

Data-Driven Design Optimization — Generative-Based

Learn where good designs live — then search, refit, condition or steer

Jinkyoo Park · KAIST

Offline, part 2 of 2 — a generator decides where to look; a performance model decides what to trust

Where we are — from judging a design to proposing one

LECTURE 4 · OBSERVE		LECTURE 5 · TRUST		LECTURE 6 · GENERATE
the question	which experiment to run next	on what evidence to judge a design no one has measured		which designs to propose, and how to steer that toward good ones
the model	a GP of f	a performance model $\hat{f}$, trusted with care		a generator of plausible designs, plus a performance model
the decision	query the acquisition's arg-max	return the surrogate's arg-max, carefully		search, refit, condition or steer the generator

Lecture 5 gave the performance model grounds for trust. This lecture adds a model of **where to look**. **A generator structures the search; it does not create evidence about performance** — that still comes from the data, and from Lecture 5's tools.

Learning route — two sessions, four places to intervene

Bring: the surrogate, the decision bound and how an optimiser exploits it (Lecture 5); Bayes' rule and likelihood (Lecture 2); sampling and weighting (Lecture 3).

SESSION	WHAT IT DOES
1 • Generate, then search or refit	What surrogates leave → two models, two jobs → KL, VAE and diffusion as tools → G1: search in learned coordinates → G2: refit the generating distribution, with CbAS worked through
2 • Ask, and steer	G3: condition on the performance you want — CVAE, MINs, DDOM → a KL-regularised view of candidate distributions → G4: steer the sampling process → the four side by side, with Lecture 5's tools in each

Keep three questions apart throughout — Lecture 5 closed on them: **is it good** (performance), **is it possible** (validity), **can the prediction be trusted** (reliability). Each method buys one and can spend another.

— SESSION 1 · ACT 1

What surrogates still leave

Q1. Lecture 5 gave the surrogate grounds for trust. Three problems are still standing.

The problem, unchanged — and what the algorithm may use

The same problem as Lecture 5, with the valid set written in:

$$\text{find } x^* = \arg \max_{x \in \mathcal{X}_{\text{valid}}} f(x) \quad \text{with only } D = \{(x_i, y_i)\}_{i=1}^N, \quad y_i = f(x_i) + \epsilon_i$$

WHAT THE ALGORITHM MAY USE

The fixed dataset, and any knowledge stated up front — a known constraint can be checked directly. **Strict offline:** f is never called while training, choosing or tuning.

WHAT IT MAY NOT

The hidden oracle that scores the final candidates — as in Lecture 5's Design-Bench. The widgets draw the true f for the teacher's eyes only.

What Lecture 5 left standing

Q1 What do surrogates leave?

Q2 How do we model designs?

Q3 Coordinates or distribution?

Q4 Condition or sampler?

A HUGE SPACE

The optimiser still moves one point through every coordinate of x — a protein of 237 residues has 20^{237} sequences. Conservatism made each move safer, not the space smaller.

VALIDITY

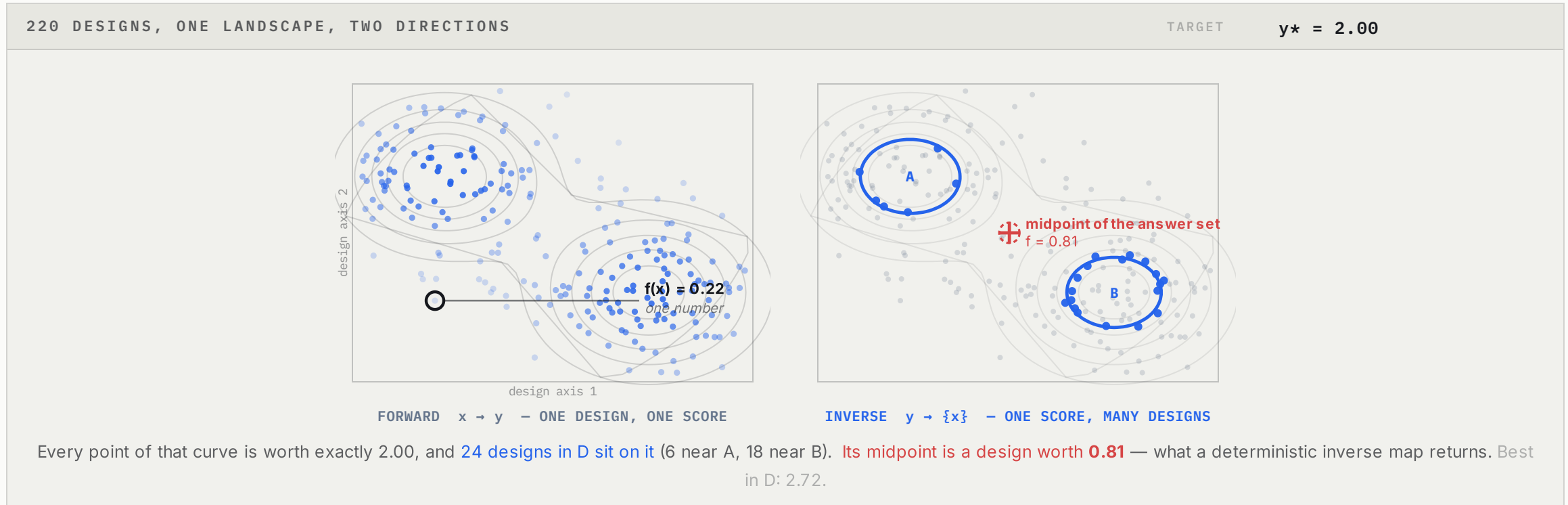
Most of x is not a design at all — not a molecule, not a folded protein, not an image of a digit. Lecture 5's last slide: **out of distribution and invalid are different problems**, and a surrogate scores both.

ONE ANSWER

The arg-max returns one design; a lab wants a batch of different good candidates. Reliable scoring alone does not provide convenient coordinates, or a distribution of diverse, plausible candidates.

The source's answer: **model what plausible designs look like, and steer that model toward good ones** — a distribution over designs, plus a performance signal.

One y , many x — the average answer can be invalid



Run the forward map backwards — from the y you need to the x that delivers it — and it is not a function: **a curve in two pieces**. The red mark is the midpoint of that set — what a least-squares inverse returns — stranded in the valley between the branches.

To keep several answers, make the inverse **stochastic**: $x = f_{\theta}^{-1}(y, z)$ with $z \sim p(z)$ choosing among them. That is a generative model.

Two models, two jobs

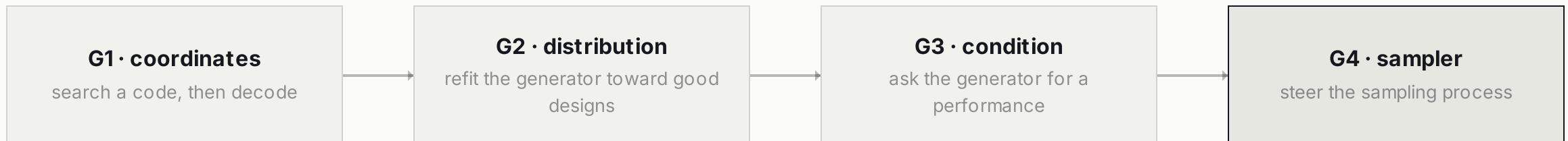
THE PERFORMANCE MODEL — LECTURE 5

How good is this candidate? Trained on pairs (x, y) . Wrong off the data, and exploited by whatever searches it — so Lecture 5 gave it grounds for trust: conservatism, uncertainty, knowledge.

THE GENERATOR — THIS LECTURE

Which candidates should we look at? Trained on designs x (sometimes with y). It offers plausibility, diversity and coordinates — but alone it only samples designs **like the data**, mediocre ones included.

A generator is not an optimiser: **every generative optimiser needs a performance signal** — the dataset's labels or a performance model. Where can it enter? Four places, G1–G4 — taken up after the tools:



Check — reading the function backwards

Lecture 5 learned $\hat{f} : x \mapsto y$ and searched it. A score-conditioned inverse model — one of this lecture's four routes — learns the other direction. What makes that harder than it sounds?

- ☒ A The map is many-to-one, so the "inverse" is not a function but a *distribution* over designs achieving a target
- ☐ B The inverse is more expensive to evaluate
- ☐ C The inverse cannot be represented by a neural network
- ☐ D The training data would have to be collected in reverse

Many different designs can score the same y . A model trained to regress x from y under a squared loss returns the **average** of those designs — very often not a valid design at all. That is why the answer has to be a generative model that samples, rather than a regressor that predicts.

— SESSION 1 • ACT 2

Generative models, compressed

Q2. Before steering a generator toward good designs, you need one that produces designs at all — and a way to say how close two distributions are.

Comparing two distributions — the KL divergence

Q1 What do surrogates leave?

Q2 How do we model designs?

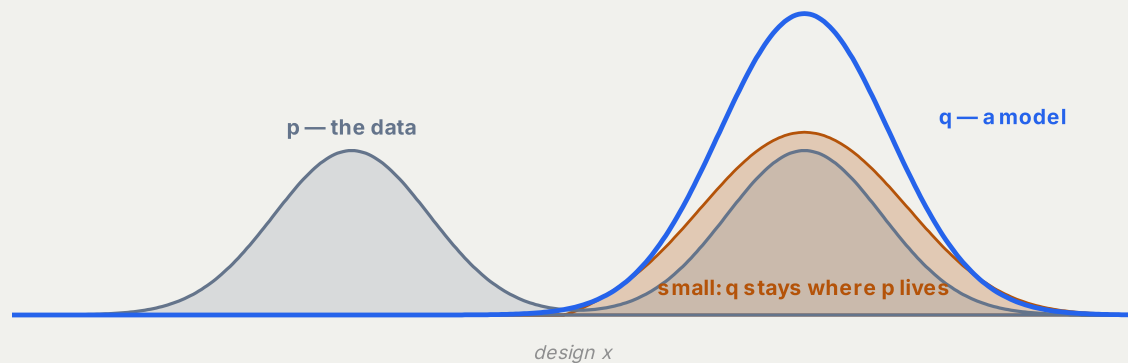
Q3 Coordinates or distribution?

Q4 Condition or sampler?

$$D_{\text{KL}}(a \parallel b) = \mathbb{E}_{x \sim a} \left[\log \frac{a(x)}{b(x)} \right] \geq 0, \quad = 0 \text{ only when } a = b.$$

WHICH WAY ROUND MATTERS

DEFINITION $\text{KL}(a \parallel b) = \mathbb{E}_a[\log a/b]$



$\text{KL}(p \parallel q)$ — average over the data **6.43**

$\text{KL}(q \parallel p)$ — average over the model **0.69**

$\text{KL}(q \parallel p)$ averages over the model, so q pays only for going where p does not. The tilt of Session 2 uses this direction: **q must stay where the generator lives** — and is infinite if q puts mass where p has none.

Press → twice: average with the data's eyes, then with the model's.

This lecture uses both directions: fitting a model averages over the data — **cover the data**; keeping a search near the generator averages over the search — **stay where the generator lives**.

The goal — a distribution concentrated on plausible designs

THE GOAL

Train $p_\theta(x)$ to match the data distribution $p_{\text{data}}(x)$, given only samples $\{x_i\}_{i=1}^N \sim p_{\text{data}}$.

$$\arg \min_{\theta} D_{\text{KL}}(p_{\text{data}} \parallel p_{\theta}) = \arg \max_{\theta} \mathbb{E}_{p_{\text{data}}} [\log p_{\theta}(x)]$$

The first term of the KL does not depend on θ : fitting a generative model **is** maximum likelihood — Lecture 2's principle, for a model whose likelihood has no closed form.

What is learned is **where plausible designs concentrate** — not a feasibility certificate. A learned density puts low, not zero, probability away from the data, and like Lecture 5's surrogate it is unconstrained where there is no data (Backup 4). Known constraints still need a check, or a decoder that enforces them.

