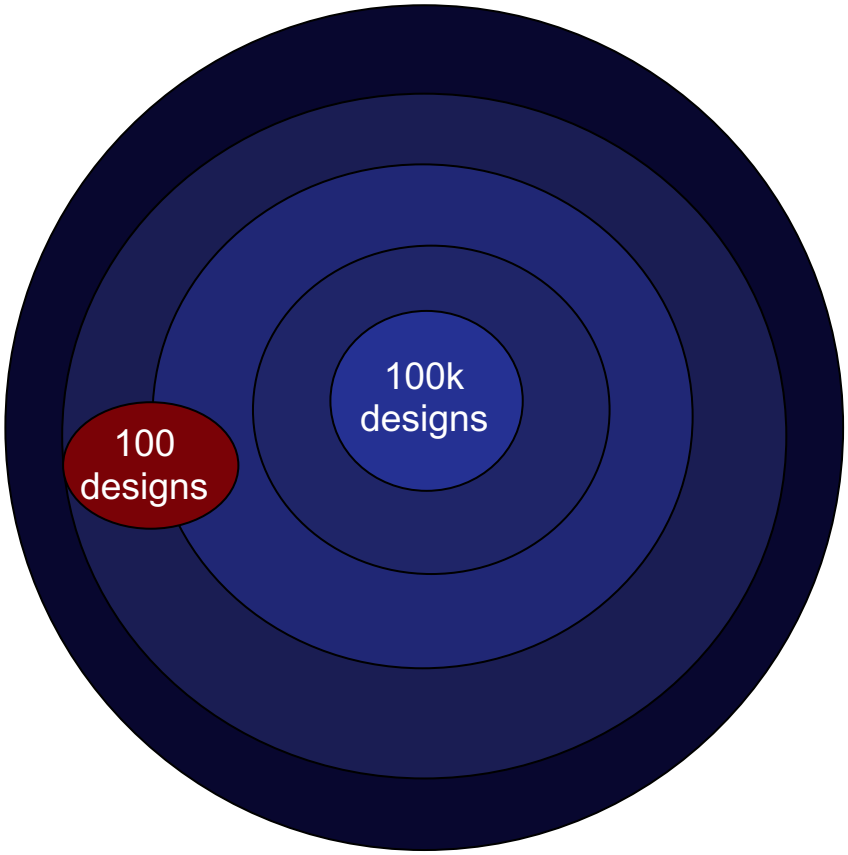
  - **Estimation error:** error of not knowing the true objective function which depends on latent dimension m
  - **On-support distribution shift:** error of forcing the diffusion model to generate samples that might be extremely rare or unseen in the training data
  - **Off-support Fidelity error:** penalty for leaving the manifold which is typically low for well-trained generative models

Outline of Tutorial

- Introduction and Overview
- Forward Approaches
- Inverse Approaches
- Theoretical Analysis
- Offline Optimization in Small Data Setting
- Open Challenges

The Small Data Challenge

Why Small Data is Different?



Why Standard ML fails?

- Overfitting and extrapolation
- Distribution Shift
- Prediction Bias

Labeled training data cover 0.1% of the design space

The Small Data Challenge

Why Standard ML fails?

Model fits observed 100 training examples too well but fails on unseen regions.

Overfitting and extrapolation

Distribution Shift

Prediction Bias

Solution: Synthetic pretraining on diverse functions

The Small Data Challenge

Why Standard ML fails?

Overfitting and extrapolation

Few training examples from poor-performing regions do not cover the optimization landscape

Distribution Shift

Prediction Bias

Solution: Generate synthetic data across full input space

The Small Data Challenge

Why Standard ML fails?

Overfitting and extrapolation

Distribution Shift

Prediction Bias

Standard ML optimizes value-matching, not the directional signals needed for optimization

Solution: Optimize for gradient matching, not value matching

Small Data Regime: Two Approaches

ExPT:

Synthetic Pretraining for Few-Shot Experimental Design

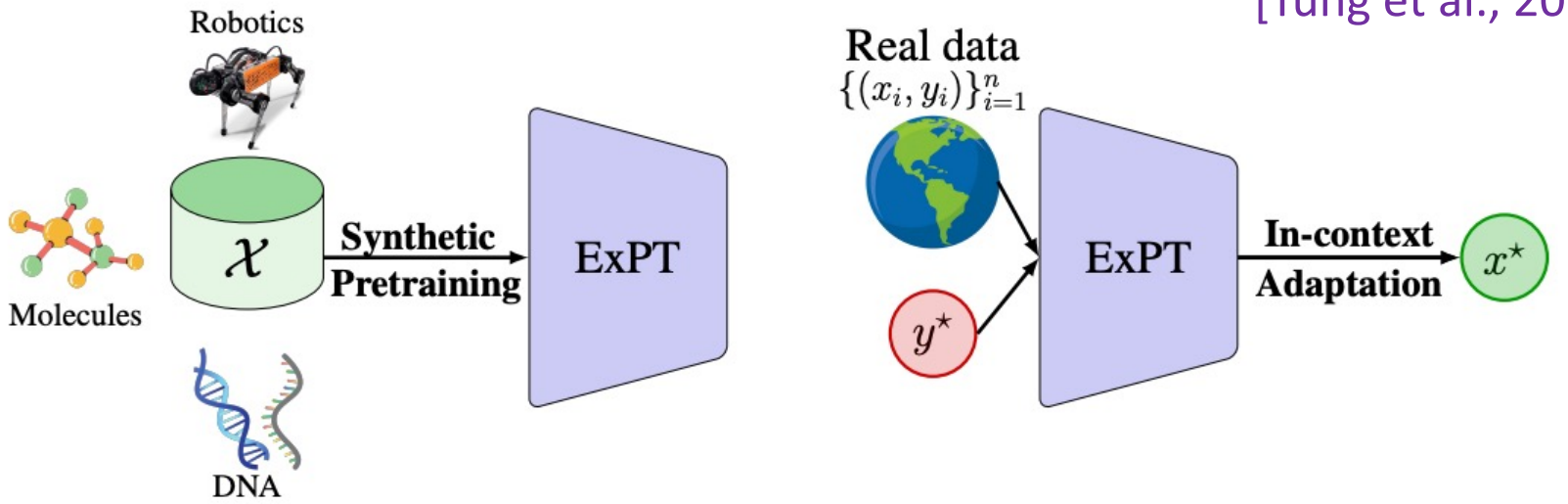
OptBias:

