The Steering Budget: Examples beat Knobs

Anonymous authors
Paper under double-blind review

Abstract

Generative models are steered by turning knobs — prompts, guidance scales — but every knob saturates: push it far enough and it stops moving the property you care about. This ceiling is a budget fixed by the training data before the model exists — no knob can push past it. We introduce *showing*, which reaches any value the whole budget allows, not just what a knob can reach, by partitioning outputs into easy-to-compute bins, auditing each, and combining bins in the proportions needed.

This audit makes the split exact: a target property's variance divides into a within-bin part (telling's reach) and a between-bin part (showing's reach), both computable from data alone, before a model exists. A second, model-facing diagnostic then confirms this split carries over to a trained model: on crystals, it correctly forecasts the carry-over order of three properties before the model was ever audited. Across two unrelated domains — image and crystal-structure generation — showing out-reaches the strongest knob baselines we could build by 4.8–26$\times$ on crystal properties and 3$\times$ on image targets, with no fine-tuning: only a different choice of examples. The same two numbers also say, in advance, when telling remains the better choice. Given showing wins, a property's curvature determines the shape of the fix: concentrate on one bin for an average goal, spread across bins for coverage. Showing further brings expressiveness a knob cannot: because the specification lives in the examples, it can steer toward a target recognized but never named.

1 Introduction

You have a generative model and want it to lean a certain way — images with the feel of nature photography, crystals with a wider band gap. You reach for the controls: a prompt, a guidance scale, a conditioning token. Past a point they stop moving the property — you push harder and the output barely changes. We show this ceiling is not a shortcoming of the model but a **budget**, fixed by the training data before the model is trained, and that a different move — *showing* the model examples instead of turning a knob — reaches a part of the budget the knob structurally cannot.

Fixing an input parameter before generation — an image's class, a crystal's element-set — picks one bin to generate into. *Turning a knob* changes some other input parameter, continuous or discrete, hoping to bias the target property, without itself picking a bin. Both are *telling*: whichever bins are in play — one, if an input was fixed, or all of them in their natural mix, if not — telling can only reweight the outputs *within* them, never move weight *between* them. This is what limits telling's reach, regardless of which or how many bins are in play.

Showing — handing over a set of examples and asking for more like them — works differently: it reweights the mix *across* bins instead of within them, reaching values no single bin contains.

Showing does this in two moves. Not every target property has an input parameter that names it directly — there is no *nature* class to name directly, only *forest* as a proxy — so rather than fixing an input parameter before generation, showing first bins the *outputs* themselves, already generated, by some easy-to-compute property. It then *audits* each bin: scoring a modest sample reveals that bin's share of the outputs and its typical target-property value. Together, these per-bin numbers expose the full range the target property can take, and let you aim for any value in that range by choosing which bins to draw from — not just whatever range one bin happens to contain.

Our central result makes this quantitative. A generative model produces outputs D (an image, a crystal structure). A cheap key $P = \pi(D)$ sorts each output into a bin — an image's class, say; for

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

a target property $L = \psi(D)$ — an image's aesthetic score, a crystal's band gap — the law of total variance splits $\mathrm{Var}(L)$ into a within-bin budget T and a between-bin budget E. Telling reweights the outputs *within* whichever bins are in play and reaches $\sqrt{T}$; showing hands over a set of examples, which reweights the mix *across* bins, and reaches $\sqrt{\chi^2 E}$. Both budgets are fixed by the data before the model exists, so one can predict, in advance, which targets a knob cannot reach — and, where a knob falls short, which *shape* of exemplar recipe closes the gap.

Showing additionally adds something a knob cannot: **expressiveness**. Because the specification lives in the examples, you can steer toward a target you can only point to — the curator who keeps ten images of a hundred, the materials scientist who flags a promising structure, for reasons they cannot fully state (Polanyi 1966). A knob demands the target be named first; showing does not.

Contributions.

1. **A computable steering budget.** How far any property can be steered is fixed by the training data alone, before the model exists, and splits exactly into two reaches: knobs are capped at a *telling* reach $\sqrt{\chi^2_{\text{tell}} T}$ (about $\sqrt{T}$ in practice), typically far short of the *showing* reach $\sqrt{\chi^2 E}$ the same data permits — a gap a saturated knob gives no hint of. Law of total variance for the split, Cauchy–Schwarz for the reaches. §3.
2. **Showing reaches any point the budget allows, measured.** Combining bins in different proportions — concentrating on one for an average goal, spreading across several for coverage — reaches any value in the budget, not just one bin's average. Measured head to head against the strongest knob each domain allows (best-of-twelve; a learned soft-prompt optimised on the scorer), it moves the batch mean 4.8–26$\times$ further on crystal properties and ~3$\times$ on high-share image targets, with a data-side crossover $E/(E+T)$ predicting *this* in advance — including the occasions (low between-bin share) when a knob still wins. The same score also helps pick which key π to audit in the first place. §3, §4, Appendix F.
3. **A cheap, optional carry-over check.** For a model that faithfully reproduces its data, the budget should carry over as a matter of course; a modest generated sample confirms it directly — held for every target we tried, bar one flagged exception (crystal stability). §3, §4.

The same predictions hold in domains that share no model, no data, and no code, which argues they are properties of the framework rather than of one setup. The two one-line proofs, the carry-over diagnostics, the lift, and the crystal-model spec are in the Appendix, which carries the full derivations and additional results.

2 Related Work

Every steering method below already occupies one of two categories, telling or showing, and already lives inside a budget this paper is the first to quantify. We take the literatures in order: telling first, then showing, then a phenomenon the split explains.

Controllable generation builds *knobs* for steering — plug-and-play and attribute-guided text decoding (Dathathri et al. 2020; Krause et al. 2021; Li & Liang 2021; Lester et al. 2021) and latent-direction image editing (Jahanian et al. 2020; Shen et al. 2020). These ask which interventions exist; we ask the prior, quantitative question of how far *any* such intervention can move a target, read off the data. Most are *telling* in our terms.