VAE — encode, decode, and an objective with two terms

Assert that designs come from a few simple hidden factors: $p(x) = \int p_\theta(x | z) p(z) dz$, $z \sim \mathcal{N}(0, I)$. A decoder $p_\theta(x | z)$ produces the design; an encoder $q_\phi(z | x)$ approximates the posterior $p_\theta(z | x)$ — Lecture 3's hidden-state inference, done by a network.

$$\mathcal{L}(x) = \underbrace{\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x | z)]}_{\text{reconstruct the design}} - \underbrace{D_{\text{KL}}(q_\phi(z | x) || p(z))}_{\text{keep codes near the prior}}$$

GENERATING

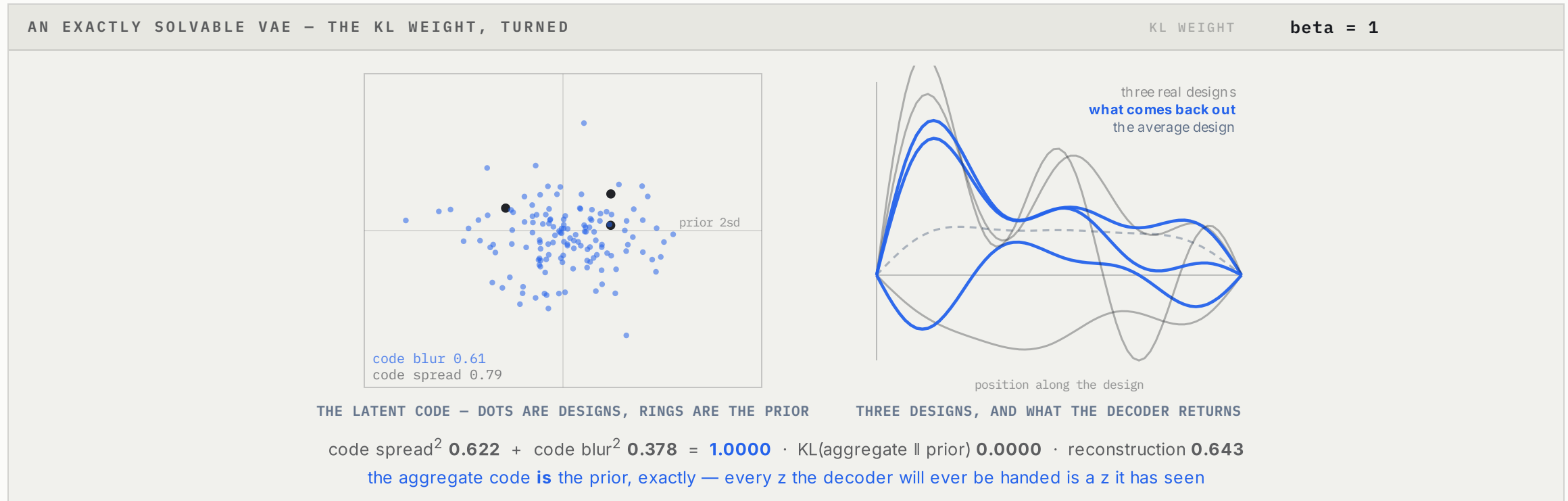
Draw $z \sim \mathcal{N}(0, I)$, decode. No data, no optimiser. The reparameterisation $z = \mu_\phi + \sigma_\phi \odot \epsilon$ lets gradients through the sample.

WHICH PRIOR?

$p(z)$ is a prior over **codes**, not Lecture 2's prior over parameters: the network's weights are point estimates, and drawing z does not express uncertainty about them.

The KL term makes the latent space **samplable**: codes from the prior decode to plausible designs. The latent space is a **coordinate system for designs** — G1 will search in it.

Turning the dial



An exactly solvable VAE with a weight β on the KL term. Each dot is a design's code, each halo its blur, the ring the prior. At $\beta = 1$ the codes fill the prior exactly; below it they spill outside, above it they collapse to a dot.

Diffusion — learn the noise, then remove it step by step

Ho, Jain & Abbeel · NeurIPS 2020

A VAE stacked T times, with the encoder **fixed** to a Gaussian noising process and the latent the size of the data — so training needs no inference network.

TRAIN — CORRUPT ONCE, PREDICT THE NOISE

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad L_{\text{simple}} = \mathbb{E} \left\| \epsilon - \epsilon_{\theta}(x_t, t) \right\|^2$$

Draw a clean design, corrupt it by a known amount, and **predict the noise that was added** — a supervised problem at every noise level.

GENERATE — START FROM NOISE, DENOISE

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(x_t, t) \right) + \sigma_t z$$

Start from $x_T \sim \mathcal{N}(0, I)$, not from a training design; each step is **a small denoising step**.

Denoising steps are steps of the algorithm, not physical time — and each one is a place where the sampler could be **steered**: Session 2 shows two ways.

Check — turning the KL weight

In a β -VAE, you raise β , the weight on the KL term. What do you get?

- ☐ A Sharper reconstructions and a more structured latent space
- ☐ B Sharper reconstructions at the cost of latent structure
- ☒ C A latent space closer to the prior and easier to sample from, at the cost of blurrier reconstructions
- ☐ D No change in either — β only affects training speed

The KL term pulls the codes toward the prior; the reconstruction term pulls them apart so that designs stay distinguishable. Raising β wins the first and loses the second: prior samples decode to plausible designs, but detail is lost. It is the same trade as Lecture 5's conservatism dial — fidelity to the data against freedom to generate.

— SESSION 1 • ACT 3

Coordinates and distribution

Q3. A generator and a performance model are on the table. Where does the performance signal get in?

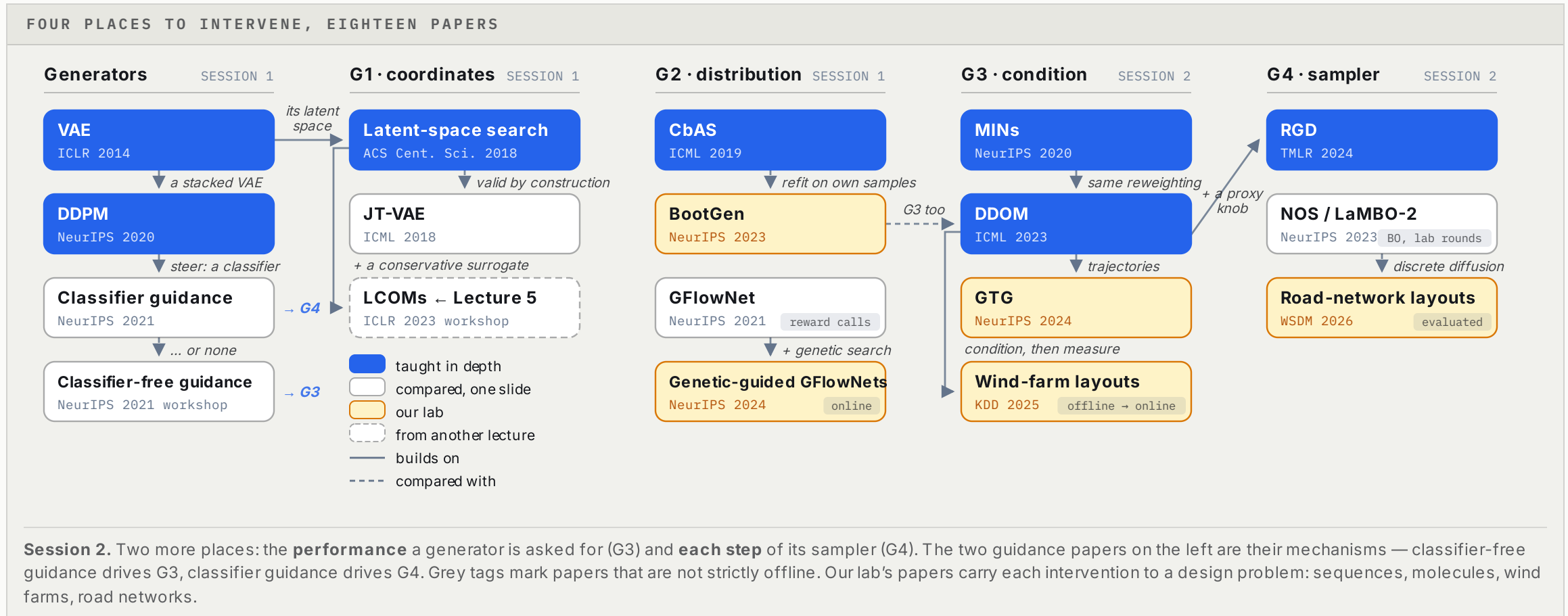
Four places to intervene



	WHERE THE PERFORMANCE SIGNAL ENTERS	WHAT CHANGES	IN DEPTH	WHERE LECTURE 5'S FAILURE CAN RE-ENTER
G1 · coordinates	the search over codes	search a code z ; decode $x = g(z)$	latent search <i>(2018)</i>	the predictor on codes is a surrogate
G2 · distribution	the generator's parameters	refit $p(x)$ toward predicted success	CbAS <i>(2019)</i>	the refit follows the success model's errors
G3 · condition	the generator's input	ask $p(x \mid y)$ for a high y	MINs <i>(2020)</i> , DDOM <i>(2023)</i>	a requested y beyond the data
G4 · sampler	the sampling process	performance information moves the sample (a gradient) or sets the guidance strength	classifier guidance <i>(2021)</i> , RGD <i>(2024)</i>	the sampler follows the performance model where it is wrong

Intervention points, not rival classes: a method can use several. For each, ask **what it assumes, what it solves, what it leaves** — and wherever a surrogate enters, Lecture 5's three streams still apply.

The papers of this lecture — why each is here



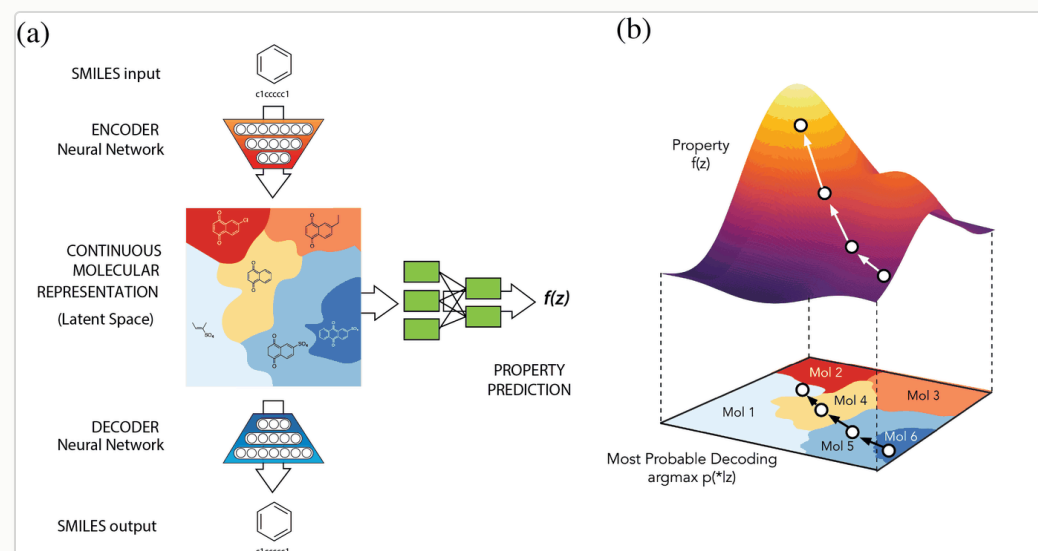
Press → for Session 2's papers; click any paper for what it does, why we read it and how it connects. The same list is a table in the appendix.

G1— search in learned coordinates

Gómez-Bombarelli et al. • ACS Central Science 2018

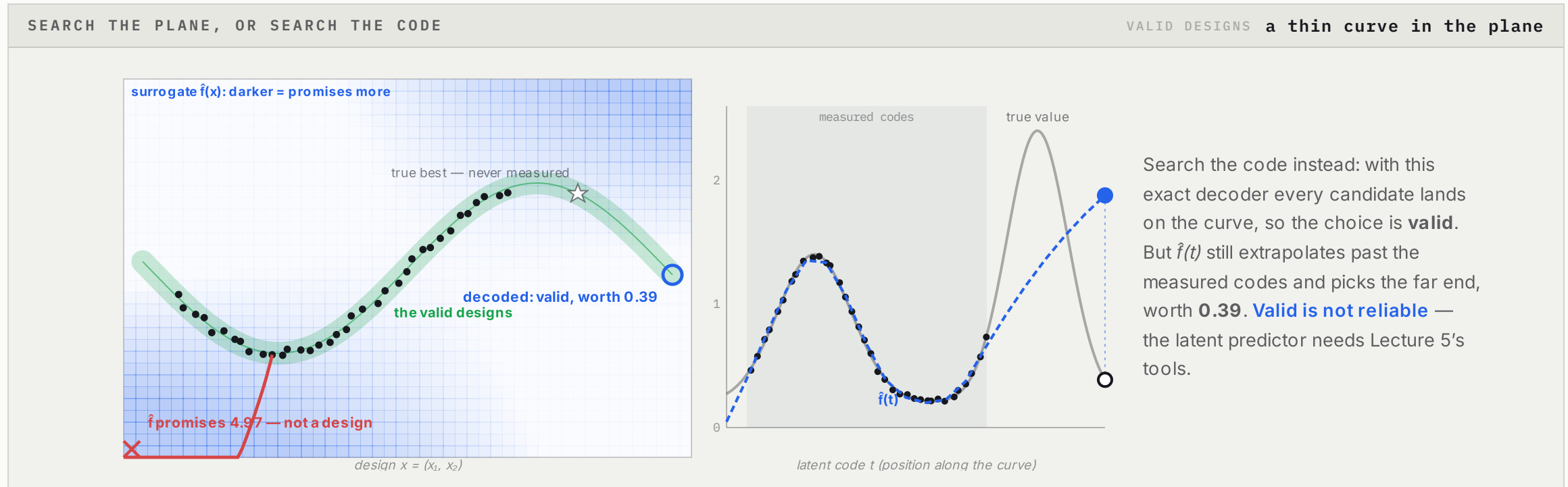
Lecture 4 met this paper as BO in a learned space. A VAE turns molecules (SMILES strings) into continuous codes, with a property predictor trained on the codes; search the codes — by gradient steps or Bayesian optimisation — and decode the best.

