How Many, Not Which: Automatic Data Budgets for Conditional GANs Using Gradient Spectra

Anonymous authors
Paper under double-blind review

Abstract

Data subset selection promises cheaper training. While the coreset literature focuses on deciding which samples to keep at a practitioner-specified size, our evidence suggests that for conditional GANs the predominant factor is how many samples. At equal budgets, subsets drawn uniformly at random during training nearly always tie or outperform gradient matching, coverage matching, and difficulty scoring. Meanwhile, changing data quantity and allocation for each condition moves quality well outside between-seed variation. To address this gap in determining a subset's quantity, we introduce FLARE (Fast Learning via Adaptive Rank Estimation), which derives the data budget from the effective rank of each condition's centered per-sample critic gradient spectrum, i.e., the number of independent directions the gradient signal spans. That budget is then split across conditions proportionate to those ranks, tilted toward the conditions the generator currently fits worst, ordered by per-condition sliced Wasserstein distance. The budget is an output of that spectrum rather than a required input, so there is no size parameter to tune. We apply FLARE to optiGAN—a conditional Wasserstein GAN with gradient penalty (WGAN-GP) that stands in for state-of-the-art Monte Carlo simulation of optical photon transport in radiation detectors—to determine a coreset size under 2% of a 513k-sample pool. With random subsets at this coreset size, optiGAN trains 19x faster (including incurred selection costs, excluding evaluation) for the same number of optimizer steps, matching full-data training on global sliced-Wasserstein similarity at a minimal quantified cost in per-condition fidelity. Because the criterion is solely based on the critic's own gradients, it transfers across domains without re-tuning. As such, we applied FLARE unchanged to calorimeter shower simulation and to tabular synthesis on a network-intrusion benchmark. FLARE's selected fraction differs across these domains by about a factor of two. Deriving the budget instead of tuning it removes the sweep that data reduction normally demands.

1 Introduction

Conditional generative models are increasingly used as fast stand-ins for expensive data-generating processes (HEP Software Foundation, 2018; Hashemi & Krause, 2024). A slow but trusted process produces a large dataset once, a network learns to sample from it, and the network replaces the process wherever full fidelity is not required (Paganini et al., 2018; Trigila et al., 2023). A surrogate is specific to the configuration it was trained on, so each change to the underlying system often requires another training run on another large dataset, and the recurring cost is training rather than inference.

In nuclear imaging, such as positron emission tomography (PET), Monte Carlo simulation is the standard for radiation detector prototyping (Agostinelli et al., 2003; Sarrut et al., 2026), typically for gamma photons of energy 100-600 keV, detecting by scintillation crystals in which they interact. Simulating the optical photons detected from a single 511 keV gamma interaction can take a second or more for a typical detector configuration (Trigila et al., 2023). A PET scan involves millions of such interactions, which puts whole-system simulation out of reach. Optical simulation is restricted in practice to a single detector or a detector pair, and system-level predictions must be extrapolated from that pair under assumptions of uniformity and independence that a real scanner does not satisfy.

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.

This limits the ability of simulations to aid in our understanding of how a change to one detector affects the performance of the whole system (Trigila et al., 2023; Naunheim et al., 2026).

Generative adversarial networks (GANs) (Goodfellow et al., 2014) have shown promise for learning these simulations. A network trained on the simulator's own output samples the target distribution directly, replacing the expensive simulation with a single inference. High energy physics (HEP) uses similar GAN stand-ins for calorimeter shower simulation (Paganini et al., 2018; Faucci Giannelli & Zhang, 2024; Krause et al., 2025). Beyond HEP, GANs synthesize network flow records for intrusion detection, where real traffic is scarce and privacy-restricted (Ring et al., 2019). In this work, we test our method in both, alongside optiGAN.

Training cost often scales with the size of its dataset, making training on a subset the most direct way to make it cheaper. Data subset selection studies how to choose that subset, and nearly all of the effort goes into membership, i.e., deciding which k samples to keep for a given budget k. Gradient matching (Killamsetty et al., 2021; Mirzasoleiman et al., 2020), difficulty scoring (Paul et al., 2021) and distribution coverage (Chen et al., 2010) are three families of answers to that question. Most such methods take the budget as input, which varies from one model/dataset to another. Finding it means training at several sizes and comparing, which spends the computation that reducing the data was meant to save.