Surrogate Learning with Optimization Bias via Synthetic Task Generation

Let us explore how each approach tackles the small data challenge

ExPT: Synthetic Pretraining for Few-Shot Experimental Design

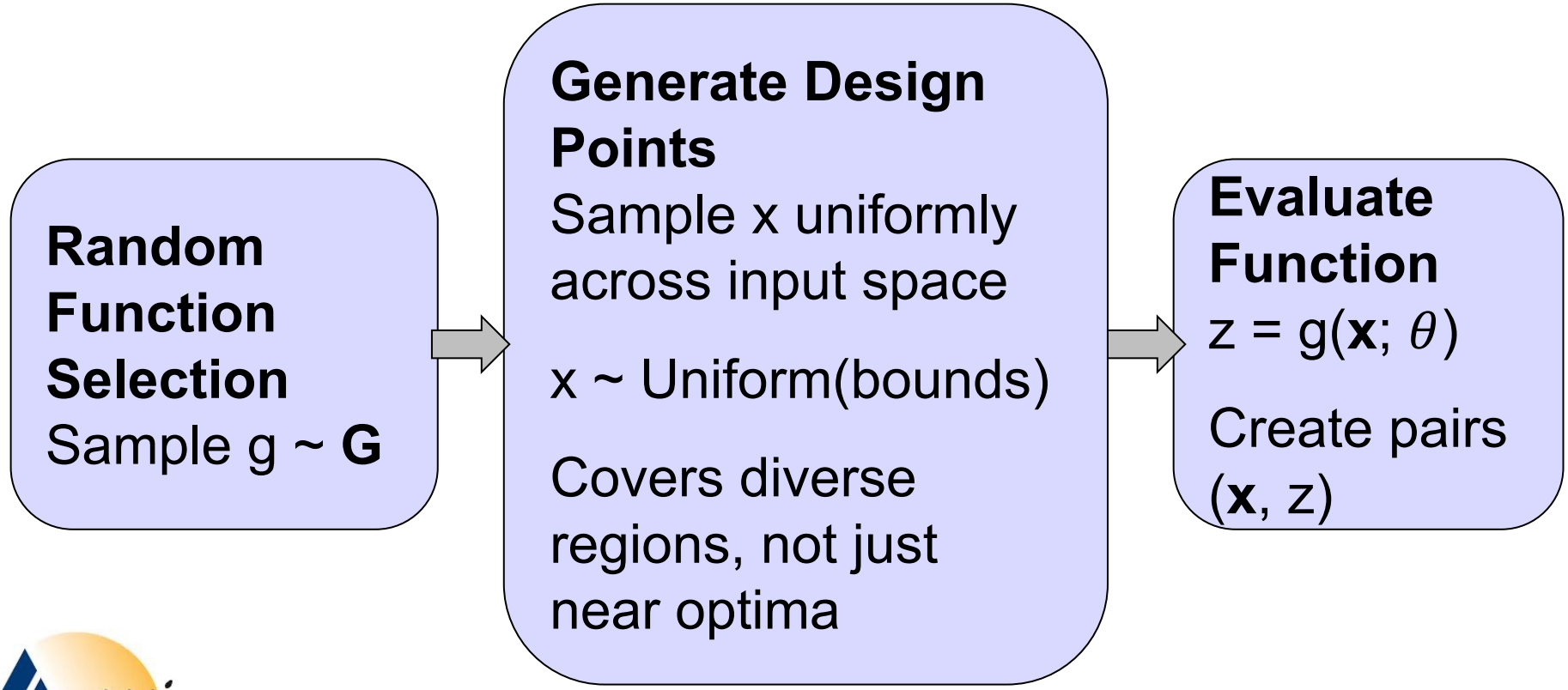
[Tung et al., 2024]