The search becomes low-dimensional and continuous, and every point it visits decodes to *something*. Whether that something is valid, and whether the predictor is right about it, are two separate bets.



Encode, predict, optimise in the latent space, decode. (Gómez-Bombarelli et al., 2018)

Search the plane, or search the code



One 2-D design problem whose valid designs form a thin curve. Press $\rightarrow$ twice: climb the surrogate in the plane, then search the code. The decoder here is the curve itself, known exactly — a teaching device; a learned decoder is only approximately valid.

What latent search buys, and what it costs

ASSUMPTION ADDED

The latent space is **faithful**: nearby codes decode to similar, valid designs, and the predictor on codes is honest.

WHAT IT SOLVES

A continuous, low-dimensional search whose candidates are designs — validity moves into the decoder.

WHAT IT LEAVES

Codes the decoder handles badly decode to **invalid** designs; the latent predictor is a surrogate, **exploitable** exactly as in Lecture 5.

THE FIX	HOW	PAPER
validity by construction	generate a tree of chemically valid substructures, then assemble — every output is a valid molecule (valid chemistry, not yet synthesisable or good)	JT-VAE (<i>Jin, Barzilay & Jaakkola, ICML 2018</i>)
reliability, from Lecture 5	train the predictor on codes with COMs' penalty	LCOMs (<i>Qi et al., 2023 workshop</i>)

Validity and reliability are repaired separately: a better decoder for one, Lecture 5's tools for the other.

G2 — condition the generator on success

Move the distribution, not a point. With the **generator as the prior** and a **success model** as the likelihood — S : "meets the requested performance" — Bayes' rule gives the designs we want:

$$p(x \mid S) = \frac{P(S \mid x) p(x)}{\sum_{x'} P(S \mid x') p(x')}$$

THE PRIOR KEEPS IT PLAUSIBLE

Designs the generator finds implausible keep little mass, however promising the success model says they are.

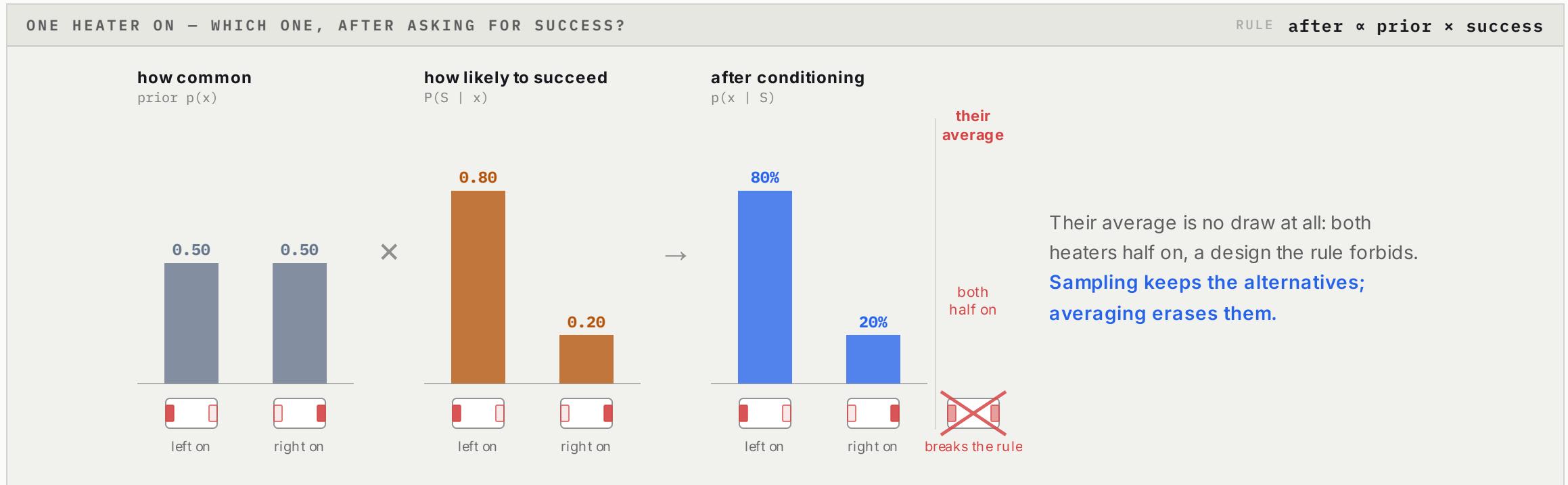
THE SUCCESS MODEL IS LECTURE 5'S

Offline, $P(S \mid x)$ is a **prediction** — e.g. $P(\hat{y} \geq \gamma \mid x)$ from a surrogate. Its errors are inside the target; a pessimistic one ($\mu - \kappa\sigma$) gives a pessimistic target.

Lecture 3 conditioned a fixed model on observed evidence. Here the "evidence" is a requested outcome, judged by a model — **the target is only as good as the success model**.

Temperature thread — condition a distribution over heater designs

shared teaching example · predict before revealing



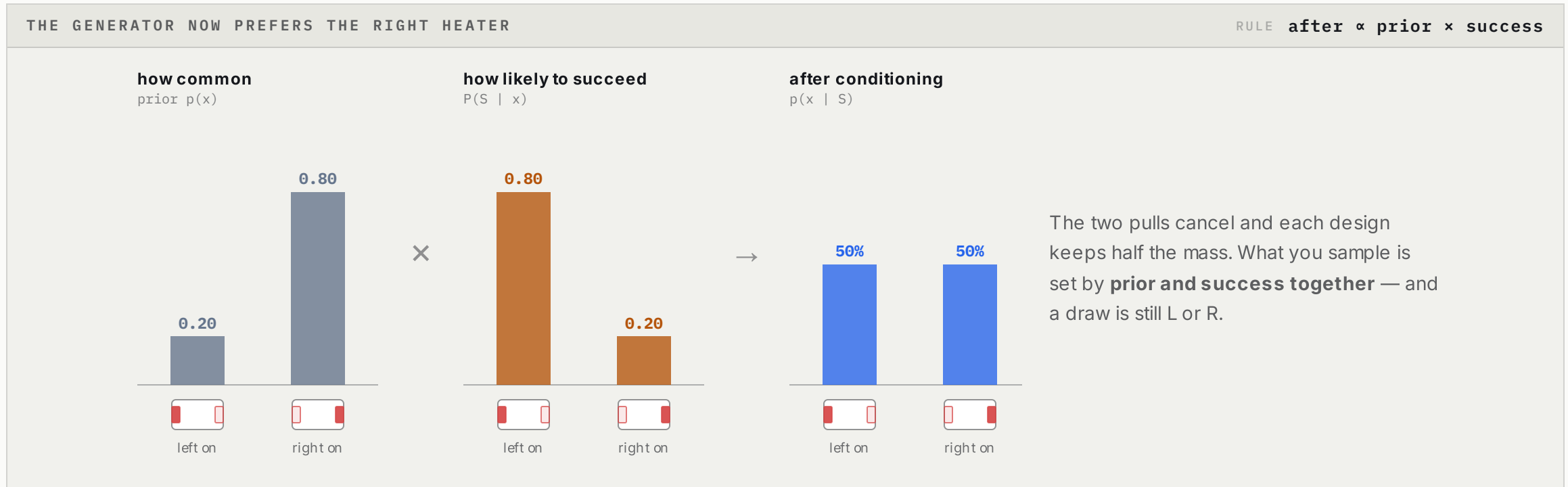
Lecture 0's heater as a discrete design: which of two heaters to switch on, instead of how much command to apply. Exactly one may be on.

Predict: After conditioning on success, which design becomes more likely — and is the average of the two a valid design?

The left heater: conditioning multiplies the prior by the success probability. And no — the average turns both heaters half on. Press → to see both.

Try it — the generator now prefers the right heater

work independently · reveal only after deciding



The same success model; the generator now proposes the right heater four times as often.

Which design is more likely after conditioning on success?

Check your answer. Neither — the two pulls cancel, half and half. What you sample is set by the prior and the success model **together**, and every draw is still L or R, never their average. Press → to see it.

CbAS — fit a generator to the conditional

Conditioning by Adaptive Sampling · Brookes, Park & Listgarten · ICML 2019

The target, with the trained generator $p(x \mid \theta^{(0)})$ as the prior, cannot be sampled directly — so fit a second generator $q(x \mid \phi)$ to it, averaging over the target (cover it):

$$p(x \mid S, \theta^{(0)}) = \frac{P(S \mid x) p(x \mid \theta^{(0)})}{P(S \mid \theta^{(0)})}, \quad \phi^* = \arg \min_{\phi} D_{\text{KL}}(p(x \mid S, \theta^{(0)}) \parallel q(x \mid \phi))$$

Derivation. Expand the KL and drop what does not depend on ϕ :

$$D_{\text{KL}} = -\mathbb{E}_{p(x|S, \theta^{(0)})} [\log q(x \mid \phi)] - H_0 = -\frac{1}{P(S \mid \theta^{(0)})} \mathbb{E}_{p(x|\theta^{(0)})} [P(S \mid x) \log q(x \mid \phi)] - H_0$$

$$\implies \phi^* = \arg \max_{\phi} \mathbb{E}_{p(x|\theta^{(0)})} [P(S \mid x) \log q(x \mid \phi)]$$

Weighted maximum likelihood: draw designs from the generator, weight each by how likely it is predicted to succeed, refit. Any generative model can be trained this way, and q need not be of the prior's form.

Why one shot fails — Lecture 3's rare cause, again

Draw from the prior and weight by $P(S \mid x)$: success is **exceedingly rare**, so almost every weight is near zero — Lecture 3's likelihood weighting with a rare cause, where almost no draw lands where the evidence points.

SAMPLE FROM LAST ROUND'S MODEL

Use the previous fit as the proposal, $r^{(t)} = q(x \mid \phi^{(t-1)})$, and correct with the importance ratio:

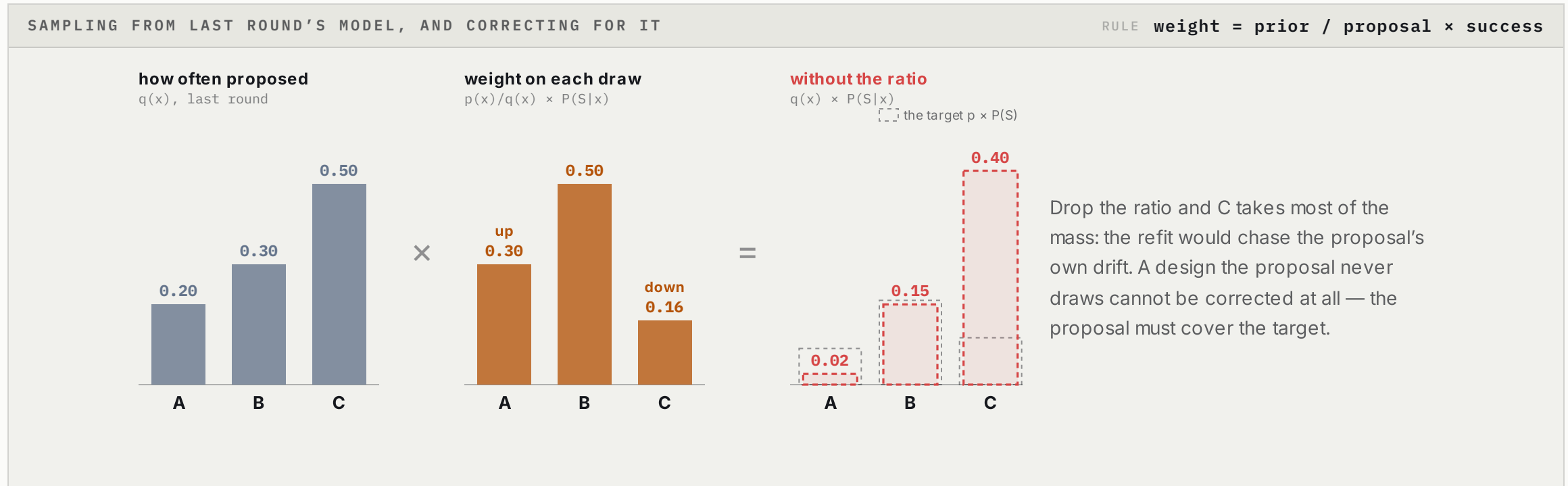
$$w_i = \frac{p(x_i \mid \theta^{(0)})}{q(x_i \mid \phi^{(t-1)})} P(S^{(t)} \mid x_i)$$

RAISE THE BAR GRADUALLY

- **(A)** each relaxed condition $S^{(t)}$ is met often enough under the current proposal;
- **(B)** $S^{(t)} \supset S^{(t+1)} \supset S$ — the bar only rises, and reaches the target.