For a conditional generator the question of "how much?" is more complex. The model fits one distribution per condition rather than one overall, and conditions are not equally hard, so the budget has to be split across conditions as well as sized. One global fraction answers neither question. In our experiments with optiGAN, no rule we tested for choosing which samples to keep improved on drawing them uniformly at random at the same subset size, while changing how many samples the subset holds, or how they are divided between conditions, moved quality by more than retraining with a different random seed does. We therefore introduce FLARE (Fast Learning via Adaptive Rank Estimation), which decides during training how many samples each condition needs and draws those samples uniformly at random. For each condition, FLARE takes the gradient each training sample contributes to the critic, the network that scores real against generated samples in a Wasserstein GAN. It then counts how many independent directions those gradients span, a quantity known as their effective rank. When a condition's gradients span only a few directions, a few samples already cover them, so that condition receives a small budget. Our contributions are (1) FLARE, which sets each condition's data budget during training from the effective rank of its per-sample critic gradients, so no subset size or ratio has to be tuned, (2) a worst-first allocation that divides that budget toward the conditions the generator currently fits worst, (3) a sliced-Wasserstein similarity normalized against a degenerate reference, and (4) a closed form for the per-sample gradient of a Wasserstein critic with a gradient penalty, which makes per-sample sizing tractable (Appendix C).

1.1 Related work

Most coreset methods fix the size and optimize membership (Moser et al., 2025), so the closest prior work falls into two groups, the membership families we test against uniform random selection and the few methods that allocate a budget across classes or derive its size.

Gradient matching coresets. GradMatch (Killamsetty et al., 2021) and CRAIG (Mirzasoleiman et al., 2020) pick a small weighted subset whose weighted gradient sum approximates the full-data gradient, GradMatch by orthogonal matching pursuit (OMP) on the gradient residual and CRAIG by greedy submodular maximization over a gradient-similarity graph. Both are designed for supervised training and both match a mean gradient. A rare sample barely moves that mean, so neither method has a reason to keep it. CoLM (Nguyen et al., 2025) reports this failure for gradient-matched mini-batch coresets, which under-represent small data sources. Greedy matching also relies on per-sample gradients spread over many directions, and a critic's per-sample gradients are close to collinear, so the residual stops falling after a handful of picks (Appendix A). OMP can stop at a residual tolerance and CRAIG at a gradient-error tolerance, sizing a subset implicitly, but the tolerance then has to be tuned. CRAIG's facility-location objective is also a coverage objective, the 1-Wasserstein quantization error of the gradients (Graf & Luschgy, 2000), so we adapt GradMatch's OMP objective to the critic and use it throughout as the membership control that uniform random selection is compared against.

Gradient-based data pruning. EL2N and GraNd (Paul et al., 2021) score each sample's difficulty early in training and keep the highest-scoring samples. EL2N is the norm of the error $p(x) - y$

between a classifier's predicted class probabilities and the label. A Wasserstein critic has no such error, while GraNd, the expected norm of a sample's loss gradient, is well defined on a critic. Keeping samples by score changes the distribution the model trains on, and Ayed & Hayou (2023) show that this distribution shift provably degrades generalization in the high-compression range of 30% or less of the data, where random pruning remains consistent and outperforms most score-based methods. A generator learns its training distribution itself, so a shift in that distribution is an error in the generator's target, and FLARE's budgets of under 2% of the pool sit far inside that range. Keeping the lowest-scoring samples instead shifts the distribution the other way (Appendix A).

Data distribution coverage matching. Coverage coresets choose points whose empirical distribution is close to the pool's under a discrepancy, a kernel discrepancy for kernel herding (Chen et al., 2010) and kernel density estimate coresets (Phillips & Tai, 2018), the energy distance for support points (Mak & Joseph, 2018), and a Wasserstein distance for optimal quantization (Graf & Luschgy, 2000), which our sliced-Wasserstein coreset applies along random projections (Bonneel et al., 2015). A uniform random subset already matches the pool's distribution in expectation, so the most a coverage coreset can add is a tighter match at the same size. Variants that spread points over the support instead, such as farthest-point sampling (Gonzalez, 1985) or k-means++ seeding (Arthur & Vassilvitskii, 2007), over-represent sparse regions and distort the density the generator learns. Instance selection (DeVries et al., 2020) distorts it deliberately in the other direction, dropping low-density samples to raise sample fidelity at the cost of the sparse regions, which suits image synthesis but not a surrogate that has to reproduce a detector's full response. On optiGAN none of these families improves on uniform random membership at a matched budget (Appendix A).