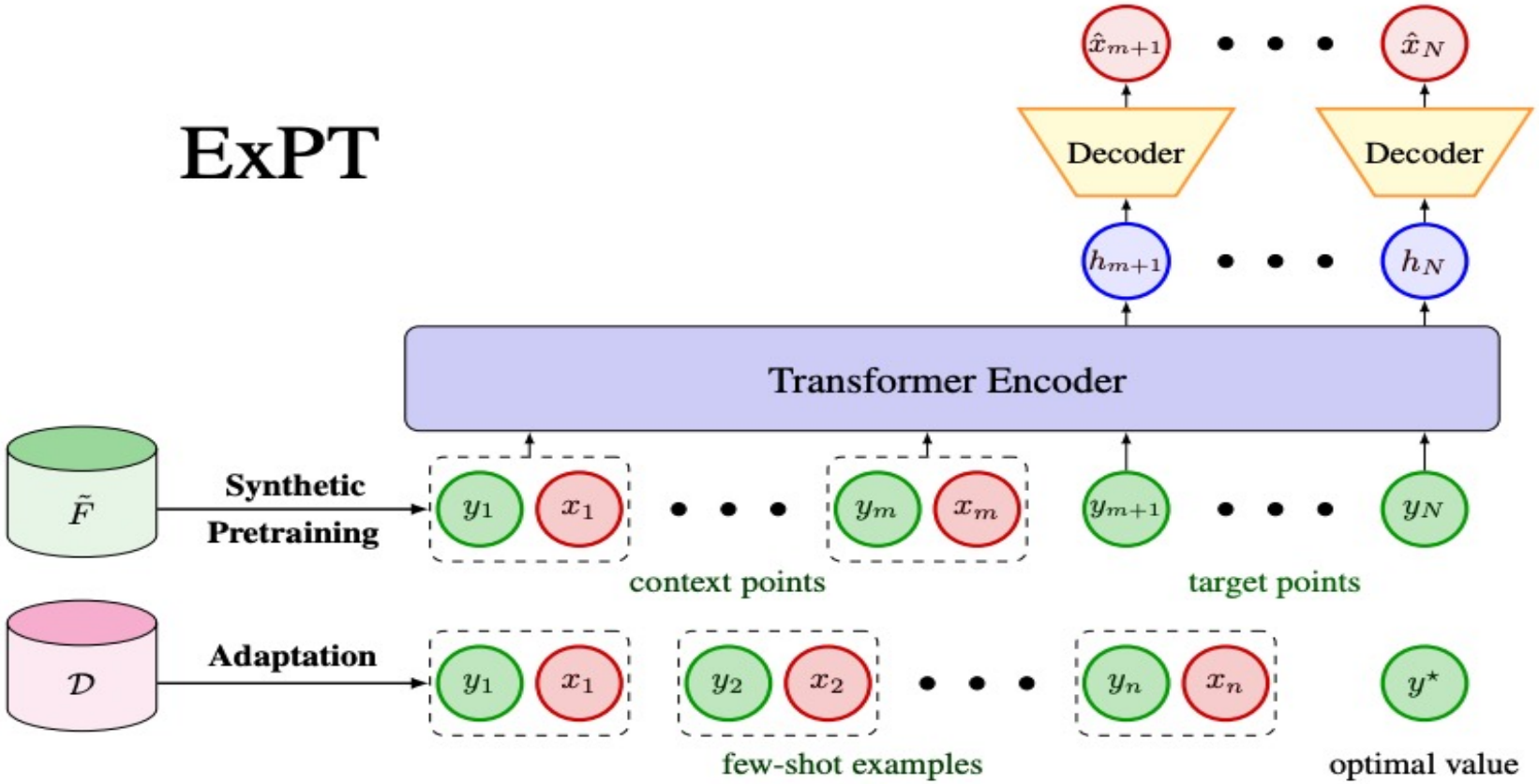
- **Pretraining:** Model learns optimization patterns from unlabeled designs across domains.
- **Adaptation:** Model conditions on few (design, score) pairs and target z* to generate optimal design **x***.

ExPT: Synthetic Pretraining for Few-Shot Experimental Design

Goal: Create diverse synthetic functions that cover different optimization landscapes