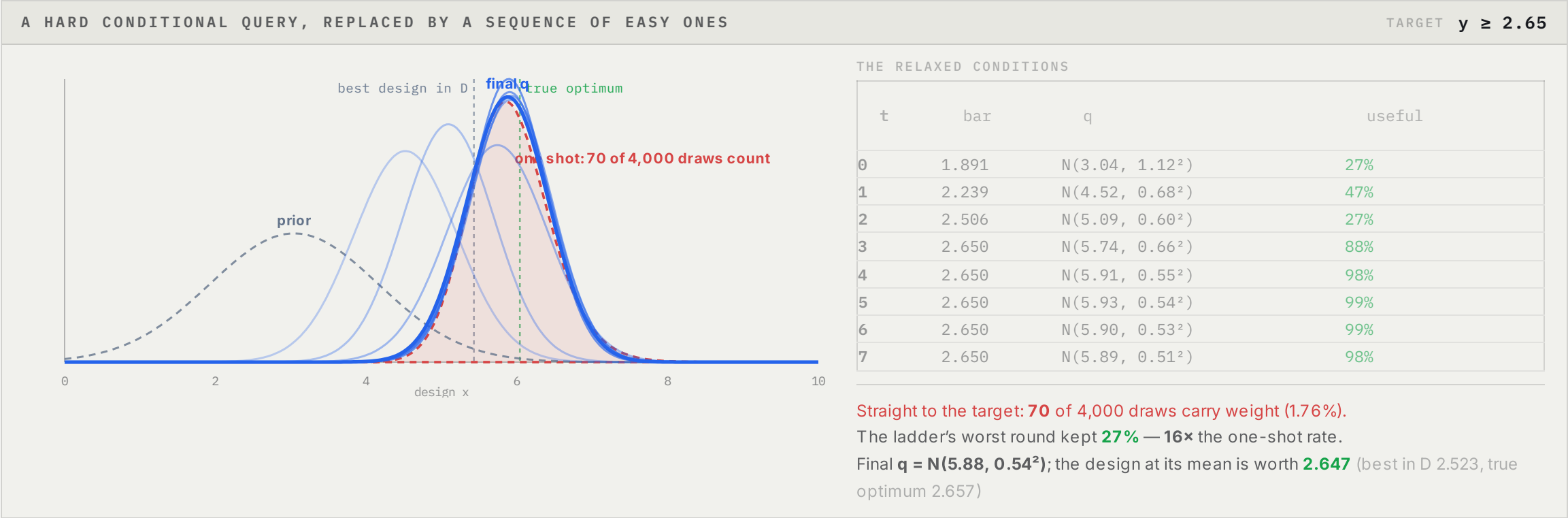
A hard conditional query, replaced by a sequence of easy ones. Unlike Lecture 3, the model is not fixed: the proposal and the condition both move.

Why the importance ratio belongs in the weight



Three designs, sampled from last round's model instead of the prior. Press → four times.

One shot, then the ladder



Press → to ask for the target in one shot, then to climb. Here q is a Gaussian refit by weighted maximum likelihood — **the pattern Lecture 4's CMA-ES also follows**: sample, weight the good candidates, refit the distribution.

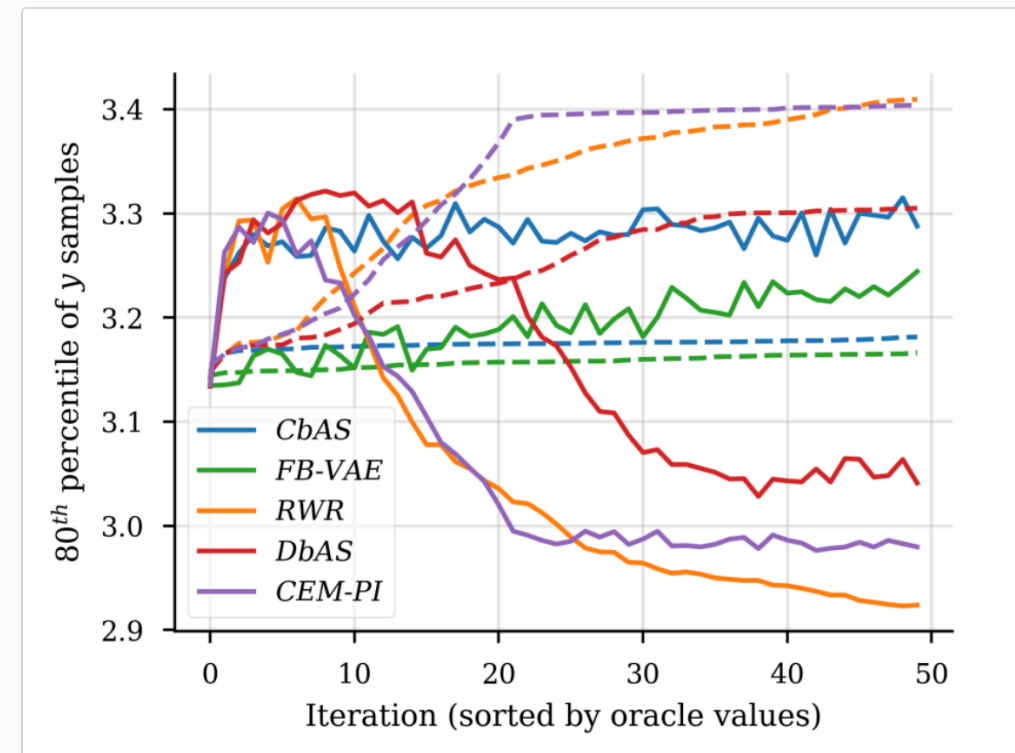
What CbAS does on a real design problem

protein fluorescence, 5,000 measured sequences · Brookes et al., 2019

Each round, **dashed**: the oracle's 80th percentile of the samples; **solid**: their true mean.

Methods that chase the oracle — RWR, DbAS, and CEM-PI, a distribution search like Lecture 4's — climb on the dashed line while the solid line **falls**: they have found where the oracle is wrong. CbAS keeps the two together.

Lecture 5's failure, now in a distribution: a generator refit to a surrogate's favourites drifts to where the surrogate is wrong, unless the prior holds it. The ratio p/q is that hold — a partial one.



80th percentile of oracle predictions (dashed) and mean ground truth of the same samples (solid). (Brookes et al., 2019, Fig. 2b)

Two more ways to move the distribution — GFlowNet and BootGen

GFLOWNET (BENGIO, JAIN, KORABLYOV, PRECUP & BENGIO · NEURIPS 2021)

Build a design step by step — an atom, a fragment, a residue — and train the steps so that a finished design is drawn **in proportion to its reward**, $P(x) \propto R(x)$: a flow-consistency loss, not a refit to weighted samples.

It needs **reward calls**: online if they query the real objective; offline, a proxy — Lecture 5's surrogate. Our lab's Genetic-guided GFlowNets (NeurIPS 2024) adds a genetic search under a small budget of real oracle calls — online.

BOOTGEN (KIM, BERTO, AHN & PARK · NEURIPS 2023 · OUR LAB)

Train $p_\theta(x \mid y)$ with weights by **rank** of y ; sample at a high $y^\dagger$, let a proxy label the samples, add the top K , retrain; pool several such generators.

G2 — it refits the training distribution toward its own best samples, as CbAS does — and **G3** — it is conditioned on y . The added samples are **proxy-labelled synthetic samples — no new performance measurements**.

Both shift probability toward good designs, by objectives different from CbAS's — and both rely on a proxy wherever the real objective is not called.

What moving the distribution buys, and what it costs

ASSUMPTION ADDED

The success model is right **where the refit goes**, and the prior gives the good designs some mass to begin with.

WHAT IT SOLVES

Many good candidates at once, from a model that stays tied to plausible designs.

WHAT IT LEAVES

The refit follows the success model's errors as far as the prior allows; a bar set too high has nothing to climb to.

Lecture 5's tools plug in here: build $P(S \mid x)$ from a **pessimistic** prediction $(\mu - \kappa\sigma)$ or a **conservative** surrogate, and the refit has less to chase.

Check — coordinates or distribution?

A latent-space search returns a code whose decoded molecule is chemically invalid; the property predictor rates it the best ever seen. Before any evaluation, what has been established?

- ☐ A The predictor is overfitted, and retraining it fixes both problems
- ☒ B A validity failure, and only that: the high prediction is not evidence of performance, and whether the predictor errs on valid designs is a separate question for evaluation
- ☐ C The design is excellent; only the decoder needs fixing to output it
- ☐ D Switching to CbAS would guarantee a valid molecule

Invalidity is observed directly: the decoder left the valid set — a problem of the coordinates, which JT-VAE addresses by construction. The predicted value is not a measurement: for an invalid design the property is not even defined, and for valid ones the predictor's reliability is Lecture 5's question, settled only by evaluating designs. CbAS keeps samples where the prior has mass, which helps, but a learned prior has tails: nothing here guarantees validity.

Session 1— what generation fixed, and what it left

MITIGATED

Plausibility and diversity: candidates come from a model of plausible designs. G1 searched its coordinates; G2 moved its distribution toward predicted success.

LEFT

- the performance signal still comes from a **surrogate**, exploited by the search (G1) or the refit (G2);
- G2 needed a bar γ , a success model and many refits.

Session 2 moves the signal closer to the generator: **ask for the performance directly** (G3), or **use it to steer the sampling process** (G4) — and a KL-regularised view connects several of them.

— SESSION 2 · ACT 4

Condition and sampler

Q4. Instead of refitting the generator, tell it what you want — or steer it at every step.

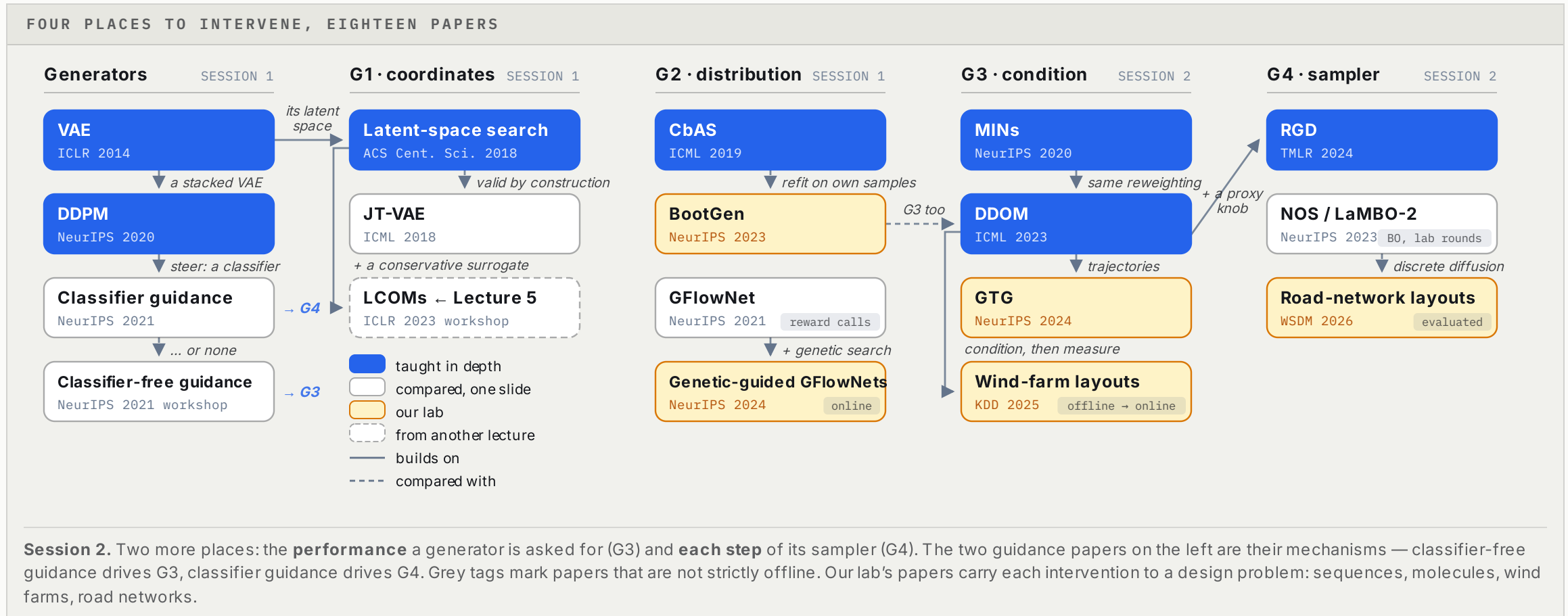
Session 2 — which distribution is this?