Per-condition budget allocation. DRoP (Vysogorets et al., 2025) and NUCS (Zhang et al., 2025) both allocate a given budget non-uniformly, DRoP by per-class error with random pruning within classes, and neither derives the total. The closest related work on deriving size is refined coreset selection (Xia et al., 2024), which searches for the minimal size by bilevel optimization with training in the loop, while we read it from one spectrum with no search. Sharifi & Harati (2026) also select training data from a spectrum, but score which samples to drop by a Fiedler vector while a pre-scheduled ratio sets how many, which is the inverse of the direction we take. Importance sampling reweights how samples are drawn at a fixed count, and GAN sample-complexity bounds hold uniformly over a class of critics, so neither sets a data budget from one trained critic's gradients (Appendix H).

2 METHODS

We develop FLARE on optiGAN (Trigila et al., 2023; Naunheim et al., 2026), a conditional Wasserstein GAN with gradient penalty (WGAN-GP) (Mirza & Osindero, 2014; Arjovsky et al., 2017; Gulrajani et al., 2017) that replaces GATE (Sarrut et al., 2026) optical photon tracking, with a pool of 513,042 photon rows over 108 training emission points and evaluation on 846 held-out ones. CaloShowerGAN (Faucci Giannelli & Zhang, 2024) and CTGAN (Xu et al., 2019) take the method unchanged, the first on 533-voxel calorimeter showers over 15 incident energies (Krause et al., 2025) and the second on KDD Cup 1999 network-intrusion records over five attack groups (Appendix B).

Selection is posed per condition and repeats on a fixed cadence. At a selection event, the critic's output layer gradient for a single training row is that row's atom, and a condition's atoms stack into its atom matrix. FLARE reads the matrix spectrum and returns a count of how many rows are needed to represent the spectral energy of each condition until the next selection event, where the total budget is the sum of those counts over conditions. FLARE never decides which rows, as samples are drawn uniformly at random within a condition.

2.1 EVALUATION METRICS

We measure quality on optiGAN with the sliced 1-Wasserstein distance (Rabin et al., 2011; Bonneel et al., 2015), which averages the one-dimensional Wasserstein distance W_1 between real and generated samples projected onto L random unit directions u_ℓ, and report it as a similarity against a degenerate reference,

$$\mathrm{SW}(P,Q) \;=\; \frac{1}{L}\sum_{\ell=1}^{L} W_1\big(P_{u_\ell},\, Q_{u_\ell}\big), \qquad s(P,Q) \;=\; 1 - \frac{\mathrm{SW}(P,Q)}{\mathrm{SW}(P,\delta)}, \tag{1}$$

where P is the real sample, Q the generated one, P_u the distribution of $u^\top x$ for $x \sim P$, and δ a reference collapsed onto a single point. The normalizer is what the distance reads for a model that has learned nothing of the distribution's shape, so s runs from 0 at that degenerate reference to 1 at an exact match, and it is the score worst-first allocation orders conditions by (Section 2.3). Globally we report s on samples pooled across conditions, and a variant that takes the 99th percentile over projections in place of the mean. The variant follows the worst-matched directions, so it picks up a missing region of the distribution, such as a tail, that the mean dilutes. Per condition we compute s on each held-out condition, and the primary endpoint is the mean over the worst quartile of conditions, with the median and the mean over all conditions beside it. Precision and recall (Sajjadi et al., 2018) and realism come from k-nearest-neighbor hyperballs (Kynkäänniemi et al., 2019), computed per condition and averaged over conditions. Precision is the fraction of generated samples landing inside the real data manifold and reads as fidelity, recall the fraction of real samples landing inside the generated one and reads as coverage, and realism is precision made continuous, scoring how far inside the real manifold a generated sample sits, not only whether it is. All three measure support rather than density, so they can disagree with the sliced-Wasserstein numbers.