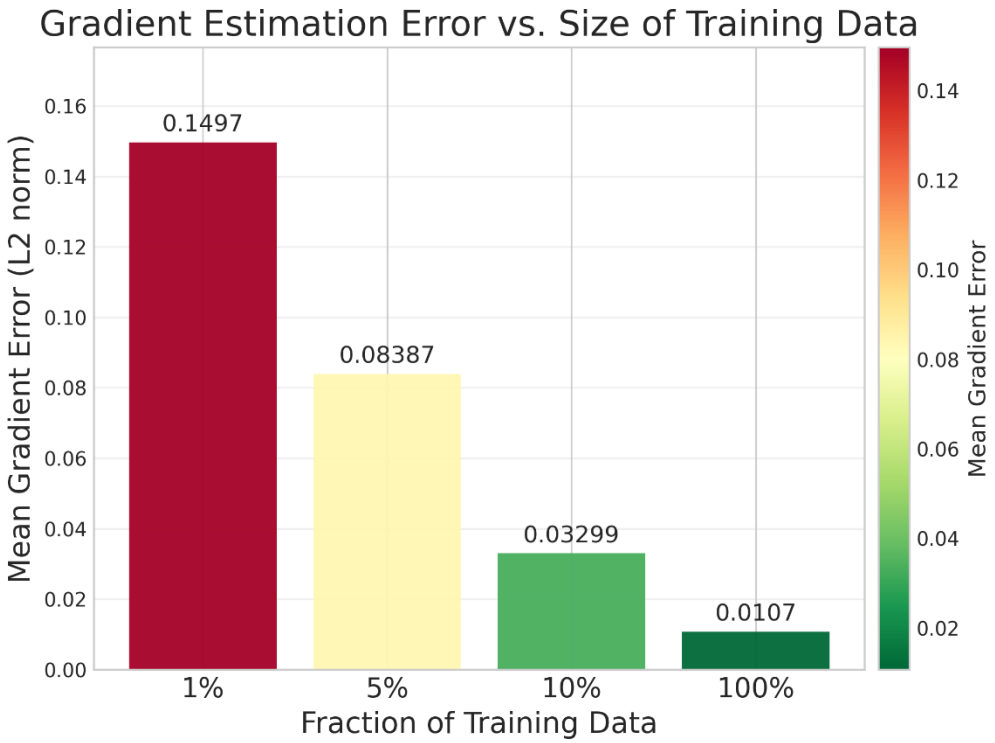


ExPT: Synthetic Pretraining for Few-Shot Experimental Design



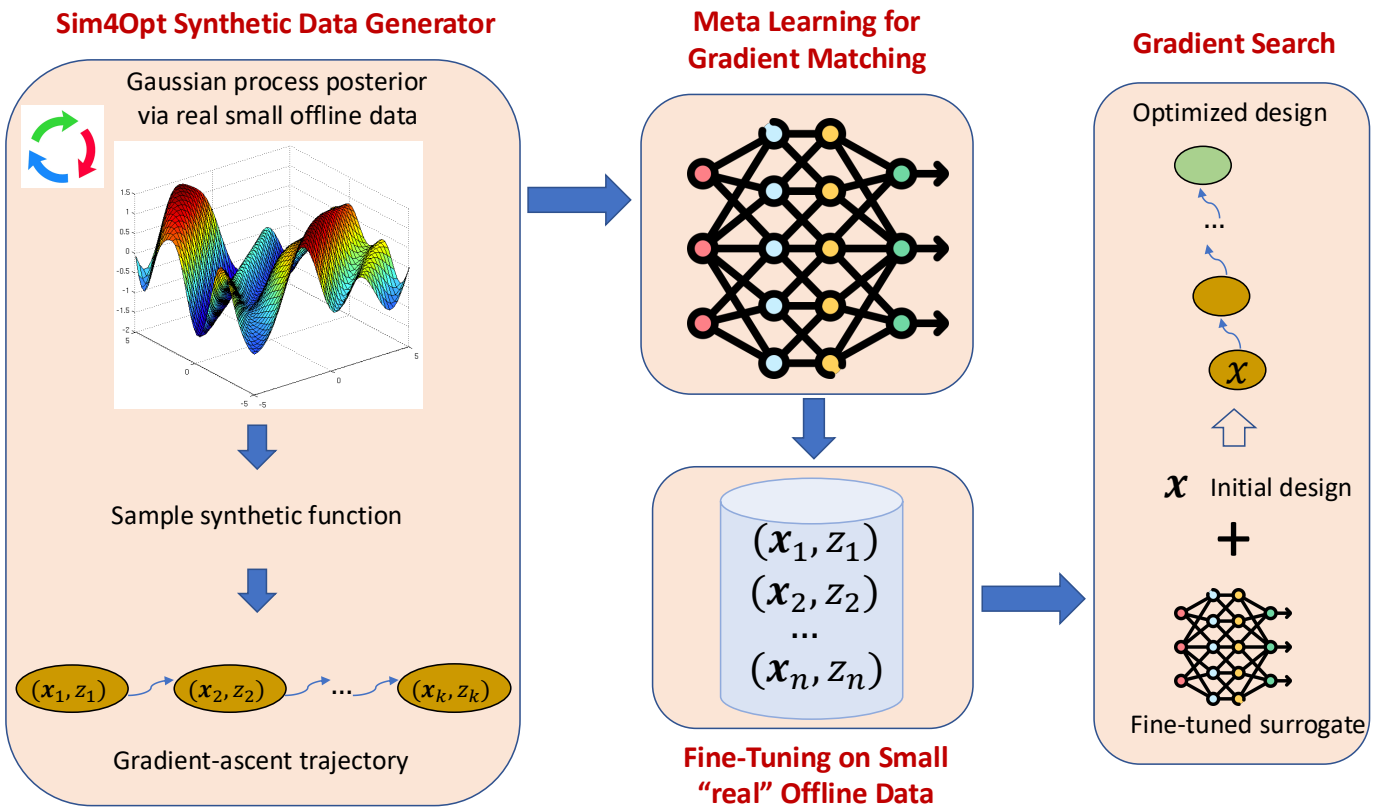
[Tung et al., 2024]