Guidance and alignment. Classifier and classifier-free guidance (Dhariwal & Nichol 2021; Ho & Salimans 2022) and fine-tuning toward a reward — RLHF (Ziegler et al. 2019; Ouyang et al. 2022), its KL-constrained Boltzmann optimum (Korbak et al. 2022), DPO (Rafailov et al. 2023) — reshape the output distribution. We absorb these as closed-form rewrites of the data distribution and recompute the budget on the reshaped data.

Exemplar- and reference-conditioned generation (Gal et al. 2023; Ruiz et al. 2023; Ye et al. 2023; Blattmann et al. 2022) conditions on example sets — what we call *showing*. This is established; what is new here is not the use of examples, but the *budget* that bounds what it can reach.

Retrieval-augmented and in-context generation. Retrieval-augmented generation (Lewis et al. 2020) and in-context / few-shot prompting (Brown et al. 2020) are *showing* in our exact sense — a set of retrieved or in-context examples reshapes the output distribution without a target being named

— in a domain (text) we do not test here. As above, our contribution is not the use of examples, but the budget that predicts, before generation, how far it will reach: the split applies unchanged (any cheap key π bins the outputs; L splits into $T + E$ as usual), and predicts that such examples move a target only to the extent its variance is *between*-bin. We verify this prediction head-to-head against the strongest knob available, in two domains neither RAG nor in-context prompting addresses — image and crystal generation, where "showing" is realized as choosing the conditioning distribution a batch is drawn from rather than literal in-context exemplars — and leave the text-domain measurement to future work.

Diversity and mode collapse (Goodfellow et al. 2014; Salimans et al. 2016) and fidelity/diversity metrics (Sajjadi et al. 2018; Kynkäänniemi et al. 2019; Naeem et al. 2020) document that optimizing for quality costs variety. We give a structural account: a per-output objective reaches only the within-bin budget; between-bin variety is reachable only by acting on the bin mixture (§3.1).

3 Method

This section fixes the vocabulary every later claim reuses: a generator, a cheap key that sorts outputs into bins, an expensive target property, and an audit that reads three numbers per bin.

A generator G produces outputs D, sorted by a **cheap key** π into label $P = \pi(D)$ — an image's class, a crystal's chemistry — giving B **bins** (Table 1 fixes notation). An **expensive scorer** ψ measures the **target** $L = \psi(D)$ we want to steer (aesthetic, band gap). An **audit** — run on any output distribution, e.g. the training distribution $\mathcal{J}$, or a model's own samples when $\mathcal{J}$ isn't available (Appendix G) — records each bin's share w_b, mean $g_b = \mathbb{E}[L \mid P_b]$, and variance v_b: w_b exact since π is cheap, g_b, v_b estimated from a modest scored sample since ψ is not. Unless noted otherwise, "the audit" means the data-side audit, on $\mathcal{J}$.

Table 1: Notation.

symbol	meaning
$\pi,\ P = \pi(D)$	cheap **key**; the label it assigns (sorts outputs into **bins**)
$\psi,\ L = \psi(D)$	expensive scorer; the **target** property to steer
$b,\ B$	a bin; the number of bins
$w_b,\ g_b,\ v_b$	bin b's **share**, **mean** of L, within-bin **variance** of L
$\bar{g}$	overall mean $\sum_b w_b g_b$
$T = \sum_b w_b v_b$	**within-bin budget** — telling's room
$E = \sum_b w_b (g_b - \bar{g})^2$	**between-bin budget** — showing's room
$\mathcal{J},\ \mathcal{J}_P$	training distribution; its bin distribution
μ_P	the **recipe** — a designed distribution over bins (showing sets it)
$\chi^2(\mu_P \Vert \mathcal{J}_P)$	**effort** — how far the recipe departs from the data's mix
reach $= \sqrt{\chi^2 \cdot \text{budget}}$	the most a reweighting can shift the mean
lift $= \mu_P(b)/w_b$	scorer-free estimate of a bin's target-clearing rate
$\delta,\ \rho$	carry-over diagnostics: bin-mean drift; within-bin spread ratio

Assumption (carry-over). For the target L, the trained model's per-bin statistics match the data's: $g_b(\mathrm{G}) \approx g_b$ and $v_b(\mathrm{G}) \approx v_b$ on the bins in play. This holds to the extent L is faithfully reproduced by the model (it is trained to match its data, not to optimize L); we do not assume it, we **test** it per target with the drift δ and spread ratio ρ (§4, Claim 1).

3.1 The budget and the two reaches

This is the paper's central identity, and the two ceilings it implies for how far telling and showing can each move a target.

The split. By the law of total variance (Casella & Berger 2002),

$$\mathrm{Var}_{\mathcal{J}}(L) \;=\; \underbrace{\textstyle\sum_b w_b v_b}_{T\ \text{(within-bin)}} \;+\; \underbrace{\textstyle\sum_b w_b (g_b - \bar{g})^2}_{E\ \text{(between-bin)}}, \qquad \bar{g} = \textstyle\sum_b w_b g_b.$$

T is the room *telling* works in (within one bin); E the room *showing* works in (across bins). The split is exact for any $(\pi, L, \mathcal{J})$ and is set, before any model exists, by the choice of key π — finer bins move variance into E, coarser into T (Appendix F).

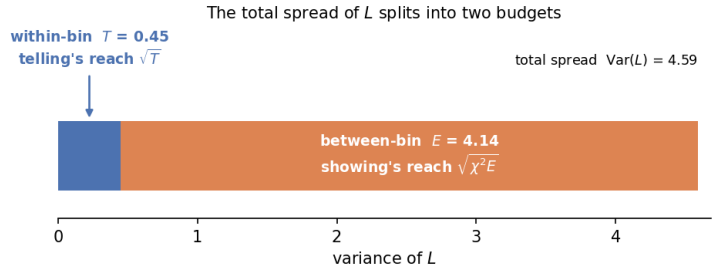

Figure 1: The budget: $\mathrm{Var}(L)$ splits into within-bin T (telling) and between-bin E (showing). Illustrative numbers, not measured data.