We score the two cross-domain models with the endpoints their own literature uses. For CaloShowerGAN this is the chi-squared per degree of freedom between histograms of total deposited energy in generated and reference showers, pooled over the 15 incident energies, lower is better (Faucci Gianelli & Zhang, 2024). For CTGAN it is machine-learning efficacy (Xu et al., 2019), in which a decision tree and a multilayer perceptron (MLP) train on synthetic rows and are tested on real held-out rows. We report the mean of their macro F1, the unweighted mean over the five classes of each class's F1 score, which is the harmonic mean of that class's precision and recall, so every class counts equally however rare it is. On CTGAN we also report the raw sliced-Wasserstein distance between synthetic and real rows as a check on fidelity.

2.2 Spectral effective rank sizing

Per-sample critic gradients for one condition span far fewer directions than the parameter dimension, consistent with deep-network gradients staying in a low-dimensional subspace (Gur-Ari et al., 2018; Zhou et al., 2021). At a selection event, condition c's atoms stack into a matrix $A_c \in \mathbb{R}^{n_c \times p}$, where p is the parameter dimension of the critic's output layer. At that layer each atom, gradient-penalty term included, has a closed form, so a batched pass replaces a per-sample loop and cuts the cost of a selection event about 30-fold (Appendix C). We center its columns and take the $r = \min(n_c, p)$ singular values $\sigma_1 \geq \cdots \geq \sigma_r$ of the centered matrix $\tilde{A}_c$ by a singular value decomposition (SVD) in float64 (Appendix D). The condition's budget k_c is how many of them are needed to hold a fraction θ of the squared spectral energy (Jolliffe, 2002; Eckart & Young, 1936),

$$k_c \;=\; \min\left\{ m \;:\; \sum_{i=1}^{m} \sigma_i^2 \;\geq\; \theta \sum_{i=1}^{r} \sigma_i^2 \right\}. \tag{2}$$

The minimization is over m, the number of singular values kept counting down from the largest, so k_c is the fewest of them that reach θ. It is a count of atoms, and those k_c atoms are what condition c contributes until the next selection event. The total budget is $\alpha \sum_c k_c$ for a rank multiple α. Every result we report runs at $\alpha = 1$ (no scalar multiple), except the budget ablation of Table 7. The total is an output of the spectrum and never an input to it, so the realized budget is reported as an outcome.

The energy criterion of Equation 2 needs θ because it counts up to a cutoff. The Roy-Vetterli entropy effective rank (erank) (Roy & Vetterli, 2007) removes the cutoff by normalizing the singular values into a distribution and taking the exponential of its Shannon entropy,

$$\operatorname{erank}(\tilde{A}_c) \;=\; \exp\Big(-\sum_{i=1}^{r} q_i \log q_i\Big), \qquad q_i \;=\; \frac{\sigma_i}{\sum_{j=1}^{r} \sigma_j}, \tag{3}$$

rounded to the nearest integer to give k_c. It reads the whole shape of the spectrum rather than counting up to a cutoff, returning r for a flat spectrum and 1 for a spectrum on a single direction, with no number left to set. We offer erank as a cheaper, fully parameter-free option, however it sizes smaller and incurs a small but resolvable quality cost. The two values of θ for the energy criterion we tested produced statistically indistinguishable quality results, although the tighter one selects twice the data (Table 1 and Figure 10). Over that range θ changes the budget but not the quality, so it does not act

as a hidden subset ratio. A third estimator, the Gavish-Donoho optimal hard threshold (Gavish & Donoho, 2014), fails on gradient atoms, and we report it as a negative result (Appendix D).

With a subbatch size s, an atom averages s rows rather than standing for one, which divides the height of A_c by s and multiplies the budget in rows by the same factor, since k_c counts atoms. optiGAN and CaloShowerGAN run at $s = 1$ and CTGAN at $s = 8$ (Appendix D).

(Figure 11) shows how the budget is a property of the data rather than of simply the pool it was drawn from. On a simulation dataset with 5x the samples, the number of samples chosen by FLARE when training optiGAN is effectively the same (within seed variation) as the original pool. A fixed ratio of samples would lead to non-optimal pool sizing. When setting a hard cap post hoc that is below the estimator's demand, the estimators collapse onto one identical budget, and sit at the cap throughout training while their ranks stay high and keep their ordering. The estimators therefore differ in demand. In practice, the count per condition is bounded by the atom matrix itself. The rank of $\tilde{A}_c$ cannot exceed $\min(n_c, p)$, so the ceiling sets the count once a condition's atom count falls below the output-layer parameter dimension. It also sets the count early in training, since the spectrum starts flat and every condition reads near the ceiling at any subbatch size. Centering costs one direction, so a condition with n_c atoms can express at most $n_c - 1$ of them. Once $n_c - 1$ falls below the condition's intrinsic dimensionality the effective rank saturates there, and the sizer stops measuring and returns the condition whole (Section 4.1).