OptBias: Black-Box Optimization from Small Offline Datasets via Meta Learning



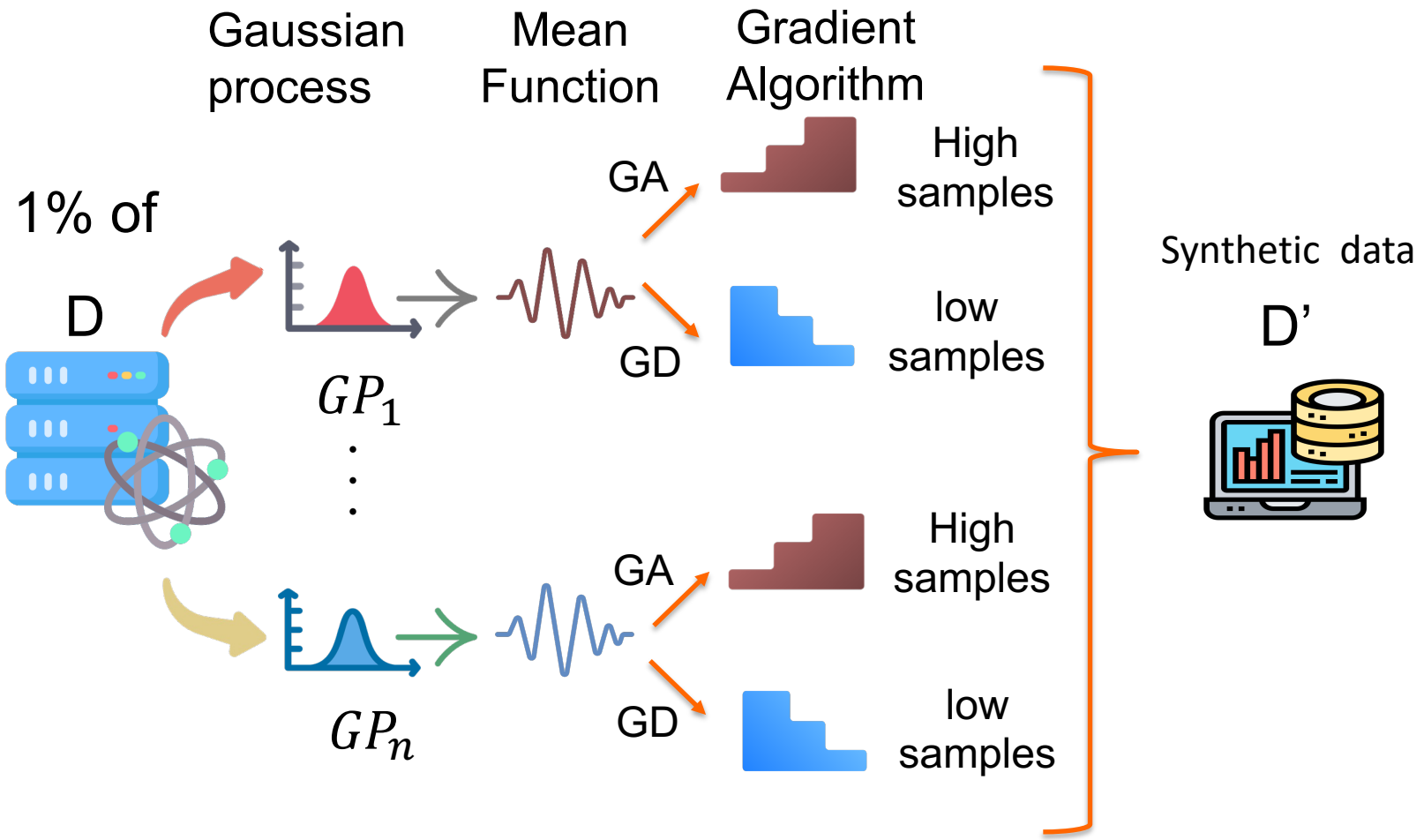
- Problem: With only 1% training data, gradient estimation error is very high (red bar)
- Solution: OptBias reduces this through better synthetic data generation and meta-learning