Worked example (Fig. 1's numbers). Four bins, equal shares, with means $g = (1.2, 3.0, 4.2, 6.8)$ and within-bin variances $v = (0.16, 1.21, 0.25, 0.18)$: then $\bar{g} = 3.8$, $T = \sum_b w_b v_b = 0.45$, $E = \sum_b w_b (g_b - \bar{g})^2 = 4.14$, so $\mathrm{Var}(L) = 4.59$ and $E/(E+T) = 0.90$. Telling's unit-effort reach is $\sqrt{T} = 0.67$, showing's is $\sqrt{E} = 2.03$: the property lives mostly *between* bins, so a knob confined to one column can move its mean only about a third as far as choosing columns.

The two reaches. Telling and showing are two ways to *reweight* the same distribution — to change which outputs the batch draws more of — and they differ only in *which* set each is allowed to reweight; both obey one law: departing from a set's own proportions shifts its mean by at most $\sqrt{\chi^2 \cdot \sigma^2}$, where σ^2 is the variance of the set reweighted and the Pearson χ^2-divergence measures the size of the departure (Cauchy–Schwarz).

Telling stays inside the named bin: a prompt or guidance knob shifts weight among the outputs *within* a bin (a "golden retriever" prompt still gives you retrievers, just brighter or more textbook ones) but cannot change which bin, so it draws on T,

$$|\Delta\mathbb{E}[L]| \le \sqrt{\chi^2_{\text{tell}}\, T}.$$

In practice the controls push gently — even maximum guidance realizes $\chi^2_{\text{tell}} \approx 1$ (§4) — so telling's working reach is about $\sqrt{T}$.

Showing changes the bin mix instead: handing over an exemplar set is steered not by any one example but by the **recipe** μ_P it implies — a designed distribution over bins (e.g. "70% high-band-gap chemistries, 30% low"; Notation, Table 1). The shift is $\Delta\mathbb{E}[L] = \sum_b (\mu_P(b) - \mathcal{J}_P(b))\, g_b$, and by Cauchy–Schwarz

$$|\Delta\mathbb{E}[L]| \;\le\; \sqrt{\chi^2(\mu_P \| \mathcal{J}_P) \cdot E}, \qquad \chi^2(\mu_P \| \mathcal{J}_P) = \sum_b \frac{(\mu_P(b) - \mathcal{J}_P(b))^2}{\mathcal{J}_P(b)}.$$

The reaches have the same form; the asymmetry is (i) **structural** — telling is locked to T and cannot reach E *at all*, since it cannot cross bins — and (ii) **empirical** — a recipe departs far harder from the data mix than a prompt does, so showing realizes χ^2 of order 100 against telling's $\approx$1 (§4). Both are **ceilings**: Cauchy–Schwarz is tight only when weight concentrates on the extreme-mean bins, so realized shifts land below and the gap grows with χ^2; the min-χ^2 recipe (§3.3) attains the ceiling.

Equal effort, structural gap. At *matched* effort (equal χ^2) the two reaches are $\sqrt{\chi^2 T}$ and $\sqrt{\chi^2 E}$, so showing out-shifts telling by $\sqrt{E/T}$ — a data-fixed ratio **no knob strength can change** (Fig. 1's numbers: $\sqrt{E/T} = 3.0\times$; crystal band gap $7\times$, density $4.9\times$). The realized head-to-head gaps of §4.1 are *larger* still — real knobs spend far less effort than recipes ($\chi^2 \approx 1$ vs ≈ 100) — but the knob-agnostic floor is $\sqrt{E/T}$: even a knob driven to showing's effort is out-shifted by that much. Where the between-bin share is small ($E < T$), $\sqrt{E/T} < 1$ and the knob wins — the brightness case, called in advance.

Telling draws only on T, showing only on E: whichever the data makes larger, wins.

3.2 Which operation, and which shape of recipe

Two questions, in order. **Telling or showing?** — a shift smaller than $\sqrt{T}$ needs only telling; a larger one is out of telling's reach, however the knob is run, so you must show (§3.1). **If showing, concentrate or spread?** — set by the goal's **curvature**. A writable goal is a function $F(\mu_P)$ of the recipe, maximized over the simplex of bin distributions. If F is convex (linear included), its maximum sits at a vertex: *concentrate* μ_P on one bin. Batch averages, rates, and conjunctions ("A and B", $F = \sum_b \mu_P(b)\, c_b$) are the type case — and the winning bin is the one the *audit* finds, not the bin telling starts in. If F is strictly concave its maximum is interior — *spread* μ_P across bins: coverage of two opposed properties is the type case, since no single bin holds both corners. Both shapes are showing; *naming* the goal (telling) reaches neither.

3.3 Designing recipes

§3.2 says which recipe shape a goal calls for; here is how to build one. To hit a target mean g^*: **pool-mixing** blends a low and a high pool with a dial α; **min-χ^2** is the gentlest recipe reaching g^*, in closed form $\mu_P(P_b) = \mathcal{J}_P(P_b)\,[1 + (g^* - \bar{g})(g_b - \bar{g})/\mathrm{Var}_{\mathcal{J}}(g_b)]$, which is the Cauchy–Schwarz-tight direction. When no scorer ψ for L is available (*tacit* steering), μ_P is the bin histogram of an exemplar set, and each bin's **lift** $= \mu_P(b)/\mathcal{J}_P(b)$ estimates the fraction of that bin clearing the target — with no scorer (Appendix).

3.4 Predictions

Four falsifiable claims, tested in §4:

1. **Split is exact and carries.** The split is exact, moves with the key, and *carries to the model* for faithfully reproduced targets.
2. **Reaches are orthogonal.** Telling and showing reach orthogonal parts of the budget; we measure how much further showing reaches than a knob.
3. **Curvature sets the shape.** A goal's curvature sets the shape of the showing recipe — concentrate for an average, spread for coverage — while naming (telling) reaches neither.
4. **Fine-tuning reshapes the budget.** Standard fine-tuning changes the budget predictably (recompute on the reshaped data).

4 Experiments

We test the claims from the Introduction, formalized in §3, in two domains that share no model, data, or code: **images**, where a bin's contents can be checked visually, and **crystals**, generated by a text-based model over text representations of elements and properties — real materials-discovery quantities.