2.3 Worst-first budget allocation

Sizing sets the total, and allocation divides it across conditions. With the tilt off the allocation weights are the per-condition counts, so condition c receives αk_c, exactly its own count at $\alpha = 1$. The tilt reweights those counts by how well the generator currently fits each condition. Order the C conditions by the sliced Wasserstein similarity of Section 2.1, worst first, and let $\pi(c)$ be condition c's position in that order. The allocation weight is

$$w_c \;=\; k_c \big(C + 1 - \pi(c)\big), \tag{4}$$

so the worst-fit condition has its count multiplied by C and the best-fit condition by one. The total $K = \alpha \sum_c k_c$ is apportioned in proportion to w_c by largest remainder, so the allocated counts $\hat{k}_c$ sum to K exactly. A condition that asks for more than it can supply is clamped, and the excess passes to conditions with headroom instead of being dropped, so the tilt moves budget between conditions without changing how much there is. The weight uses position in the ordering rather than the similarity value itself, which keeps the allocation invariant to the scale of the score and stops one badly fit condition taking the whole budget. Scores are read on the training conditions and refreshed whenever the sizer runs, so the held-out evaluation set never enters the allocation.

3 Results and Experiments

Both optiGAN sweeps train for 5000 epochs at six repeats per configuration and share one full-data baseline of six repeats. A reduced-data configuration scales its batch size with its realized budget, so every run takes the same number of optimizer steps as the full data training. A configuration sits above or below a reference on an endpoint when the 95% bootstrap interval of the difference excludes zero. It is equivalent to full data when a two one-sided test (Schuirmann, 1987) places the difference within twice full data's own across-repeat standard deviation.

Estimator and tilt. The first sweep crosses the tilt, on or off, with erank and the energy criterion at $\theta = 0.999$ and $\theta = 0.9999$ (Table 1). Every configuration trains on under 2% of the pool, and the two tilted energy configurations are equivalent to full data on the primary endpoint. With the tilt on, the parameter-free erank sits below the energy criterion at $\theta = 0.999$, with the interval of the difference clear of zero. All six configurations sit slightly below full data on precision and recall, and the tilted energy configuration at $\theta = 0.999$ also sits below it on the mean and median condition (Section 4.1). Switching the tilt on with the sizer setting held fixed raises the primary endpoint in all three pairs, with every interval clear of zero. Two of the three tilted configurations gain on less data than their untilted pair and the third gains as much at a matched budget, so the gain comes from reallocation, not size. Budgets differ within a pair because allocation fixes the total only within a selection event, and the composition it sets changes the spectrum that sizes the next one. In every pair the tilt also lowers recall, while global similarity, precision and realism do not move.

Table 1: The two optiGAN sweeps against one full-data baseline, six repeats per configuration. Budget is the median realized share of the pool over the final 1000 epochs, Δ the difference in worst-quartile similarity from full data with its 95% bootstrap interval, and speedup is training plus selection against full data.

Configuration	Budget	Δ vs full data	Equivalence	Speedup
Estimator and tilt				
erank, tilt off	0.48%	-0.0259 [-0.0330, -0.0188]	different	12.6x
erank, tilt on	0.44%	-0.0130 [-0.0217, -0.0048]	inconclusive	12.0x
energy $\theta = 0.999$, tilt off	1.02%	-0.0134 [-0.0223, -0.0035]	inconclusive	12.5x
energy $\theta = 0.999$, tilt on	**0.77%**	**$+0.0020$ [-0.0058, $+0.0103$]**	**equivalent**	**12.7x**
energy $\theta = 0.9999$, tilt off	1.55%	-0.0143 [-0.0204, -0.0074]	different	12.7x
energy $\theta = 0.9999$, tilt on	1.56%	$+0.0004$ [-0.0072, $+0.0083$]	equivalent	12.8x
Cadence, membership and budget				
FLARE (resize 100)	1.16%	$+0.0042$ [-0.0019, $+0.0112$]	equivalent	13.0x
FLARE (resize 500)	**0.96%**	**$+0.0023$ [-0.0066, $+0.0107$]**	**equivalent**	**19.1x**
FLARE, $\alpha = 2$	2.25%	$+0.0056$ [$+0.0002$, $+0.0118$]	equivalent	12.2x
FLARE, $\alpha = 4$	4.62%	$+0.0166$ [$+0.0100$, $+0.0241$]	different (better)	9.7x
OMP membership control	0.70%	-0.2019 [-0.2158, -0.1873]	different	7.1x
Full data	100%	—	—	1.0x