Overview of OptBias



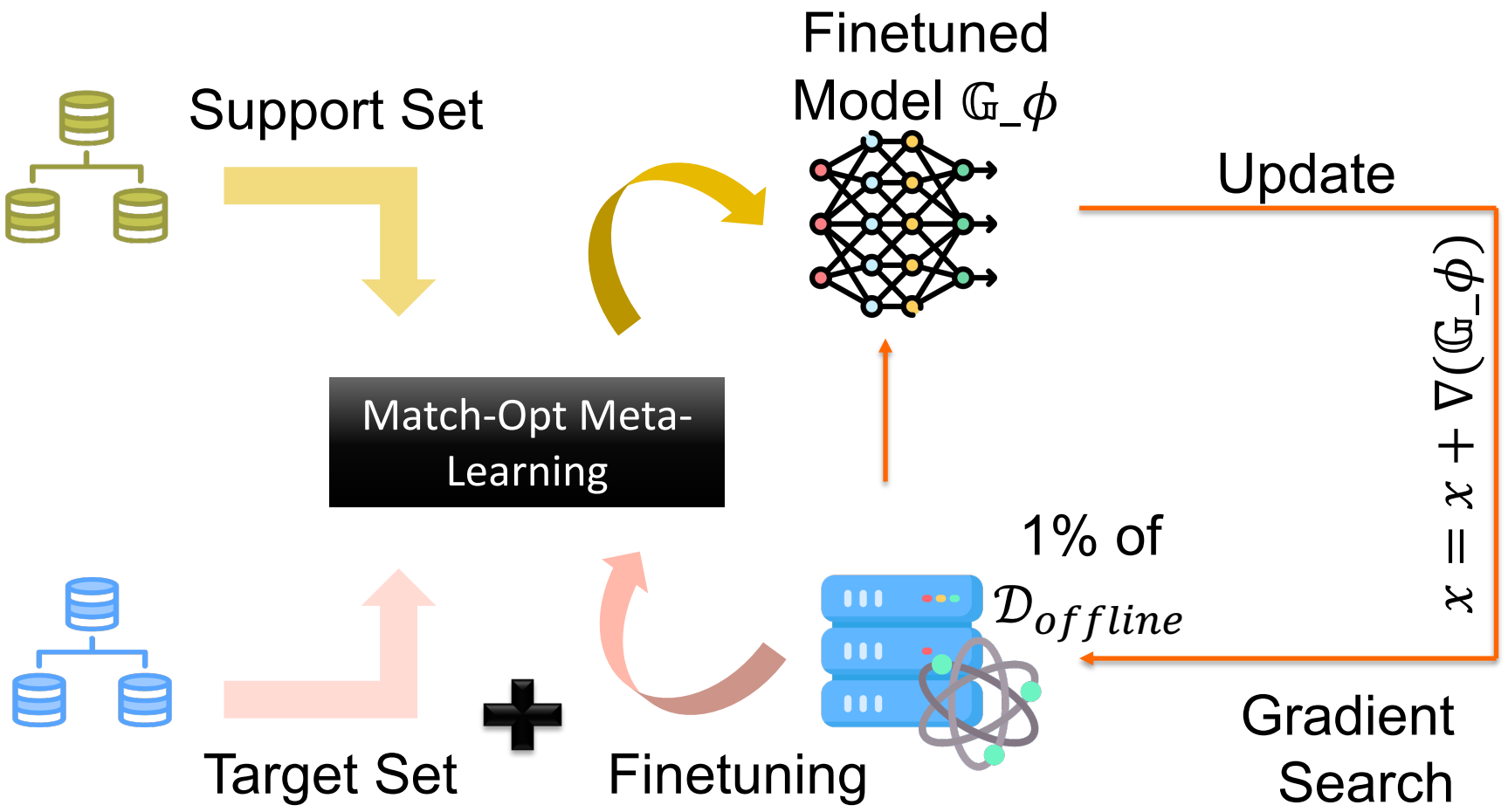
Key Idea: OptBias uses Sim4Opt to generate oracle-like synthetic functions, then applies meta-learning with gradient matching to align the surrogate with the true oracle's gradient field

Overview of OptBias



Step 1: Synthetic data generation

Overview of OptBias



Step 2: Training meta learning model via MATCH-OPT Loss

Step 3: Gradient Search

ExPT vs. OptBias

Attribute	ExPT	OptBias
Key Strategy	Synthetic pretraining + inverse modeling	Synthetic Pretraining + meta-Learning
Data Generation	Random GP sampling from unlabeled data	GP posterior with gradient -ascent
Model architecture	Transformer encoder + VAE decoder	Neural surrogate with meta learning
Optimization Focus	Predict optimal designs directly	Learn gradient field to guide search

Experimental Evaluation: ExPT vs. OptBias

Table 1: Performance achieved by baselines under the limited data settings (using 1% of the offline dataset) across various benchmark tasks.

	Benchmarks						
Method	**Ant**	**D'Kitty**	**TFBind8**	**RNA1**	**RNA2**	**RNA3**	**Mean Rank**
D_{best} from 1% offline data	0.123	0.307	0.124	0.028	0.027	0.066	None
GA	0.734 ± 0.054	0.831 ± 0.034	0.782 ± 0.117	0.475 ± 0.134	0.497 ± 0.091	0.317 ± 0.072	5.17
MINs	0.767 ± 0.084	0.895 ± 0.011	0.762 ± 0.135	0.076 ± 0.062	0.019 ± 0.029	0.132 ± 0.059	7.67
COMs	0.908 ± 0.042	0.705 ± 0.012	0.438 ± 0.000	0.279 ± 0.059	0.278 ± 0.120	0.293 ± 0.044	6.08
DEMO	0.800 ± 0.150	0.845 ± 0.037	0.677 ± 0.168	0.166 ± 0.034	0.141 ± 0.045	0.295 ± 0.043	6.50
LTR	0.865 ± 0.118	0.91 ± 0.026	0.45 ± 0.027	0.19 ± 0.189	0.215 ± 0.262	0.417 ± 0.176	5.17
Match-Opt	0.859 ± 0.007	0.912 ± 0.013	0.438 ± 0.002	0.129 ± 0.000	0.119 ± 0.000	0.166 ± 0.000	6.92
Batch BO (q-EI)	0.482 ± 0.066	0.816 ± 0.030	0.928 ± 0.026	0.490 ± 0.114	0.523 ± 0.103	0.462 ± 0.097	4.00
REINFORCE	0.327 ± 0.062	0.588 ± 0.162	0.872 ± 0.057	0.071 ± 0.054	0.062 ± 0.054	0.119 ± 0.064	8.83
ExPT	0.9276 ± 0.009	0.955 ± 0.009	0.879 ± 0.080	0.249 ± 0.013	0.239 ± 0.009	0.341 ± 0.041	3.33
OptBias (ours)	0.960 ± 0.017	0.947 ± 0.012	0.945 ± 0.024	0.503 ± 0.025	0.540 ± 0.113	0.450 ± 0.134	**1.33**

Table 2: Performance comparison between our proposed method OptBias, which uses meta-learning for surrogate training with synthetic data, and its variant, which replaces meta-learning with a pre-training procedure.