DISTRIBUTION	THE QUESTION IT ANSWERS	WHERE WE MET IT
$p_{\theta}(x)$	which designs to propose	Session 1: the generator
$p_{\theta}(x \mid y)$	which designs for a requested y	today: G3
$\hat{f}(x), P(S \mid x), \mu \pm \kappa\sigma$	how good a design is predicted to be	Lecture 5: the performance model
$p(z)$	how to draw a code	the VAE's latent prior
$p(w \mid D)$	which model weights are credible, given the data — a Bayesian posterior	Lecture 2
ensemble disagreement	how much independently trained models differ — a non-Bayesian uncertainty signal	Lecture 5 (Deep Ensembles)

The diversity of $p_{\theta}(x)$ is not uncertainty about performance, and **a requested y is not a measured y** . Today: the **condition** (G3), the **sampler** (G4), and a KL-regularised view of both.

The papers of Session 2



The two right-hand columns are today's. Click a paper for what it does, why we read it and how it connects.

Condition the encoder and the decoder — the inverse map is built

the conditional VAE — G3 in its simplest form

$$\max_{\theta, \phi} \mathbb{E}_{q_{\phi}(z|x, y)} [\log p_{\theta}(x | z, y)] - D_{\text{KL}}(q_{\phi}(z | x, y) || p_{\theta}(z | y))$$

Feed the condition y into both networks. Trained on street-number photographs, each row is the digit asked for and each column a style: the condition fixed, the rest of the freedom sampled.

And the source says the quiet part: **y can be the desired property, and x the corresponding design** — with the warning that a requested y is not a measured one.



Samples from a CVAE trained on SVHN: row = requested digit. (source deck, Part 1 p. 14)

G3 — ask the inverse map for a score

Model Inversion Networks · Kumar & Levine · NeurIPS 2020

TRAINING — WHICH y TO LEARN ON?

Accuracy at high y matters most, but the best few points alone are too noisy. So reweight:

$$w_i = \frac{p(y_i)}{p_D(y_i)}, \quad p(y) \propto \frac{N_y}{N_y + K} e^{y-y^*}$$

Up-weight high scores — **provided enough data share that score**.

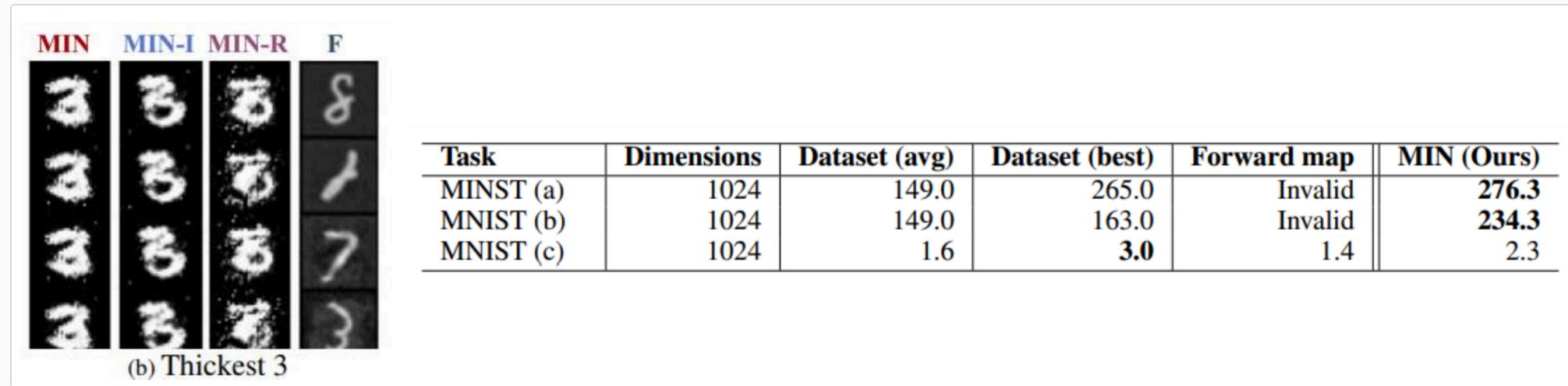
QUERYING — HOW FAR PAST THE DATA TO ASK?

An **optimisation over** (y, z) : the largest y that an independently trained forward model still confirms:

$$\begin{aligned} \tilde{y}, \tilde{z} &= \arg \max_{y, z} f_{\theta}(f_{\theta}^{-1}(y, z)) \\ \text{s.t. } &\|y - f_{\theta}(f_{\theta}^{-1}(y, z))\|^2 \leq \epsilon_1, \quad p(z) \geq \epsilon_2 \end{aligned}$$

Even the inverse route keeps a forward model — as a **consistency check on how far to ask**, Lecture 1's trust region in a new costume.

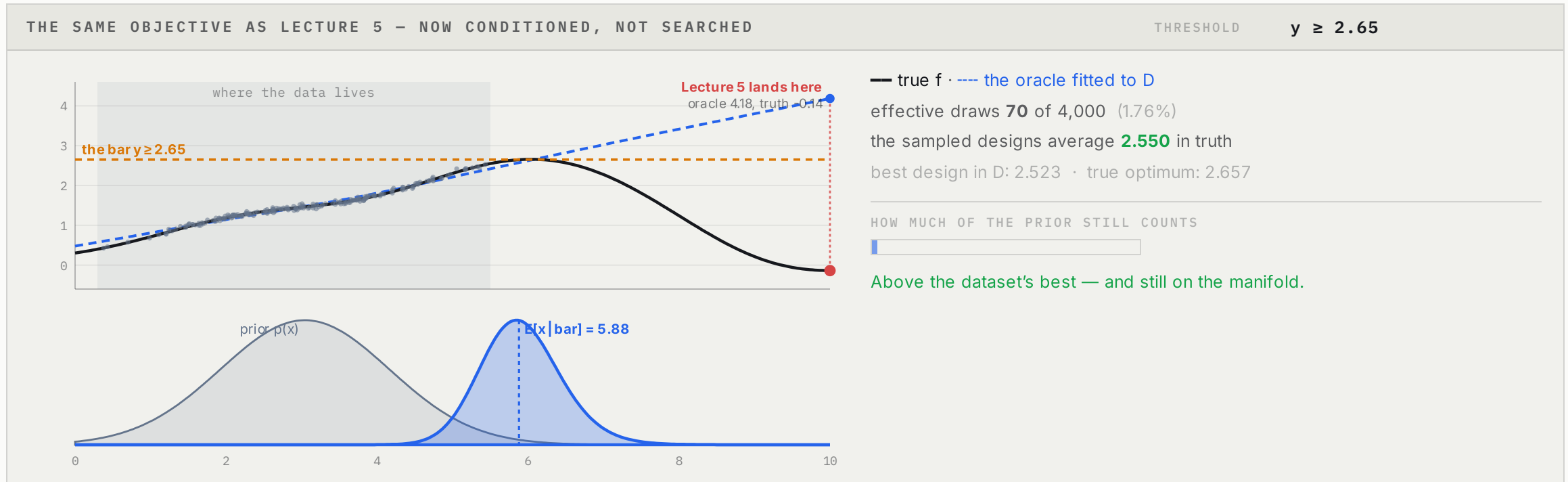
An inverse map returns designs; the forward map's arg-max did not



"The thickest 3 that is still recognisable": MIN and its ablations (left) against the forward map's arg-max (F), and the scores. (Kumar & Levine, 2020)

On two of these tasks the forward map's answer is **not a wrong design but no design at all** — an array of pixels that is not a digit. Lecture 5's failure, reported as a table entry: *Invalid*.

How far past the data can you ask?



The same two hundred designs and oracle, driven by the bar γ alone. Raise it and the samples move toward the good region, then **thin out and decay**. The red marker is Lecture 5's answer on the identical problem.

A requested score beyond the data is **extrapolation moved from x to y** . An exact conditional puts no mass where the prior has none; learned generators have tails, so over-asking still produces poor or invalid designs.

Classifier-free guidance — push a conditional model toward its condition

Ho & Salimans · NeurIPS 2021 workshop

Train **one** diffusion model with the condition and without it — drop y at random — and combine its two noise predictions at sampling:

$$\hat{\epsilon} = (1 + w) \epsilon_{\theta}(x_t, t, y) - w \epsilon_{\theta}(x_t, t)$$

WHAT IT AMOUNTS TO

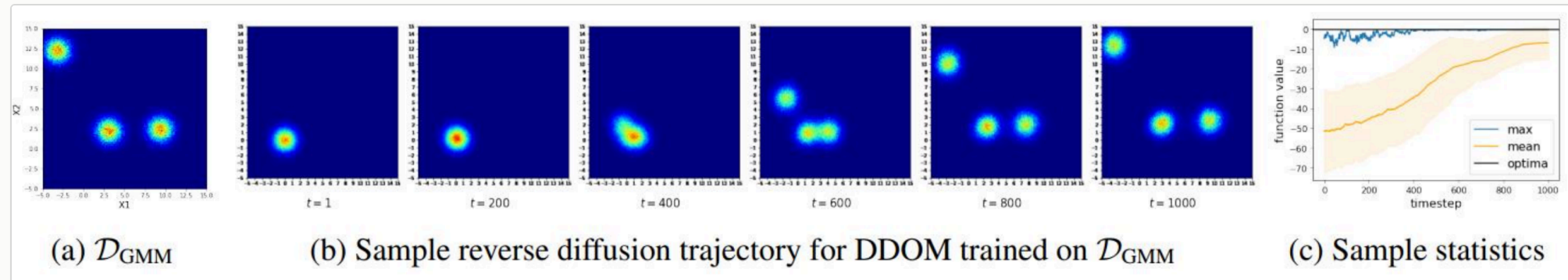
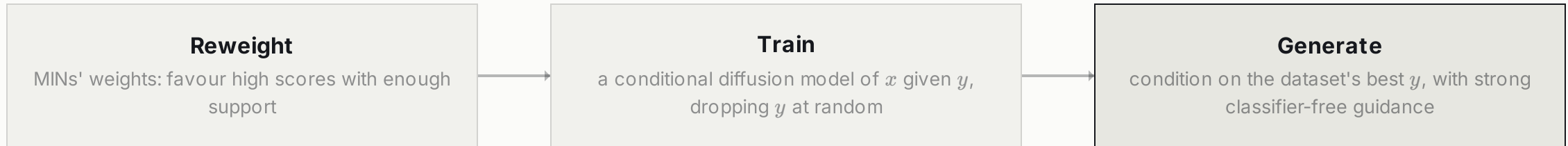
Adding $(1 + w) \nabla \log p_t(y \mid x_t)$ for an **implicit** classifier, $p(y \mid x) \propto p(x \mid y)/p(x)$. The strength w trades diversity for agreement with the condition.

WHY IT IS G3, NOT G4

There is **no separate model of performance**: the only signal is the labels the model was trained on. Its risk is G3's — asking for a y the data cannot vouch for.

DDOM — a diffusion model as the inverse map

Krishnamoorthy, Mashkaria & Grover · ICML 2023

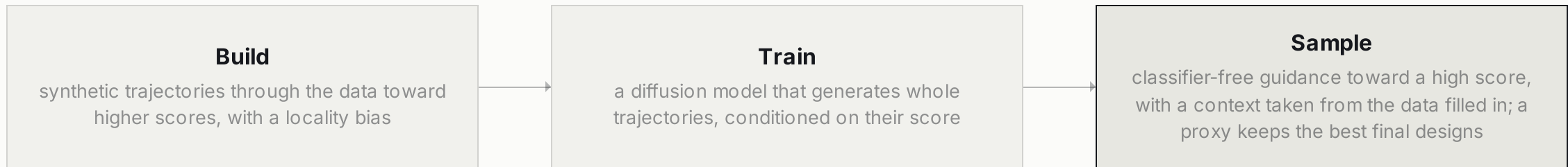


A three-mode toy problem: the reverse trajectory splits into all three modes; the samples' mean value climbs to the optimum. (Krishnamoorthy et al., 2023)

The one-to-many inverse, represented properly, and no surrogate anywhere. It asks only for the best label it has seen — whether samples exceed it is for the evaluation to say.