Training plus selection costs about the same in all six configurations, although the largest budget is more than three times the smallest (Table 1). Training time stays flat because every run takes the same number of steps and a step at these batch sizes costs nearly the same whatever the batch holds. Selection time stays flat because each event gathers and decomposes the whole pool, whatever count the sizer returns. With the tilt on, the energy criterion at $\theta = 0.999$ therefore trains on nearly twice the data of erank and runs no slower.

Cadence, membership and budget. The second sweep reruns the tilted energy configuration at $\theta = 0.999$ unchanged with new seeds, and each of four further configurations changes one field of it (Table 1). The rerun is again equivalent to full data, although its realized budget came out larger than in the first sweep (Section 4.1). Selection runs every 100 epochs, and a resize interval of 500 epochs re-runs the sizer only at every fifth event, redrawing membership at the others from the cached per-condition counts. The resize-500 configuration is equivalent to the resize-100 repeat on the primary endpoint and spends about a fifth of its selection time. Training plus selection then runs 19x faster than full data (Table 1), and FLARE uses this cadence by default. End-to-end wall clock improves only to 4.7x, because periodic quality evaluation costs the same in every run, full data included, and takes most of a reduced-data run's wall clock (Table 8 and Figure 9). Every FLARE configuration reaches the worst-condition similarity of full data at under a tenth of the training plus selection cost (Figure 1).

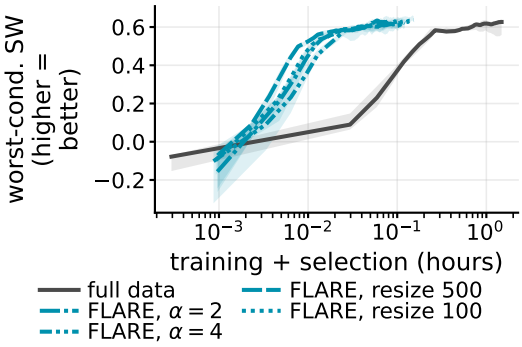

Figure 1: Worst-condition sliced-Wasserstein similarity against cumulative training plus selection cost, excluding evaluation. Median of six repeats with the range shaded. The FLARE curves start at a higher cost than full data because their first point includes the initial selection event.

The membership control replaces uniform random membership with OMP and leaves the sizer and tilt unchanged. It falls far below full data on every endpoint we report, and on the primary endpoint its gap is about eight times the largest gap of any configuration with uniform membership (Table 1). The gap is not a budget effect, since the tilted erank configuration of the first sweep selects even less data with uniform membership and stays much closer to full data. The membership null of Appendix A therefore holds with the default sizer and tilt as well. Raising the rank multiple to $\alpha = 2$ and $\alpha = 4$ doubles and quadruples the budget with the allocation rule held fixed, and every endpoint in Table 7

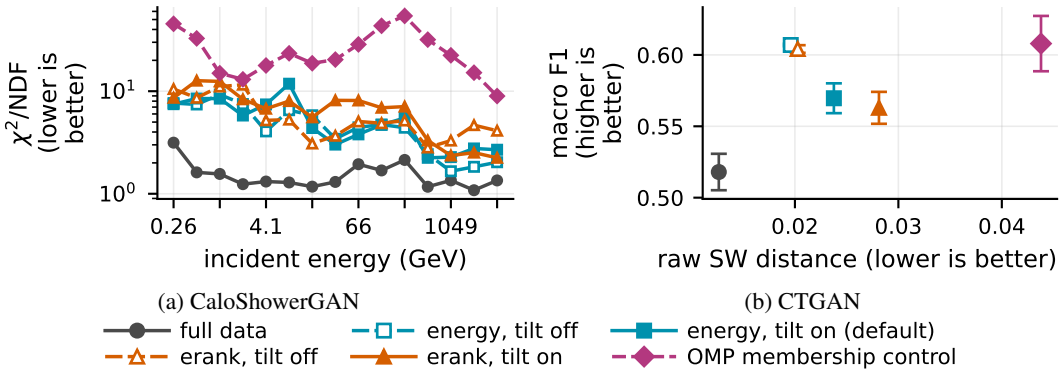

Figure 2: Cross-domain results, mean of three repeats. (a) Chi-squared per degree of freedom of the deposited energy on CaloShowerGAN at each of the 15 incident energies, at the checkpoint each run selects on the training file. (b) Macro F1 against raw sliced-Wasserstein distance to the real table on CTGAN, with the across-repeat standard deviation of macro F1 as the bar.