	Benchmarks					
Method	**Ant**	**D'Kitty**	**TFBind8**	**RNA1**	**RNA2**	**RNA3**
OptBias (pre-training)	0.884 ± 0.043	0.932 ± 0.007	0.438 ± 0.000	0.129 ± 0.000	0.119 ± 0.000	0.166 ± 0.000
OptBias (meta-learning)	0.960 ± 0.017	0.947 ± 0.012	0.945 ± 0.024	0.503 ± 0.025	0.540 ± 0.113	0.450 ± 0.134

Outline of Tutorial

- Introduction and Overview
- Forward Approaches
- Inverse Approaches
- Theoretical Analysis
- Offline Optimization in Small Data Setting
- Open Challenges

Open Challenges: #1 Better Benchmarks

- Current benchmarks (e.g., DesignBench) have served the community but important to set up new benchmarks with best practices

Open Challenges: #1 Better Benchmarks

- Setting up benchmarks require significant effort due to specific software requirements and dependency issues
  - Better tooling (see Bencher by Leonard Papenmeier and Luigi Nardi)
- Construction of surrogate models as Oracles for evaluation
  - Add explicit constraints
- Synthetic objective functions that are related to real-world problems
  - e.g., Ehrlich functions for biological sequence design
- Policy of collecting offline datasets
  - Ideally, should be related to real-world data collection
  - Random policies are not always a sound choice

Open Challenges: #1 Better Benchmarks

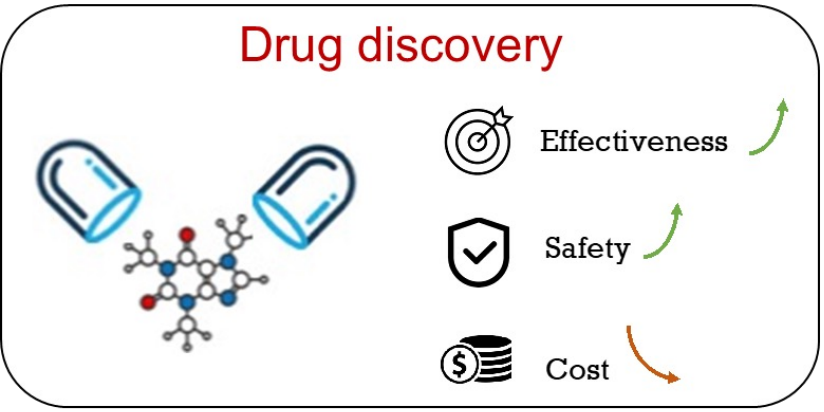
- Evolve benchmarks over time to avoid overfitting
  - Procedurally generated benchmarks
  - Lots of new ideas in LLM/GenAI literature

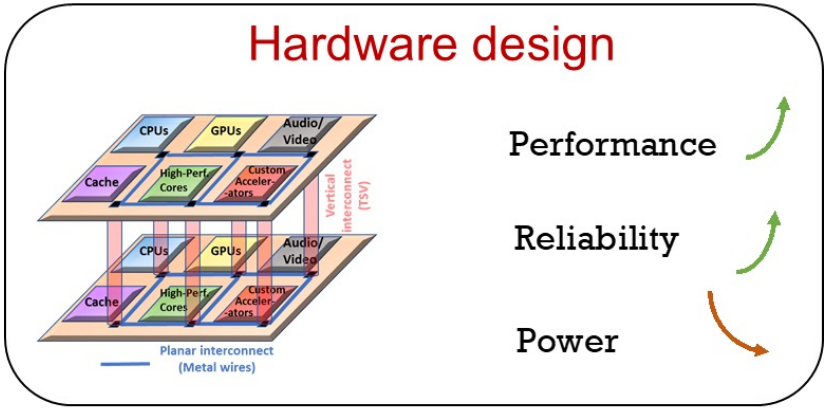
Open Challenges: #2 Hyperparameter Tuning

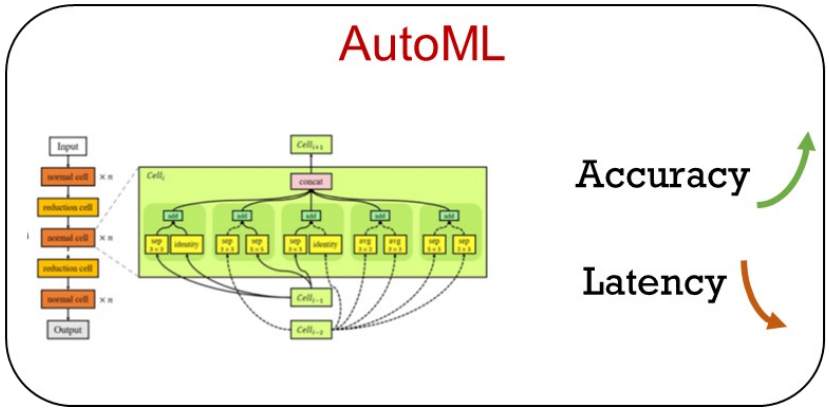
- Important requirement to only use offline dataset
  - Leverage ideas from offline RL (e.g., see Nie et al., Data-Efficient Pipeline for Offline Reinforcement Learning with Limited Data)