	Images (inspect by eye)	**Crystals** (real materials-discovery stakes)
generator	pretrained DiT-XL/2-256 (Peebles & Xie 2023), used as-is	pretrained text-based encoder–decoder (Appendix: full spec)
training data	ImageNet (Deng et al. 2009)	Alexandria-derived data (Schmidt et al. 2023)
key π	ResNet-50 class (He et al. 2016; 1000 bins)	chemistry: elements + atom count
targets L	brightness; a learned aesthetic score; file-size/MP; four CLIP (Radford et al. 2021) content scores (animal/vehicle/food/nature)	band gap; energy-above-hull; density; dielectric constant; *Combined* — the fraction of generated structures that are novel, unique, and at least metastable, a proxy for viable new-material yield

Both generators are used exactly as released, with no fine-tuning: every result below is produced by choosing which examples to draw, not by retraining anything.

Uncertainty: every headline number carries a 95% CI (bootstrap over bins/structures, $\geq$3 seeds). **Cost:** the data-side claims need only the audit *shadow* (per-bin share, mean, variance — 132 KB) and reproduce with `numpy` in under a second, no model and no GPU.

4.1 Results

Four claims, tested in this order because each leans on the validity of the previous test:

1. **Split is exact, moves with the key, predicts carry-over (pre-registered).** Describes the *trained model*, not just the training data.
2. **Telling and showing reach orthogonal budgets; showing reaches further, except where the budget says otherwise.** The size of showing's win, and its honest limits.
3. **A goal's curvature sets the recipe's shape** — concentrate for an average, spread for coverage. Hits what a goal calls for, not just an average.
4. **Training changes the budget, but predictably, until it doesn't.** Survives realistic training changes, with a named boundary.

Claim 1 — split exact, moves with the key, carries to the model. The split is an identity; coarsening the key shifts it toward within-bin as predicted (images: between $0.0030 \to 0.0002$ merging 1000 classes into a structured 50; a *random* 50-bin key instead collapses $E \approx 18\times$ — so the budget reads the key's structure, not its resolution).

The real test is carry-over: does the trained model's per-bin behavior match the data's? On crystals, three separate targets must land in the right order, not just one number to hit; images then confirm the same story with a single, looser check. Carry-over is measured per bin, by target, with 95% CIs from bootstrap over chemistries and seeds (Fig. 2).

Geometry-like targets carry tightly; stability carries weakly and uncertainly (a small structural error flips it off the hull), exactly where the carry-over assumption (§3) is expected to fail. **This ordering was pre-registered:** from an independent pilot using only support overlap and the δ, ρ diagnostics, we committed in advance (timestamped) to density > bonding > stability, density $\gtrsim 0.9$ and stability $\lesssim 0.4$. The converged audit confirmed the **ordering** exactly (density 0.99, bonding 0.88, stability 0.49 — just above its pre-registered band). On images, the model reproduces 0.96–$1.14\times$ the data's within-bin spread ($\rho \approx 1$), and class-aligned per-bin means track the data at r^2 0.85–0.94. Every comparison in Claims 2–4 below assumes this holds: a magnitude claim on a target that didn't carry over would be measuring the training set, not anything a practitioner could act on.

In short: the data-side budget isn't just a bookkeeping identity — it predicts what a trained model will do, before that model is audited.

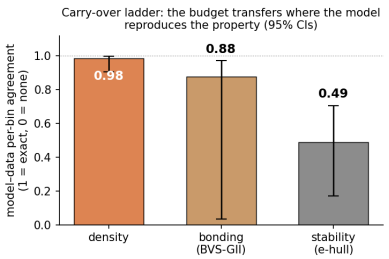

Figure 2: Claim 1: carry-over by target (crystal), 95% CIs: density 0.99, bonding 0.88, stability 0.49 — the pre-registered ordering, confirmed.

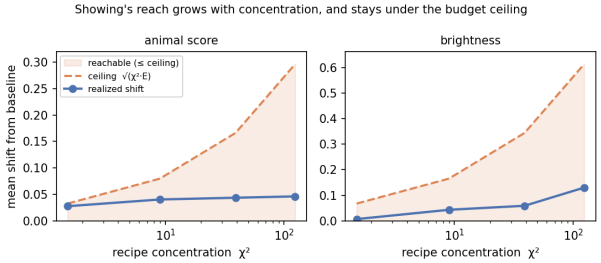

Figure 3: Claim 2: showing's shift vs the ceiling $\sqrt{\chi^2 E}$ as the recipe concentrates — every point stays under its own ceiling.

Claim 2 — telling and showing reach orthogonal budgets, and how much further. Telling stays within one bin; showing ranges across them. We first measure the *effort* each spends: telling swept to maximum classifier-free guidance (CFG = 15) realizes $\chi^2 \approx 1$–1.6, while exemplar recipes reach χ^2 of order 100. At every guidance scale the realized shift stayed under telling's own within-bin ceiling $\sqrt{\chi^2 T}$ — so $\sqrt{T}$ is not a wall telling hits, just the $\chi^2 \approx 1$ scale it can afford. As the recipe concentrates (top 400→100→25→8 of 1000 classes) the realized showing shift grows and stays under $\sqrt{\chi^2 E}$ (animal $0.031 \to 0.051$; brightness $0.036 \to 0.118$). Under the **min-χ^2** recipe the ceiling becomes a *forecast*: at a small target shift the realized move (0.0162) lands on the predicted ceiling (0.0195, inside CI $[0.0114, 0.0209]$), while blunt top-k at the same χ^2 reaches only a fraction of it.