Table 2: Cross-domain sweeps at three repeats, endpoints as in Figure 2 and Appendix B. Batch is the mean batch in rows, sd the across-repeat standard deviation, and bold marks the default.

	CaloShowerGAN					CTGAN				
Configuration	Budget (%)	Batch	χ^2/NDF ↓	sd	Δ vs. full data	Budget (%)	Macro F1 ↑	sd	Δ vs. full data	Raw SW ↓
full data	—	1024	1.56	0.07	—	—	0.5180	0.0128	—	0.0126
energy, tilt off	1.58	16	4.88	0.86	+3.33	0.768	0.6069	0.0009	+0.0890	0.0196
energy, tilt on	**1.60**	**16**	**5.39**	**0.87**	**+3.83**	**0.776**	**0.5696**	**0.0105**	**+0.0517**	**0.0238**
erank, tilt off	1.23	13	5.94	0.47	+4.38	0.446	0.6042	0.0027	+0.0862	0.0203
erank, tilt on	0.88	9	6.87	1.20	+5.31	0.430	0.5629	0.0112	+0.0450	0.0281
OMP control	1.54	16	26.06	3.47	+24.51	0.788	0.6079	0.0194	+0.0900	0.0438

improves with it except recall, which does not move. The flat cost of the first sweep holds at $\alpha = 2$ but not at $\alpha = 4$, where the larger batch lengthens training (Table 1).

Cross-domain transfer. CaloShowerGAN and CTGAN each run the energy criterion at $\theta = 0.999$ and erank with the tilt on and off, the OMP membership control and full data, at three repeats per configuration, scored by the endpoints of Section 2.1. A difference is resolved when it exceeds two across-repeat standard errors and, on CTGAN, the benchmark's resolution floor of 0.033 macro F1. On CaloShowerGAN the sizer returns under 2% of the pool with no tuning, against a ceiling of 2.5% set by 15 conditions and the critic's 201-parameter output layer (Table 2). Every reduced-data configuration with uniform membership has 3.1 to 4.4 times full data's chi-squared, with the gap widest at low incident energy (Figure 2a). The OMP membership control, at the same budget and batch as the default configuration, sits more than six times further from full data than the default does, so the membership null holds here as on optiGAN. As on optiGAN, the batch scales with the budget, leaving 9 to 16 rows per step against full data's 1024, so the gap reflects a small batch as well as a small subset (Section 4). The tilt and estimator contrasts are unresolved on chi-squared at three repeats (Table 3). On in-loop worst-condition similarity the tilt improves the energy configuration, and on erank it ties while lowering recall (Appendix B).

On CTGAN every FLARE configuration scores a higher macro F1 than full data while training on under 0.8% of the pool, the OMP control included (Table 2, Figure 2b). Every configuration runs at full data's batch and step count, cycling the subset through each epoch, so the reduced-data ones differ from full data only in the rows they train on. The sizer sets each class's count by the effective rank of its gradients rather than its frequency, so a pool that is 79% one class becomes a subset that is 11 to 19% that class. The gain sits entirely in the three rare classes, and the two already at ceiling do not move. CTGAN's own log-frequency training-by-sampling already flattens the class mix, and the sizer flattens it further without being told the class frequencies. Full data keeps the lowest raw sliced-Wasserstein distance to the real table, likely because that distance weights every

row equally and so is dominated by the majority class the sizer thins. No configuration gives each class an equal share of the same budget, so the sweep cannot yet separate spectral sizing from plain class rebalancing.