GTG — ask for an improvement, not a point

Yun, Yun, Lee & Park · NeurIPS 2024 · our lab



THE CONDITION

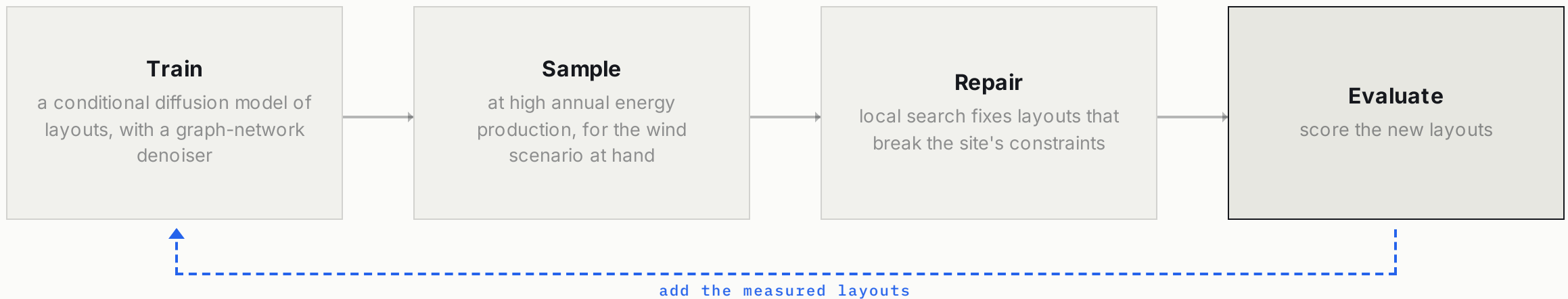
A trajectory carries a **direction** — from where the data are toward where they are not. Its steps are steps of improvement, not time and not denoising steps.

THE PROXY

It only **filters** the final candidates; it never steers the sampler. Still G3.

Wind-farm layouts — condition, measure, repeat

Shin, Yun, Yun, Woo, Lim & Park · KDD 2025 · our lab



G3 on a real design problem — with [Lecture 4's loop closed](#): every round is measured and added to the data. Not strictly offline: the condition asks, the evaluation answers.

What asking for a score buys, and what it costs

ASSUMPTION ADDED

The generator **generalises to the score you ask for** — usually the best in the data, or beyond it.

WHAT IT SOLVES

Many modes and diverse candidates from one model. MINs searches over (y, z) through the generator and a forward model; DDOM generates without a separate forward surrogate.

WHAT IT LEAVES

Asking past the data is extrapolation in y . MINs bounds it with a forward model and an optimisation over (y, z) ; DDOM asks only for the best label seen; the wind-farm loop measures every round.

A KL-regularised view of candidate distributions

The best average estimate $\hat{f}$ that stays close to the generator p — the KL averaged over the search:

$$q^* = \arg \max_{q \ll p} \mathbb{E}_q[\hat{f}(x)] - \frac{1}{\beta} D_{\text{KL}}(q \| p) \quad \implies \quad q^*(x) = \frac{p(x) e^{\beta \hat{f}(x)}}{Z_\beta} \quad (\beta > 0, Z_\beta < \infty)$$

Derivation. $\mathbb{E}_q[\hat{f}] - \frac{1}{\beta} D_{\text{KL}}(q \| p) = \frac{1}{\beta} \mathbb{E}_q \left[\log \frac{p e^{\beta \hat{f}}}{q} \right] = \frac{1}{\beta} [\log Z_\beta - D_{\text{KL}}(q \| q^*)]$, largest — uniquely — at $q = q^*$. ■

THE TWO ENDS OF β

$\beta \rightarrow 0$: q^* is the generator itself. As β grows, q^* concentrates — under suitable conditions — near the maximisers of $\hat{f}$ on the generator's support. That is a statement about the **target distribution**, not about what a search algorithm returns.

WHAT IT REGULARISES

Where we search, not what we believe: Lecture 5's COMs changed the predictor; this KL limits how far the candidate distribution moves from the generator.

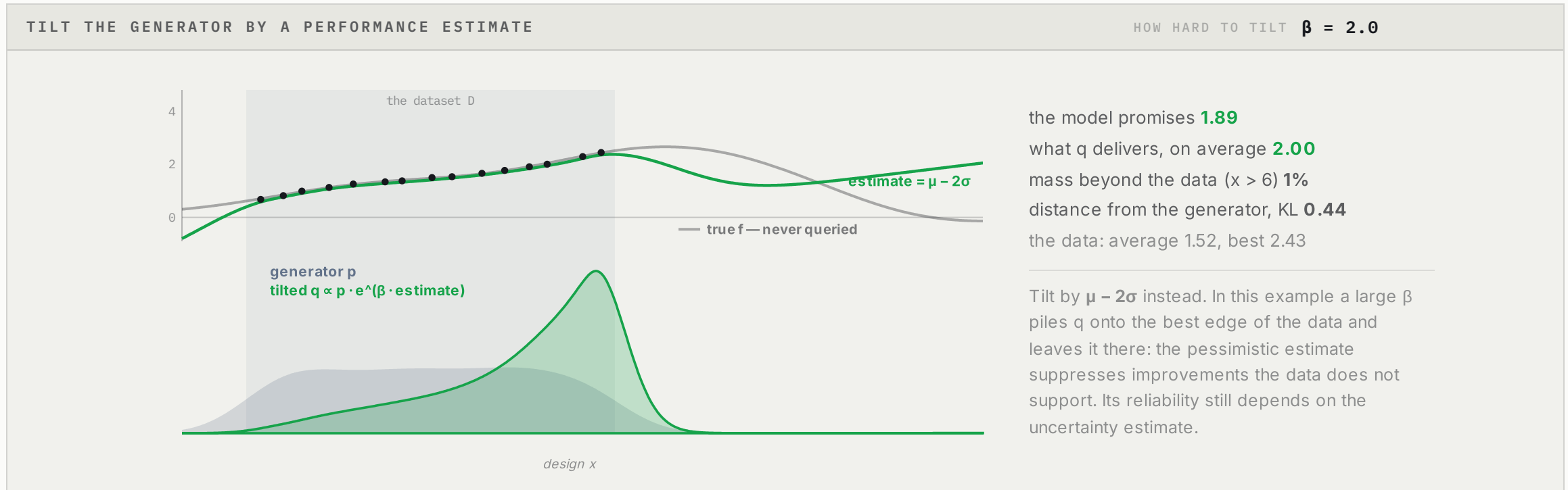
For this optimisation problem the exponential tilt is **the exact optimum**. Next: which methods it helps to read — and which it does not.

How the KL view relates to the methods

METHOD	RELATION TO $q^* \propto p e^{\beta \hat{f}}$
CbAS (G2)	its target is exactly q^* with $\hat{f} = \log P(S \mid x)$ and $\beta = 1$ — fitted with the other KL direction
guidance (G3, G4)	each noise level's score is that of a tilted p_t ; the full run need not sample the final tilted distribution, even with exact scores
DiBO (ICML 2025, our lab)	$p \cdot e^{\beta \text{UCB}}$, sampled by a fine-tuned diffusion model — online, Lecture 4
MINs, GFlowNet, BootGen	not this problem: a constrained query over (y, z) ; a reward-proportional sampler with no generator prior; rank weights and bootstrapping

A useful lens for some methods, not a common objective for all. And deriving q^* exactly does not mean a sampler draws from it exactly. Tilting by a conservative or pessimistic estimate combines this view with Lecture 5.

Too hard a tilt, and Lecture 5 returns



Lecture 5's fifteen designs and hidden f ; the generator is a smoothed copy of the data with a thin tail. Drag β ; press $\rightarrow$ to tilt by Lecture 5's pessimistic $\mu - 2\sigma$ instead.

The KL is **an average, not a leash**: it does not bound how far any one design moves. Where the generator has a thin tail and the estimate is wrong, a hard tilt moves the mass there.

G4 — use performance information to steer the sampler

classifier guidance · Dhariwal & Nichol · NeurIPS 2021

Classifier guidance — Bayes' rule for density scores (gradients of log-densities, not performance scores), with a classifier p_ϕ trained on noisy inputs and its term scaled by s :

$$\nabla_{x_t} \log p_t(x_t | y) = \nabla_{x_t} \log p_t(x_t) + \nabla_{x_t} \log p_\phi(y | x_t)$$

With a regression surrogate, define the candidate distribution at each noise level directly, $q_t(x) \propto p_t(x) e^{s\hat{f}_t(x)}$, so that

$$\nabla \log q_t = \underbrace{\nabla \log p_t}_{\text{the generator's pull}} + s \underbrace{\nabla \hat{f}_t}_{\text{the surrogate's push}}$$

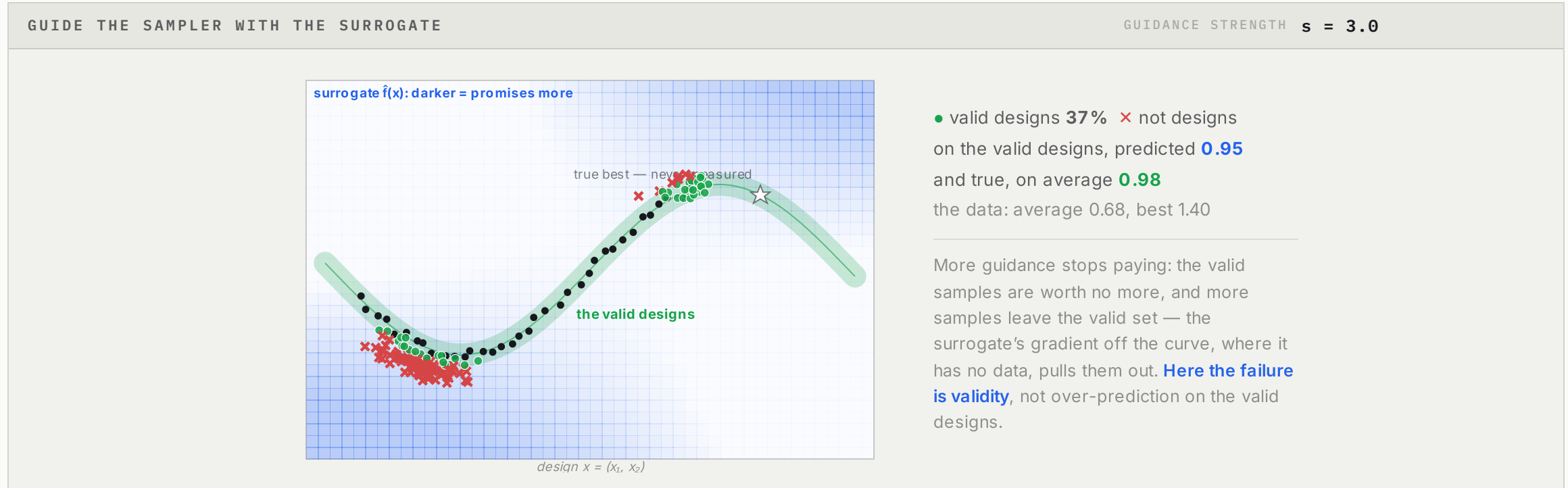
A score at each noise level — not a guarantee that the whole denoising run samples the final tilted distribution.

This **direct** form is **gradient ascent on the surrogate**, **interleaved with denoising** — Lecture 5's optimiser inside the sampler. RGD, later, is G4's other form: the proxy sets only the guidance strength.



Classifier guidance on one class, scale 1 (left) and 10 (right). (Dhariwal & Nichol, 2021)

Too much guidance leaves the valid set



The thin-curve problem of Session 1, now sampled with direct guidance. Drag s : on the valid samples prediction and truth stay close — **the failure here is validity**: the surrogate's gradient off the curve, where it has no data, pulls samples out.

RGD — give the proxy one knob

Robust Guided Diffusion · Chen, Beckham, Liu, Liu & Pal · TMLR 2024

PROXY-ENHANCED SAMPLING

Sample with **classifier-free** guidance — no proxy gradient on x — and let the proxy choose only the guidance **strength** ω , adjusted so the denoised sample scores higher.

DIFFUSION-BASED PROXY REFINEMENT

Refine the proxy toward the diffusion model's own $p(y | x)$, so that the proxy and the generator agree where the proxy would otherwise over-promise.

G4's second mechanism. Lecture 5 says the proxy is unreliable off the data; RGD **shrinks what it controls** — one scalar per step instead of every coordinate of x — and corrects it with the generator. That reduces the exposure; it does not make the proxy right.

Discrete designs — guided discrete diffusion

NOS / LAMBO-2 (GRUVER ET AL., NEURIPS 2023)

Protein sequences have no gradient in x , so NOS guides the denoiser's **continuous hidden states**. LaMBO-2 uses it inside a Bayesian-optimisation procedure with laboratory rounds — an application, not offline MBO.

ROAD-NETWORK LAYOUTS (YUN ET AL., WSDM 2026 · OUR LAB)

A city's road layout is a discrete structure; a **guided discrete diffusion** model generates layouts for a given traffic pattern.

G4 travels to sequences and graphs: the same intervention point, wherever there is a continuous place to put the gradient.

What steering the sampler buys, and what it costs

ASSUMPTION ADDED

The performance information is right where the sampler goes — a gradient at every noise level, or a proxy's ranking for the strength.

WHAT IT SOLVES

One trained generator, any objective at sampling time, and a dial for how hard to push.

WHAT IT LEAVES

Exploitation and invalid samples that grow with the strength. Limiting the proxy (RGD) or guiding by a pessimistic estimate (Lecture 5) can reduce the exposure; neither removes it, and both depend on their own estimates.