How much further: bins vs. knobs, property by property. Bins beat the deployed knob on eight of nine targets, exactly as $E/(E+T)$ predicts in advance — brightness, the one exception, is exactly where that share says a knob should win:

target	$E/(E+T)$	$\sqrt{E/T}$	bin/knob
band gap (cryst.)	0.98	7.0	≈**800**×
density (cryst.)	0.96	4.9	≈**91**×
stability (cryst.)	0.57	1.15	≈**41**×
vehicle (img.)	0.57	1.15	8.0
food (img.)	0.63	1.31	7.2

target	$E/(E+T)$	$\sqrt{E/T}$	bin/knob
file size/MP (img.)	0.30	0.65	6.0
aesthetic (img.)	0.27	0.61	4.6
animal (img.)	0.72	1.60	3.0
nature (img.)	0.53	1.06	2.8
brightness (img.)	0.17	0.45	**0.5**

The $\sqrt{E/T}$ column is the **equal-effort** gap — what showing wins over a knob driven to the *same* χ^2. It is knob-agnostic (7× band gap, $4.9\times$ density), exceeds 1 wherever the between-bin share is large, and falls below 1 only for brightness (where the knob wins). The realized bin/knob column is *larger* because real knobs realize $\chi^2 \approx 1$ while recipes reach ≈ 100 above — an **effort** gap, not a definitional one — but a stronger knob is still out-shifted by at least the $\sqrt{E/T}$ floor. On the three crystal targets, the deployed knob's 95% CI includes zero — it does essentially nothing — while bin selection's excludes it, so the realized *ratio* (≈41–800×) has a near-zero denominator; treat the knob-free $\sqrt{E/T}$ column instead as the trustworthy equal-effort comparison.

In short: the two operations reach genuinely different, predictable amounts of the budget, and the data-side share says in advance which one wins.

Claim 3 — curvature sets the recipe's shape. Concentrate for an averaging goal; spread for a coverage goal; naming (telling) reaches neither.

An averaging goal (concentrate). Ask for images that are *both* animal and natural — an average over the batch, so the theory says *concentrate*. There is no "animal in nature" class in the dataset, so naming fails, staying near one in seven even with Composable Diffusion (the strongest way to compose two conditions); the audit instead points to bins already both (otters, flamingos), and concentrating the batch on the single richest bin wins outright (66.5%; 128 images/seed × 10 seeds; Fig. 4) — the extreme of showing, since a knob confined to any one bin reaches only $\sqrt{T}$.

Expressiveness bonus. The same exemplar set ranks its own bins by **lift** — how much it over- or under-draws a bin relative to its share — without ever scoring the target, and that ranking predicts realized payoff at rank correlation 0.94: a capability no knob has, since a knob needs the target named and scored first.

The same averaging goal on crystals — wide-gap *and* stable — adds a wrinkle: mixing chemistries (13.2%) beats a naive request five-fold (2.7%) but only ties a strong single request conditioned on the model's own wide-gap prior (15.5%). Stability — the target Claim 1 found does not carry over — sits near one in two for every recipe and caps the joint regardless; concentrating on the right region is enough, spreading adds nothing (Fig. 5).

A spread goal (spread). Now a batch that must **cover both corners** of two properties that pull apart; score coverage as the smaller corner-fraction, so a recipe scores only if the batch holds *both*. Predictions committed before running:

coverage (min corner-fraction)	images	crystals
ask for one (telling, one bin)	0.00	0.00
compose / condition on both	0.00	0.02
mix a corner-A and a corner-B bin (showing)	**0.30**	**0.40**

Telling sits in one corner by construction; composing the two conditions collapses to the muddy middle (car–food chimeras, 92% of the batch) — the go-to "both at once" tool returns the average, not the spread; only showing places genuine examples of both corners, the only recipe above zero by a CI-separated margin. **The gap widens with the number of corners:** telling and composing each cover exactly one corner regardless of N, while showing covers all of them — two of two at $N = 2$, three of three at $N = 3$ — a gap of $N - 1$.

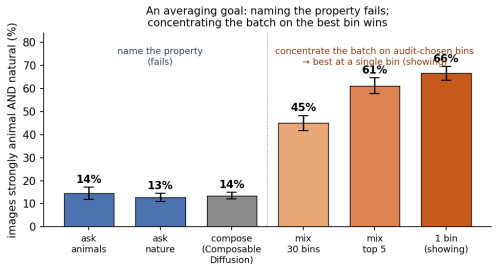

Figure 4: Claim 3 (images): naming the property stays near one in seven; concentrating the batch on audit-chosen bins climbs, peaking at the single richest bin (66.5%).

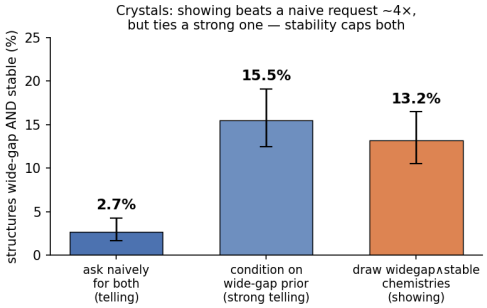

Figure 5: Claim 3 (crystals): showing beats a naive request ~5× but ties a strong one — stability, which does not carry over (Claim 1), caps both.

In short: curvature, not just telling-vs-showing, decides which recipe to use — and both shapes were called correctly in both domains.

Claim 4 — training changes the budget predictably. Standard training and inference-time moves — pretraining, filtering, a guidance sweep — reshape the training distribution $\mathcal{J}$ into an effective $\mathcal{J}'$ in closed form (Appendix B): the new per-bin means follow from the existing audit's numbers alone, without generating outputs under the change and auditing them directly. Our experiments confirm this holds up to a point. Every prediction below lands within a fraction of a bin width of the observed value, except the last row, where the model leaves the training distribution entirely and prediction fails outright:

training move	target(s)	predicted vs. observed
plain pretraining	crystal density	0.99 agreement — on the diagonal
plain pretraining	image content targets	0.85–0.94 agreement, tightening with sample size
filtering (reward-best fraction; $\mathcal{J}' \propto \mathcal{J}e^{R/\beta}$, §2/Appendix B)	crystal	matches the closed-form prediction to 0.03 of a bin width
filtering	image	matches to 0.002 of a bin width
guidance sweep (CFG 1, 3, 7, 12)	image, animal score	0.18 → 0.20, monotone as predicted
guidance sweep	image, brightness	0.52 → 0.57, monotone as predicted
aggressive fine-tune for novelty (off-distribution)	crystal	held-out score −15.1/token [−15.2, −15.0] vs −9.7 random — every structure below; **prediction breaks**

Plain pretraining reproduces $\mathcal{J}$ unchanged, so the budget is the data's budget; filtering and guidance both move the per-bin means in the direction and amount the closed-form rewrite predicts. The last row is that boundary: past it, sample and audit the model directly.