- Better methods for uncertainty quantification tied to the end-goal of design

Open Challenges: #3 Multi-Objective Optimization







- Important, but highly under-studied problem setting

Summary of Tutorial

- Introduction and Overview
- Forward Approaches
- Inverse Approaches
- Theoretical Analysis
- Offline Optimization in Small Data Setting
- Open Challenges

Nascent but important area. Lot more work needs to be done. Please join and contribute ☺

References: Forward Modeling

- Trabucco, Brandon, et al. "Conservative objective models for effective offline model-based optimization." *International Conference on Machine Learning*. PMLR, 2021. https://arxiv.org/abs/2107.06882
- Yu, Sihyun, et al. "Roma: Robust model adaptation for offline model-based optimization." *Advances in Neural Information Processing Systems* 34 (2021): 4619-4631. https://arxiv.org/abs/2110.14188
- Yao, Michael, et al. "Generative adversarial model-based optimization via source critic regularization." *Advances in neural information processing systems* 37 (2024): 44009-44039. https://arxiv.org/abs/2402.06532
- Dao, Manh Cuong, et al. "Boosting offline optimizers with surrogate sensitivity." *arXiv preprint arXiv:2503.04181* (2025). https://arxiv.org/abs/2503.04181

References: Forward Modeling

- Dao, Cuong, et al. "Incorporating surrogate gradient norm to improve offline optimization techniques." *Advances in Neural Information Processing Systems* 37 (2024): 8014-8046. https://arxiv.org/abs/2503.04242
- Hoang, Minh, et al. "Learning surrogates for offline black-box optimization via gradient matching." *arXiv preprint arXiv:2503.01883* (2025).https://arxiv.org/abs/2503.01883
- Chemingui, Yassine, et al. "Offline model-based optimization via policy-guided gradient search." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 10. 2024. https://ojs.aaai.org/index.php/AAAI/article/view/29001

References: Inverse Modeling

- Brookes, David H., and Jennifer Listgarten. "Design by adaptive sampling." *arXiv preprint arXiv:1810.03714* (2018).https://arxiv.org/abs/1810.03714
- Brookes, David, Hahnbeom Park, and Jennifer Listgarten. "Conditioning by adaptive sampling for robust design." *International conference on machine learning*. PMLR, 2019.https://arxiv.org/abs/1901.10060
- Mashkaria, Satvik Mehul, Siddarth Krishnamoorthy, and Aditya Grover. "Generative pretraining for black-box optimization." *International Conference on Machine Learning*. PMLR, 2023.https://arxiv.org/abs/2206.10786

References: Inverse Modeling

- Krishnamoorthy, Siddarth, Satvik Mehul Mashkaria, and Aditya Grover. "Diffusion models for black-box optimization." *International Conference on Machine Learning*. PMLR, 2023. https://arxiv.org/abs/2306.07180
- Dao, Manh Cuong, et al. "ROOT: Rethinking Offline Optimization as Distributional Translation via Probabilistic Bridge." *arXiv preprint arXiv:2509.16300* (2025). https://arxiv.org/abs/2509.16300
- Annadani, Yashas, et al. "Preference-Guided Diffusion for Multi-Objective Offline Optimization." arXiv preprint arXiv:2503.17299 (2025). https://arxiv.org/abs/2503.17299

References: Small Data Setting

- Nguyen, Tung, Sudhanshu Agrawal, and Aditya Grover. "Expt: Synthetic pretraining for few-shot experimental design." *Advances in Neural Information Processing Systems* 36 (2023): 45856-45869. https://arxiv.org/abs/2310.19961

References: Theoretical Analysis

- Grudzien Kuba, Jakub, et al. "Functional Graphical Models: Structure Enables Offline Data-Driven Optimization." *arXiv e-prints* (2024): arXiv-2401. https://proceedings.mlr.press/v238/grudzien24a/grudzien24a.pdf

- Li, Zihao, et al. "Diffusion model for data-driven black-box optimization." *arXiv preprint arXiv:2403.13219* (2024). https://arxiv.org/abs/2403.13219

References: Survey

- Kim, Minsu, et al. "Offline Model-Based Optimization: Comprehensive Review." arXiv preprint arXiv:2503.17286 (2025). https://arxiv.org/abs/2503.17286

References: Benchmarks and Evaluation

- Trabucco, Brandon, et al. "Design-bench: Benchmarks for data-driven offline model-based optimization." *International Conference on Machine Learning*. PMLR, 2022.

  https://arxiv.org/abs/2202.08450

- Package, ViennaRNA. "The ViennaRNA Package." *Algorithms for Molecular Biology* 6 (2011): 1-26.

  https://www.tbi.univie.ac.at/RNA/?l+RNAcofold