Check — which intervention point?

DDOM samples a conditional diffusion model with classifier-free guidance at the dataset's best score. RGD lets a proxy choose the guidance strength. Which statement is right?

- ☐ A Both are G4, because both use guidance
- ☒ B DDOM is G3 — no surrogate, only its training labels; RGD adds a G4 element, restricted to one scalar
- ☐ C RGD is more exposed to surrogate exploitation than plain classifier guidance
- ☐ D DDOM extrapolates safely because a diffusion model has no tails

Classifier-free guidance strengthens a condition the model learned from labels — a G3 intervention. G4 means a separate model of performance steers the sampler; RGD lets the proxy set only the strength ω , far less than a gradient on every coordinate of x . DDOM's samples still need evaluation: strong guidance toward the best label it has seen can still leave the data.

— CLOSING

Closing

Four places for a performance signal to get in — and Lecture 5's tools in each.

Four interventions, side by side

	G1 • COORDINATES	G2 • DISTRIBUTION	G3 • CONDITION	G4 • SAMPLER
in depth	latent search	CbAS	MINs, DDOM	classifier guidance, RGD
the signal enters	the search over codes	the refit's weights	the requested y	the sampling process
plausibility from	the decoder	the prior ratio	the generator	the generator's pull
what it leaves	invalid decodes; an exploitable predictor	follows the success model's errors	extrapolation in y	invalid samples and exploitation that grow with strength
Lecture 5's tool there	conservative latent predictor (LCOMs)	pessimistic $P(S \mid x)$	a forward check (MINs), or measure	a limited proxy (RGD), or pessimistic guidance
our lab	—	BootGen • Genetic GFN	GTG • wind farms	road networks

Not rivals: BootGen is G2 and G3, RGD is G3 with a G4 knob, and [the KL view](#) connects several of them — a lens, not their common objective.

Performance, validity, reliability — the two lectures together

Lecture 5 closed on these three questions; here is where each lecture stands.

PERFORMANCE

Is the design good? — *the objective, in both lectures*

VALIDITY

Is it possible at all? — **this lecture**:
plausibility from the generator; feasibility
from a decoder that enforces it, or a
check

RELIABILITY

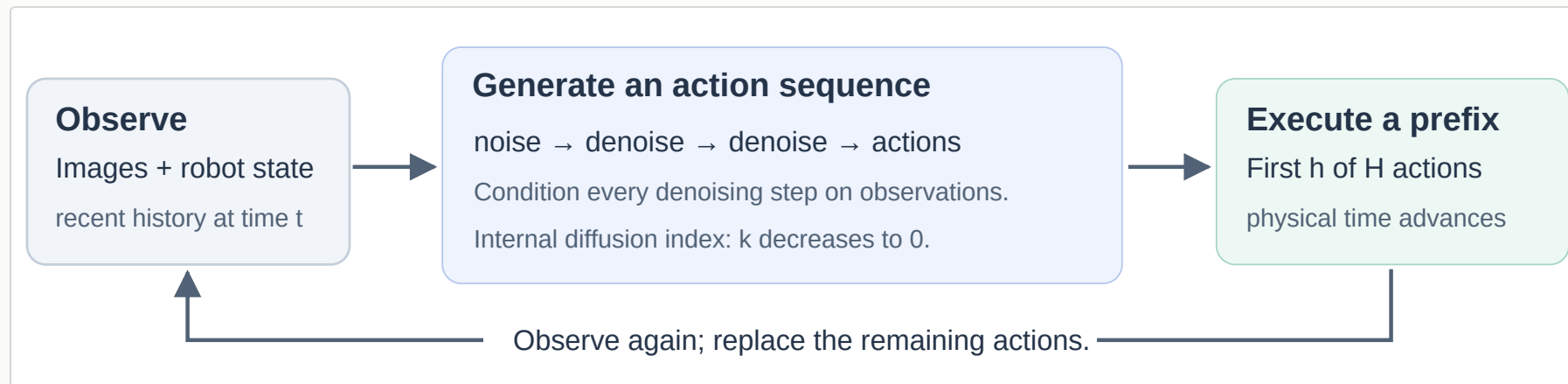
Can the prediction be trusted? — **Lecture 5**: conservatism, uncertainty, knowledge
— wherever a surrogate enters

Generate to stay plausible, score with care, measure to settle: a generator proposes, a conservative or uncertainty-aware model ranks, and only an evaluation decides. Two settings: BootGen selects with a proxy and is evaluated afterwards; the wind-farm loop adds evaluations to the data while it optimises.

Diffusion Policy — generate a sequence of actions

Chi et al. · RSS 2023 · from design generation to robot behaviour

Replace a static design with an **action sequence**, conditioned on the robot's recent observations; train on demonstrations with the diffusion noise-prediction loss. Execute a few actions, observe again, generate anew.



The observation-conditioned generate-and-execute loop. Denoising steps happen inside one decision; they are not physical time steps.

A sequence keeps different demonstrated ways around an obstacle as different **modes**. This is **imitation**, not reward-based RL: the objective is the demonstrator's actions. Executing a little and re-planning anticipates **feedback and MPC** (Lectures 9–11). [Paper](#)

Where we are — a performance model and a generator

	LECTURE 5	LECTURE 6
the model	a performance model $\hat{f}(x)$	a generator $p(x)$ or $p(x y)$, plus a performance model
the decision	search $\hat{f}$ — carefully	search, refit, condition or steer the generator
what it secures	grounds for trusting a prediction	plausible, diverse candidates

A MODEL OF HOW GOOD — VALUE

Lectures 7–8 learn how good each decision is, $Q(s, a)$, and act by maximising it — and meet Lecture 5's hazard: the policy exploits Q where Q is wrong.

A MODEL OF WHAT TO PROPOSE — POLICY

Lecture 10 learns a distribution over actions, $\pi_\theta(a | s)$, and samples it. A comparison worth keeping, not the same algorithm: the two are usually trained together.

So far: static design choices, experiments that buy information, and inference over time (Lectures 3–4). From Lecture 7 on, the current action changes the next state and the future reward — and the decision is optimised over that sequence.

A generator decides where to look; a performance
model decides what to trust.

The signal can enter at four places — and wherever a surrogate enters, Lecture 5 comes with it. A
measurement still has the last word.

Questions?

Two things to carry out of here. **A generator buys plausibility, not evidence** — the performance signal still has to be trusted, as in Lecture 5. And **ask where the signal enters** — coordinates, distribution, condition or sampler — because that is where it can be exploited.

— APPENDIX

Backup slides

Complete derivations and reported numbers, kept out of the narrative.

Reading guide — Session 1 papers: generative models and G1

PAPER	WHY WE READ IT	HOW IT CONNECTS
VAE · Kingma & Welling, ICLR 2014 — <i>in depth</i>	a generator with an explicit latent space	the coordinates G1 searches
DDPM · Ho, Jain & Abbeel, NeurIPS 2020 — <i>in depth</i>	the generator behind most current design methods	steered by the two guidance papers
Classifier guidance · Dhariwal & Nichol, NeurIPS 2021	a separate model's gradient steers the sampler	the mechanism of G4
Classifier-free guidance · Ho & Salimans, NeurIPS 2021 workshop	steering without a separate model	the sampler of G3
Latent-space search · Gómez-Bombarelli et al., ACS Cent. Sci. 2018 — <i>in depth</i>	G1: search in learned coordinates	also Lecture 4
JT-VAE · Jin, Barzilay & Jaakkola, ICML 2018	validity by construction	fixes G1's decoder
LCOMs · Qi et al., ICLR 2023 workshop	a conservative surrogate on latent codes	Lecture 5 inside G1

Reading guide, continued — Session 1 papers: G2

PAPER	WHY WE READ IT	HOW IT CONNECTS
CbAS · Brookes, Park & Listgarten, ICML 2019 — <i>in depth</i>	G2: refit the generator toward success	the tilt with $\beta = 1$
GFlowNet · Bengio et al., NeurIPS 2021	sample in proportion to reward	needs reward calls or a proxy
BootGen · Kim, Berto, Ahn & Park, NeurIPS 2023 — <i>our lab</i>	refit on its own proxy-scored samples	G2 and G3
Genetic-guided GFlowNets · Kim, Kim, Choi & Park, NeurIPS 2024 — <i>our lab</i>	GFlowNet training under a small oracle budget	online, not offline

Reading guide, continued — Session 2 papers: condition and sampler

PAPER	WHY WE READ IT	HOW IT CONNECTS
MINs · Kumar & Levine, NeurIPS 2020 — <i>in depth</i>	G3: an inverse map asked for a score	keeps a forward model as a check
DDOM · Krishnamoorthy, Mashkaria & Grover, ICML 2023 — <i>in depth</i>	G3 with diffusion: every mode kept	MINs' weights, no proxy
GTG · Yun, Yun, Lee & Park, NeurIPS 2024 — <i>our lab</i>	condition a trajectory, not a point	the proxy only filters
Wind-farm layouts · Shin et al., KDD 2025 — <i>our lab</i>	G3 on a real design problem	measured every round
DiBO · Yun, Om, Lee, Yun & Park, ICML 2025 — <i>our lab</i>	the tilted target $p e^{\beta UCB}$, sampled by a fine-tuned diffusion model	online; Lecture 4; a GFlowNet-family loss
RGD · Chen et al., TMLR 2024 — <i>in depth</i>	G4 with the proxy limited to one knob	classifier-free guidance + a proxy
NOS / LaMBO-2 · Gruver et al., NeurIPS 2023	guidance for discrete sequences	a BO application
Road-network layouts · Yun et al., WSDM 2026 — <i>our lab</i>	guided discrete diffusion for city design	G4 on graphs

Backup 1—the ELBO, derived, and the reparameterisation trick

We want $\log p_\theta(x)$, which integrates over z . Introduce $q_\phi(z | x)$:

$$\log p_\theta(x) = \underbrace{\mathbb{E}_{q_\phi} \left[\log \frac{p_\theta(x, z)}{q_\phi(z | x)} \right]}_{\text{ELBO } \mathcal{L}_{\theta, \phi}(x)} + \underbrace{\text{KL}(q_\phi(z | x) \| p_\theta(z | x))}_{\geq 0}$$

The KL is non-negative, so the ELBO is a lower bound — **not generally the exact evidence**; it is tight only when q_ϕ equals the true posterior. Expanding $p_\theta(x, z) = p_\theta(x | z)p(z)$ gives the working form

$$\mathcal{L}_{\theta, \phi}(x) = \mathbb{E}_{q_\phi(z | x)} [\log p_\theta(x | z)] - \text{KL}(q_\phi(z | x) \| p(z)) \quad \blacksquare$$

Reparameterisation. Write $z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$: the randomness moves off the parameters and the gradient flows.

Why the KL term is the samplability term. With a linear-Gaussian decoder the β -optimal encoder is $\mu = (W^\top W + \beta\sigma^2 I)^{-1} W^\top (x - b)$ and $\Sigma = \beta\sigma^2 (W^\top W + \beta\sigma^2 I)^{-1}$. Along an eigendirection of $W^\top W$ with eigenvalue w the aggregate posterior variance is $\text{Var}(\mu) + \Sigma = (w^2 + \sigma^2 w + \beta\sigma^2 w + \beta^2\sigma^4)/(w + \beta\sigma^2)^2$, whose difference from 1 is $\sigma^2 w(1 - \beta)/(w + \beta\sigma^2)^2$ — zero at $\beta = 1$. The *Turning the dial* widget is that formula, plotted.

Backup — one VAE calculation

For one latent coordinate, $q(z \mid x) = \mathcal{N}(1, 0.5^2)$ and $p(z) = \mathcal{N}(0, 1)$:

$$\text{KL}(q\|p) = \frac{1}{2} (\mu^2 + \sigma^2 - 1 - \log \sigma^2) = \frac{1}{2} (1 + 0.25 - 1 - \log 0.25) \approx 0.818.$$

With an expected reconstruction log-likelihood of -2 , the ELBO is $-2 - 0.818 = -\mathbf{2.818}$. One reparameterised draw with $\epsilon = -1$: $z = 1 + 0.5(-1) = 0.5$.

Reconstruction rewards explaining the input; KL charges for moving the code distribution away from the prior.

Backup 2 — the four families of generative models

FAMILY	TRAINING	LATENT	TRADE-OFF
VAE	maximise the ELBO	explicit, $\sim \mathcal{N}(0, I)$	stable; samples can be blurry
GAN	adversarial, generator vs. discriminator	implicit	sharp; unstable, mode collapse
Normalising flow	exact maximum likelihood	invertible, same dimension	exact density; architectural constraints
Diffusion	denoising score matching	fixed noise schedule	high quality; slow sampling

Any of the four can serve as the prior $p(x)$ of a generative design pipeline. The training objective of an inverse map can likewise be any divergence: $L_p(D) = \mathbb{E}_{y \sim p(y)} [\text{Div}(p_D(x | y), p_{f_\theta^{-1}}(x | y))]$ — KL gives maximum likelihood, JS a GAN objective (*Part 2 p. 21*).