In short: the budget updates predictably under realistic training changes, with a named, measured boundary past which it doesn't.

Two more results test the claims above under pressure.

The strongest knob still loses, or is honestly beaten. We pushed each knob to its strongest realizable form, ruling out a weak-baseline explanation. On crystals, **best-of-twelve** — the strongest within-bin selection a knob can do, at matched effort $\chi^2 \approx 5.3$ — is still out-shifted 26×/14×/4.8× on gap/density/stability. On images, a **learned soft-prompt** trained directly on the scorer flips the two lowest-share targets (aesthetic, brightness) to knob-wins — but only by leaving its own bin entirely, which means it no longer counts as a knob under our own definition. Bins still win ~3× on the higher-share targets. $\sqrt{E/T}$ predicted both outcomes in advance.

Choosing the key is itself auditable. The same $E/(E+T)$, scored data-side, also ranks candidate keys — rank correlation 0.88 to the realized ratio (Appendix F).

What this section showed. All four claims held, in both domains: the split predicts a trained model (Claim 1); showing out-reaches the strongest knob we could build, except where the budget itself says a knob wins (Claim 2); showing hits the shape a goal calls for, not just its average (Claim 3); and training reshapes the budget predictably, up to a named boundary (Claim 4).

4.2 One mechanism behind familiar failures (interpretation)

Four familiar training failures — mode collapse, the alignment tax, reward hacking, and the failure to satisfy two requirements at once — share one cause here. Each asks a per-output objective, such as a likelihood or a per-sample reward, to deliver something only a collection holds: variety, coverage, a joint of two properties. Those live between bins, not inside any single output, so no per-output objective can reach them. Only showing acts across bins — it is the one operation built to fix what a collection needs, not what any one sample does.

5 Limitations

Four caveats are worth naming plainly before the detail: the reach bounds are idealized and can loosen off-support; carry-over — the theory's one contact with a trained model — held for every target we tried, bar one flagged exception; one crystal measurement leans on an approximate proxy (Appendix E); and each domain was tested on a single generator. None of these touch the budget itself, which is a property of the data alone, not of any model.

The reach bounds in §3 are selection-only, second-moment ceilings that hold where the model matches the data on its bins; precise scope and failure modes are in Appendix E. The no-$\mathcal{J}$ fallback, for when the training data itself isn't available, is reliable for the qualitative call and for E under a representative reference, less so under a badly wrong one, and for T either way — Appendix G.

Carry-over is the theory's one empirical input, and the one place it can fail even when everything else holds: it assumes the trained model reproduces the data's per-bin statistics — false for stability, the one target that didn't carry (Claim 1), which is also why it caps the wide-gap-and-stable joint (Claim 3) and why prediction stops working past Claim 4's fine-tuning boundary. Both are measured, pre-registered diagnostics, not caveats noticed after the fact.

Each domain uses one generator: the T/E budget is a property of the data and the key, not the model, so it is invariant to model scale and architecture by construction — only *carry-over* is model-specific, with a per-model test (Appendix B). A second generator per domain would probe carry-over robustness, not the budget; we leave it to future work.

6 Conclusion

How far a generative model can be steered toward a property is fixed, before training, by a budget split into a part a knob reaches and a part only examples reach — read off the data alone, no model, no GPU, and cheaply confirmed to carry over to a trained model. Verified in two unrelated domains, it predicts which targets a knob cannot reach and, where it falls short, which *shape* of exemplar recipe closes the gap: concentrate the batch on one audit-chosen bin for an average, spread it across bins for coverage. Showing also reaches what a knob cannot: a target only recognized, never named. The practical rule: audit, see which half of the budget a goal lives in, and use the operation that covers it.

Ethics Statement

This work uses only public, non-personal data — image-generation benchmarks and materials-generation datasets — and pretrained/openly released generators. It does not involve human subjects, personal data, or crowd labor, and we do not foresee dual-use or societal-harm concerns beyond those already associated with the underlying generative models.

Reproducibility Statement

All experiments use public datasets and openly released or reproducible generators; architectures, training data, and checkpoint hashes for the crystal generator are given in Appendix D, and scorers/diagnostics in Appendices B–C. A complete code repository that reproduces every reported number exists; to preserve double-blind anonymity it is not linked here, but will be provided to reviewers on request and made public upon acceptance.

AI Use Statement

In this work, we used generative AI tools to assist with executing experiments — writing and running the code that produced the results reported here — and with formatting the manuscript (the pandoc → LaTeX build pipeline and document preparation). We did not use generative AI for the theory, claims, proofs, hypotheses, methodology, data cleaning, qualitative analysis, or interpretation of results. All AI-assisted output was reviewed and verified by the authors.

References

- Batatia, Kovács, Simm, Ortner, Csányi. "MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields." NeurIPS 2022.
- Blattmann, Rombach, Oktay, Müller, Ommer. "Retrieval-Augmented Diffusion Models." NeurIPS 2022.
- Brown, Mann, Ryder, Subbiah, Kaplan, Dhariwal, et al. "Language Models are Few-Shot Learners." NeurIPS 2020.
- Casella, Berger. *Statistical Inference*, 2nd ed. Duxbury, 2002.
- Dathathri, Madotto, Lan, Hung, Frank, Molino, Yosinski, Liu. "Plug and Play Language Models: A Simple Approach to Controlled Text Generation." ICLR 2020.
- Deng, Dong, Socher, Li, Li, Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database." CVPR 2009.
- Dhariwal, Nichol. "Diffusion Models Beat GANs on Image Synthesis." NeurIPS 2021.
- Emmerich, Deutz, Klinkenberg. "The Computation of the Expected Improvement in Dominated Hypervolume of Pareto Front Approximations." Technical Report, Leiden University, 2006.
- Gal, Alaluf, Atzmon, Patashnik, Bermano, Chechik, Cohen-Or. "An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion." ICLR 2023.
- Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio. "Generative Adversarial Nets." NeurIPS 2014.
- He, Zhang, Ren, Sun. "Deep Residual Learning for Image Recognition." CVPR 2016.
- Ho, Salimans. "Classifier-Free Diffusion Guidance." arXiv:2207.12598, 2022.
- Jahanian, Chai, Isola. "On the 'Steerability' of Generative Adversarial Networks." ICLR 2020.
- Korbak, Perez, Buckley. "RL with KL Penalties is Better Viewed as Bayesian Inference." Findings of EMNLP 2022.
- Krause, Gotmare, McCann, Keskar, Joty, Socher, Rajani. "GeDi: Generative Discriminator Guided Sequence Generation." Findings of EMNLP 2021.
- Kynkäänniemi, Karras, Laine, Lehtinen, Aila. "Improved Precision and Recall Metric for Assessing Generative Models." NeurIPS 2019.
- Lester, Al-Rfou, Constant. "The Power of Scale for Parameter-Efficient Prompt Tuning." EMNLP 2021.
- Lewis, Perez, Piktus, Petroni, Karpukhin, Goyal, et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." NeurIPS 2020.
- Li, Liang. "Prefix-Tuning: Optimizing Continuous Prompts for Generation." ACL-IJCNLP 2021.
- Naeem, Oh, Uh, Choi, Yoo. "Reliable Fidelity and Diversity Metrics for Generative Models." ICML 2020.
- Ouyang, Wu, Jiang, Almeida, Wainwright, et al. "Training Language Models to Follow Instructions with Human Feedback." NeurIPS 2022.
- Peebles, Xie. "Scalable Diffusion Models with Transformers." ICCV 2023.
- Polanyi. *The Tacit Dimension.* Doubleday, 1966.
- Radford, Kim, Hallacy, Ramesh, Goh, Agarwal, et al. "Learning Transferable Visual Models From Natural Language Supervision." ICML 2021.
- Rafailov, Sharma, Mitchell, Ermon, Manning, Finn. "Direct Preference Optimization: Your Language Model is Secretly a Reward Model." NeurIPS 2023.
- Ruiz, Li, Jampani, Pritch, Rubinstein, Aberman. "DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation." CVPR 2023.
- Sajjadi, Bachem, Lucic, Bousquet, Gelly. "Assessing Generative Models via Precision and Recall." NeurIPS 2018.

- Salimans, Goodfellow, Zaremba, Cheung, Radford, Chen. "Improved Techniques for Training GANs." NeurIPS 2016.
- Schmidt, Hoffmann, Wang, Borlido, Carriço, Cerqueira, Botti, Marques. "Machine-Learning-Assisted Determination of the Global Zero-Temperature Phase Diagram of Materials." *Advanced Materials* 35(22):2210788, 2023.
- Shen, Gu, Tang, Zhou. "Interpreting the Latent Space of GANs for Semantic Face Editing." CVPR 2020.
- Ye, Zhang, Liu, Han, Yang. "IP-Adapter: Text Compatible Image Prompt Adapter for Text-to-Image Diffusion Models." arXiv:2308.06721, 2023.
- Ziegler, Stiennon, Wu, Brown, Radford, Amodei, Christiano, Irving. "Fine-Tuning Language Models from Human Preferences." arXiv:1909.08593, 2019.

Appendix

A. Proofs of the two reaches (one line each)

The split (law of total variance). For any $(\pi, L, \mathcal{J})$, conditioning L on its bin label P gives $\mathrm{Var}(L) = \mathbb{E}[\mathrm{Var}(L \mid P)] + \mathrm{Var}(\mathbb{E}[L \mid P]) = \sum_b w_b v_b + \sum_b w_b (g_b - \bar{g})^2 = T + E$. ■

The reaches (Cauchy–Schwarz). Reweighting a set of variance σ^2 by a distribution μ (from its own proportions $\mathcal{J}$) shifts its mean by $\Delta = \sum_b (\mu_b - \mathcal{J}_b) g_b = \langle \mu/\mathcal{J} - 1,\ g - \bar{g} \rangle_{\mathcal{J}}$. By Cauchy–Schwarz in the $\mathcal{J}$-inner product, $|\Delta| \le \|\mu/\mathcal{J} - 1\|_{\mathcal{J}} \, \|g - \bar{g}\|_{\mathcal{J}} = \sqrt{\chi^2(\mu \| \mathcal{J})} \sqrt{\sigma^2}$, tight when $\mu - \mathcal{J} \propto (g - \bar{g})$ (the min-χ^2 recipe of §3.3). Applied within a bin ($\sigma^2 = T$, telling) or across bins ($\sigma^2 = E$, showing) gives the two reaches. ■

B. Carry-over diagnostics (δ, ρ) and the data→model bridge

Carry-over is **not** knowable before the model exists: δ and ρ compare the *trained model's* per-bin audit to the data's, so they need model samples. What is pre-model is the *prediction* of carry-over from cheap pilot signals (support overlap + a loss-sensitivity proxy) — which is what we pre-registered (§4.1, Claim 1). The two checks: - **Bin-mean drift** $\delta(\mu_P) = \sqrt{\mathbb{E}_{\mu_P}[(g_{\mathrm{G}}(P) - g_{\mathcal{J}}(P))^2 / v_P]}$ — did the bin means move? - **Within-bin spread ratio** $\rho(b) = v_{\mathrm{G}}(b)/v_{\mathcal{J}}(b)$ — did the within-bin variety carry?

A model can pass one and fail the other (right bins, wrong labels within them). For a *known* training intervention the model approximates a reshaped $\mathcal{J}'$ in closed form — MLE: $\mathcal{J}' = \mathcal{J}$; RLHF/KL-PPO: $\mathcal{J}'(D) \propto \mathcal{J}(D) e^{R(D)/\beta}$; filtering on S: $\mathcal{J}' = \mathcal{J}\,\mathbf{1}[D \in S]/Z$; DPO: Boltzmann with implicit reward — so recompute the per-bin w_b, g_b, v_b on $\mathcal{J}'$ and every result carries over at no new audit cost. When the intervention is aggressive enough to leave $\mathcal{J}$'s support, audit the model's own samples instead.