Backup 2b — diffusion as a hierarchical VAE

Forward and reverse. $q(x_{1:T} \mid x_0) = \prod_{t=1}^T q(x_t \mid x_{t-1})$ and $p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1} \mid x_t)$, and the same ELBO argument gives

$$\mathcal{L}(\theta) = \mathbb{E}_q \left[\underbrace{\text{KL}(q(x_T \mid x_0) \parallel p_\theta(x_T))}_{L_T} + \sum_{t=2}^T \underbrace{\text{KL}(q(x_{t-1} \mid x_t, x_0) \parallel p_\theta(x_{t-1} \mid x_t))}_{L_{t-1}} - \underbrace{\log p_\theta(x_0 \mid x_1)}_{-L_0} \right]$$

Why Gaussian. The kernel is chosen so that $q(x_t \mid x_0)$ and $q(x_{t-1} \mid x_t, x_0)$ have closed forms and $q(x_T \mid x_0)$ forgets x_0 . With $q(x_t \mid x_{t-1}) = \mathcal{N}(\sqrt{1 - \beta_t}x_{t-1}, \beta_t I)$, $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i \leq t} \alpha_i$:

$$q(x_t \mid x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)I), \quad q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$$

$$\tilde{\mu}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}x_t, \quad \tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t, \quad \bar{\alpha}_t \rightarrow 0 \Rightarrow q_\infty = \mathcal{N}(0, I)$$

Backup 3 — the DDPM objective collapses to one squared error

Both $q(x_{t-1} \mid x_t, x_0)$ and $p_\theta(x_{t-1} \mid x_t)$ are Gaussian, so L_{t-1} is an analytic KL:

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \left\| \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) \right\|^2 \right] + C$$

Substituting $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$ suggests predicting the *noise*:

$$\mu_\theta \stackrel{\text{def}}{=} \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) \implies L_{t-1} = \mathbb{E}_{x_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \left\| \epsilon - \epsilon_\theta(x_t, t) \right\|^2 \right]$$

Dropping the weight — which the authors found works better — gives $\mathcal{L}_{\text{simple}} = \mathbb{E}_{t, x_0, \epsilon} \left[\left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t) \right\|^2 \right]$, a reweighted objective rather than the exact bound.

One training step, in numbers. $x_0 = 2$, $\bar{\alpha}_t = 0.64$, $\epsilon = 1$ give $x_t = 0.8(2) + 0.6(1) = 2.2$; predicting $\hat{\epsilon} = 0.5$ costs $(1 - 0.5)^2 = 0.25$. (Ho, Jain & Abbeel, 2020)

Backup 4 — score matching, Langevin dynamics, and the SDE that unifies them

The estimator. Fit $s_\theta(x) \approx \nabla_x \log p(x)$ by the Fisher divergence, computable as $\mathbb{E}_p[\text{tr}(\nabla_x s_\theta) + \frac{1}{2} \|s_\theta\|^2]$ (Hyvärinen, 2005) or in the denoising form $\frac{1}{2} \mathbb{E}_{q_\sigma(\tilde{x}|x)p(x)} [\|\nabla_{\tilde{x}} \log q_\sigma(\tilde{x} | x) - s_\theta(\tilde{x})\|^2]$ (Vincent, 2011). For an energy model the partition function drops out of the score: $\nabla_x \log p_\theta(x) = -\nabla_x \varepsilon_\theta(x)$.

The blind spot. The Fisher divergence weights the error by the data density, $\mathbb{E}_{p(x)} \|\nabla_x \log p(x) - s_\theta(x)\|^2 = \int p(x) \|\cdot\|^2 dx$, so the score is learned badly where data is thin — a learned object is unconstrained where there is no evidence, as Lecture 5's surrogate was. **Its cure.** Train on data perturbed at L noise scales, $p_{\sigma_i}(x) = \int p(y) \mathcal{N}(x; y, \sigma_i^2 I) dy$, with a noise-conditional model $s_\theta(x, i)$ and loss $\sum_i \sigma_i^2 \mathbb{E}_{p_{\sigma_i}} \|\nabla \log p_{\sigma_i} - s_\theta(x, i)\|^2$; sample by **annealed Langevin dynamics**, $\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2} \nabla_x \log p(\tilde{x}_{t-1}) + \sqrt{\epsilon} z_t$, for $i = L, \dots, 1$.

The continuous limit. The perturbation becomes $dx = f(x, t)dt + g(t)dw$, whose reverse is

$$dx = [f(x, t) - g^2(t) \nabla_x \log p_t(x)] dt + g(t) d\bar{w}$$

NCSN is the variance-exploding SDE; DDPM the variance-preserving one, $dx = -\frac{1}{2} \beta(t)x dt + \sqrt{\beta(t)} dw$.

Backup — guidance, with and without a classifier

Train $\epsilon_\theta(x_t, t, y)$ and, by dropping y at random, $\epsilon_\theta(x_t, t)$. Since $\epsilon_\theta \approx -\sqrt{1 - \bar{\alpha}_t} \nabla \log p_t$, the guided prediction $\hat{e} = (1 + w)\epsilon_\theta(x_t, t, y) - w\epsilon_\theta(x_t, t)$ corresponds to the score

$$(1 + w)\nabla \log p_t(x_t \mid y) - w\nabla \log p_t(x_t) = \nabla \log p_t(x_t) + (1 + w)\nabla \log p_t(y \mid x_t),$$

because $p(y \mid x) \propto p(x \mid y)/p(x)$. Classifier guidance adds $s \nabla \log p_\phi(y \mid x_t)$ with a separately trained p_ϕ instead. In both, each noise level's guided score is the score of a tilted distribution, $p_t(x) p_t(y \mid x)^{\text{strength}}$ — but the full sampler need not draw from the final tilted distribution, even with exact scores: Bradley & Nakkiran (2024, "Classifier-Free Guidance is a Predictor-Corrector") show that CFG with DDPM or DDIM does not generate the γ -powered distribution.

Backup — the inverse map, as the source motivates it

$$f_{\theta} : \mathcal{X} \rightarrow \mathcal{Y} \quad \implies \quad f_{\theta}^{-1} : \mathcal{Y} \rightarrow \mathcal{X}$$

The forward model answers "*given x , what is y ?*". We wanted "*given the y I need, which x delivers it?*" — a map whose **input is one number**.

WHAT THE REVERSAL FIXES

The inverse map is not asked to search $\mathcal{X}$; it is asked to *emit* a point of it.

WHAT IT DOES NOT FIX

The **output** is still high-dimensional and still confined to a thin valid set. Reversing the arrow moves the difficulty; it does not remove it.

Backup 5 — MINs and CbAS, in full

MINS — TRAIN, THEN QUERY

Train. $L_p(D) = \mathbb{E}_{y \sim p(y)} [\text{Div}(p_D(x | y), p_{f_\theta^{-1}}(x | y))]$,
reweighted by $w_i = p(y)/p_D(y)$, $p(y) \propto \frac{N_y}{N_y + K} e^{y - y^*}$.

The weight comes from a bias–variance bound with three terms:

variance from thin data $\mathbb{E}_y[1/N_y]$, *distribution shift*
 $d_2(p||p_D)/|D|$, and *bias from the target* $D_{\text{TV}}(p^*, p)^2$.

Query. $x = f_\theta^{-1}(z, y_{\text{target}})$, with y_{target} from Approx-Infer.

Ablation on CIFAR-10 (uniform): MIN 77.12, without Approx-Infer 76.31, without reweighting 74.87 — **both components matter**.

The subtlety both share. Querying above the observed range is extrapolation — Lecture 5's hazard, moved from x into y . Practical systems temper y_{target} and re-rank samples with a (conservative) forward model: **generate to stay valid, score to select**.

CBAS — THE ADAPTIVE LADDER

```

 $\theta^0 \leftarrow$  fit the prior  $p(x|\theta^0)$  to D
 $q \leftarrow p(\cdot|\theta^0)$ 
repeat:
   $x_1..x_M \sim q$   $\leftarrow$  propose
   $\gamma \leftarrow \min(\text{target}, Q\text{-th pct of}$   

     $\hat{f}(x_1..x_M))$   $\leftarrow$  relax
   $w_i \leftarrow p(x_i|\theta^0)/q(x_i) \cdot P(y \geq \gamma | x_i)$ 
   $q \leftarrow$  weighted MLE on  $\{x_i, w_i\}$   $\leftarrow$  refit

```

The ratio uses the *previous* iterate as the proposal; finite variance additionally needs the proposal to cover the target. γ rises only as fast as the model can follow — condition (B).

What MINs does on real design problems

MINs, three task families · note the forward-map column

TASK	DIMENSION	DATASET AVG	DATASET BEST	FORWARD MAP	MIN
MNIST, thickest recognisable "3"	1,024	149.0	265.0	<i>Invalid</i>	276.3
MNIST, variant (b)	1,024	149.0	163.0	<i>Invalid</i>	234.3
Faces, youngest (≥ 15)	12,288	38.7	−15.0	<i>Invalid</i>	−12.2
Faces, youngest (≥ 25)	12,288	41.5	−25.0	<i>Invalid</i>	−23.9
HopperController reward	3,843	442.9	1915.5	93.1	1960.1
Pendulum reward	1,537	14.7	344.5	3.4	1000.0

The forward map is Lecture 5's naive route: on four of six tasks its answer is **not a design at all**; on Hopper it scores 93 where the dataset already held 1915. MIN beats the dataset's best on five of six. (*Kumar & Levine, 2020*)

DDOM and BootGen — the reported numbers

as the source deck reports them

DDOM (ICML 2023)

Reweighting ablation (normalised): D'Kitty 0.926 $\rightarrow$ 0.930, Ant 0.888 $\rightarrow$ 0.960, TFBind8 0.957 $\rightarrow$ 0.971 — improvements are task dependent.

Six Design-Bench tasks, budget 256: best mean rank 2.8, against COMs' 3.7 and gradient ascent's 3.5.

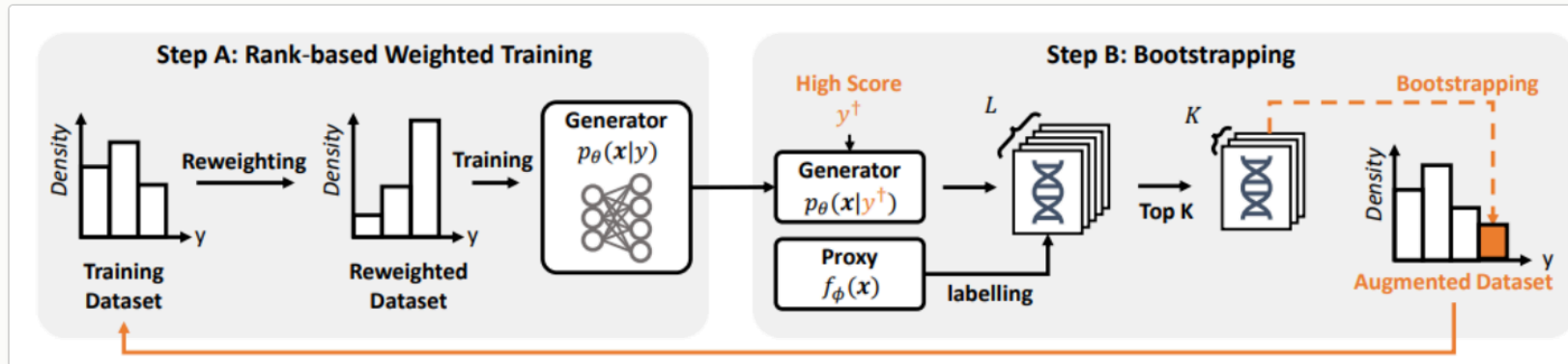
BOOTGEN (NEURIPS 2023, OUR LAB)

Six biological-sequence tasks, 100th percentile: average 0.895 against the dataset's 0.365 and the best baseline's 0.792 — with the highest **diversity and novelty** of the methods compared.

Benchmark results on the papers' tasks, not guarantees for a new dataset: every returned design still needs evaluation.

BootGen — generated labels do not become new measurements

rank weighting and iterative augmentation



Step A, rank-weighted training of a score-conditioned generator; Step B, bootstrapping: sample at a high score, label with a proxy, keep the top K , retrain. (Kim et al., 2023, Fig. 2.1)

Weights $\propto [k|D| + \text{rank}(y, D)]^{-1}$, normalised: higher-ranked designs weigh more, and the scale of y does not matter.

The training pool grows without new oracle evaluations: it gains **proxy-labelled synthetic samples, not new performance measurements**, so proxy errors can be reinforced as the loop repeats. This is pseudo-labelling and augmentation — not resampling the data to build a bootstrap ensemble.