C. The lift — scorer-free bin selection

With no scorer for L, the recipe is the bin histogram of an exemplar set, and each bin's **lift** $= \mu_P(b)/w_b$ estimates the fraction $q_b = s \cdot \mathrm{lift}_b$ of that bin clearing the target (s = the exemplar set's overall hit rate). A small bin that soaks up exemplars has large lift; a bin the exemplars ignore has lift ≈ 1. The global $\chi^2 = \sum_b w_b (\mathrm{lift}_b - 1)^2$ is just the w-weighted spread of the lifts (rank correlation 0.94 to realized rates, §4.1).

D. The crystal model

Encoder–decoder transformer (6 encoder / 18 decoder layers, model dim 768, ≈**238 M** params; 16k SentencePiece vocab), trained by supervised next-token cross-entropy on an Alexandria-derived set (≈**225k** structures, <20 atoms, low energy-above-hull, LeMat-Bulk overlap excluded). A prompt is `<elements> | <natoms> | <property-bin tokens>`; the decoder emits a structure encoding parsed to an ASE `Atoms`. Distribution-steering uses naked (`elements | natoms |`) prompts. Stability = single-MACE (Batatia et al. 2022) energy vs an MP hull; *Combined* = SUN + MSUN (stable/metastable ∧ novel ∧ unique); also BVS-GII, SMACT/validity. The checkpoint (hash-pinned) and all scorers are released.

E. Scope and the curvature table

The reaches bound the **mean shift by selection** — not literal extremes — and hold where the model matches the data on the bins in play (off-support, $\chi^2 \to \infty$, flags failure); they are second-moment

ceilings, so heavy-tailed bin means make them loose. Set-level targets (coverage, diversity) are a *different object* than a mean: the framework says **which** operation reaches them (a mixture — showing) but not how large the metric gets. **Curvature table:** convex/linear goals (mean, rate, precision, conjunction) → *concentrate* on one bin; concave goals (coverage, diversity, spread, match) → *spread* across bins; both are showing. For n properties the reachable means fill the convex hull of the bin points, and covering a Pareto front (Emmerich et al. 2006) needs specialist bins unless one bin's within-bin spread already reaches the corners. (The variance-vs-range argument and the full pitfalls list are in the extended version.)

A domain-specific proxy. The crystal-joint band gap claim (§4.1, Claim 3) uses a composition surrogate in place of the full structural calculation, stricter than it at the threshold. The result's direction and the stability ceiling are robust to this substitution; the absolute rate is not.

F. Choosing the key π and bin granularity

Two extremes bound a usable key. One bin for the whole distribution ($B = 1$) forces every output to share the audited mean, so $E = 0$ and $T = \mathrm{Var}(L)$: the whole budget sits within one bin, and since ψ is expensive, any draw returns the same average — nothing is targetable. One bin per output ($B = N$) is scoring every output directly, forfeiting the cheap-key/expensive-scorer economy the framework rests on (§3): with a single scored sample per bin there is no within-bin sample to estimate v_b from, and the audit no longer generalizes past whichever outputs happened to be scored. A usable π sits between: coarse enough that g_b, v_b are estimated from a modest scored sample per bin, fine enough that bins actually differ.

Bin *count* alone does not fix the split — the key's *structure* does. On images, coarsening 1000 ImageNet classes into a structured 50 moves E from 0.0030 to 0.0002 (§4.1, Claim 1) — the expected effect of coarsening — but a *random* 50-bin partition of the same size collapses E a further $\approx 18\times$. Two keys of identical granularity can differ almost twenty-fold in what they expose, because one groups outputs by a property the target actually varies with and the other does not: the budget reads the key's structure, not its resolution.

This makes key choice itself auditable rather than a modeling guess: the same crossover score used to compare telling and showing, $E/(E+T)$, computed data-side for each candidate key, ranks candidates by how much of the target's variance they expose between bins — rank correlation 0.88 to each key's realized bin-selection gain, with an uninformative random partition scoring at the floor (§4.1). Picking π is a data-side audit like any other quantity in this framework, not a choice made blind.

G. Auditing without training data

Everything above assumes access to the training distribution $\mathcal{J}$. When only the trained model G is available — a third-party checkpoint, an undocumented fine-tune, training data that no longer exists — the audit still runs, on samples instead of rows.

The recipe. Choose a reference distribution ν_P over the key's support, sample many outputs from G under it, and treat them as a synthetic dataset: read off $\pi(D_i)$ and score $L(D_i)$ for each, then run the identical audit (T, E, per-bin means) on this sample in place of $\mathcal{J}$. Three defaults for ν_P, by purpose: the training marginal $\mathcal{J}_P$ if it can be inferred (closest analog to a real data audit); the deployment distribution, if the goal is characterizing G in actual use; or a uniform sweep over π's support, for per-cluster diagnostics with no natural reference. Report ν_P alongside any such audit — the recovered budget depends on it.

What we found. We tested this in both domains: sample a released generator under several guessed ν_P and compare the recovered T, E to the true, $\mathcal{J}$-based values (crystals: uniform-over-real-bins, a real-but-skewed pool, a pool with no support overlap with the training data, and an oracle reweighting; images: uniform, a skewed subset, and a disjoint but still in-distribution subset). Two things hold in both domains: the *qualitative* call — whether telling or showing dominates — survives any ν_P that is at least broadly representative of the real population, even a skewed one ($E/(E+T)$ stays within a few points of its true value); and E, the number that decides that call, recovers to within a few percent under the same guesses. T recovers far less cleanly — it is sensitive to both ν_P and to how outputs are binned (by the intended prompt or by what the model actually produced), sometimes in opposite directions — so a T recovered this way should be read as approximate, never as the split's exact value. One target-dependent miss also showed up on images, independent of ν_P: for a target

where telling is already the more competitive operation, E was consistently underestimated by about half — a limitation of the audit in that regime, not of the reference-distribution choice.

What still breaks it. A ν_P with essentially no support overlap with the training population — tested on crystals only — collapses both numbers and flips the qualitative call outright ($E/(E+T)$ fell from 0.97 to 0.15 in our test). No comparable wrong-support test was run on images, so this failure mode should not be assumed to generalize, or not, to that domain — it is simply untested there. The recipe above is only as good as ν_P's overlap with what the model actually saw; report ν_P alongside any such audit.