The tilt lowers macro F1 under both estimators on CTGAN, although every tilted configuration still scores above full data (Table 4). It moves budget into a class that is under 1% of the real test split and lowers that class's F1, likely because the downstream classifiers then over-predict it. The two estimators do not separate at either tilt setting, although erank selects a little over half as much data. With the tilt on, OMP membership scores a higher macro F1 than its matched uniform twin, by a margin that barely clears the resolution floor. It only reaches the macro F1 that uniform membership reaches with the tilt off, however, and has the largest raw sliced-Wasserstein distance in the sweep, so it does not improve on uniform membership once both endpoints are counted.

4 Discussion

On all three domains, the gains come from the budget and its split across conditions, while no membership rule improves on uniform random membership once every endpoint is counted (Section 3 and Appendix A). A condition's per-sample critic gradients are close to collinear, spanning a few tens of directions across thousands of rows (Figure 10). A gradient-based rule then finds little stable signal to separate one row from another, since its selections barely reproduce across runs of identical code. Its preferences show up instead as a distorted density (Figure 6), while uniform random membership is unbiased for each condition's distribution by construction. The sizer reads the same low effective rank as a small count, so the structure that defeats gradient-based membership is what lets FLARE keep so little data. CTGAN's pool comes from a KDD99 file that is 70.5% exact duplicate rows by our count (cf. Tavallaee et al., 2009), so uniform draws there return many copies of the same records. OMP rarely picks an atom close to the span of its earlier picks and so skips near-duplicates, which likely explains why OMP membership raises macro F1 on CTGAN with the tilt on.

The tilt moves budget toward the conditions the generator currently fits worst. For optiGAN the hardest conditions improve at no measurable cost except recall, likely because budget concentrated on hard conditions is budget taken from everywhere else, and recall is the support-coverage metric. Recall follows the budget split rather than its size, since the untilted energy configurations stay close to full data's recall and four times the budget leaves it flat while every other endpoint improves (Table 7). The rank multiple scales every condition's count alike, so the best-fit condition keeps a weight of one against the worst-fit condition's C at any budget. That is likely why a higher budget does not bring recall back. On CTGAN the tilt raises the worst-fit class's share of the synthetic training rows further above its share of the real data and lowers macro F1, likely by shifting the downstream classifiers' class prior (Section 3). The tilt is therefore an option to choose per task, on when the worst conditions set the requirement, as for a detector surrogate that has to hold up at every emission point, and off when coverage matters more or a downstream model is sensitive to the class mix.

Most of the remaining gap to full data on optiGAN appears to come from the smaller batch that turns the data saving into speed, rather than from the data itself. At the default budget, several hundred rows per step replace the 90,000 rows of the full data. In an earlier optiGAN sweep, the tilted erank configuration trained on 0.68% of the pool at full data's batch and step count, cycling the subset, and exceeded full data on the worst quartile by $+0.0189$ [$+0.0129$, $+0.0254$] with no resolved difference on precision or the mean condition, while the same configuration with its batch scaled sits below full data on both. Subset size accounts for part of that difference, since the scaled-batch configuration selected only 0.43% of the pool. The full-batch configuration runs at 0.92x full data's speed, however, so the speedup comes from the scaled batch, whose cost falls mainly on the typical conditions. CaloShowerGAN shows the largest gap of the three domains, likely because its scaled batch leaves the critic only 9 to 16 rows per step. CTGAN keeps full data's batch, and every configuration there scores above full data on macro F1.

4.1 Limitations

Eligibility and scaling. The critic must score one sample at a time, or no per-sample atom exists, so CTGAN runs with its critic's sample packing switched off. The conditions must also be included in

the data to ensure allocation. The total budget is a sum of per-condition counts, so the budget grows with the number of conditions by design, with the rank multiple and the subbatch size adjusting it without fixing a ratio. We have not measured that scaling within a single domain, since across our three domains the condition count changes together with the model and the data.

Speedup. Data reduction removes FLOPs, but wall clock only falls if the baseline was spending its time on FLOPs. At small model size or small batch size, the per-iteration cost is dominated by kernel-launch and Python overhead that is fixed regardless of how much data is in the batch. OptiGAN is 19x faster on training plus selection, but not on the two cross-domain models. In a latency probe on one GPU, CaloShowerGAN at a 2.10% budget returns 0.99x and CTGAN at 3% returns 1.01x, while the same budgets on CPU, where the identical work is compute-bound, return 4.3x to 5.9x.

Scope. The sizer produces a budget with no tuning in all three domains, while the disparity between budget quality and full-data quality depends on each domain's training protocol. The tilt transfers as a mechanism but not as a default, because what it improves depends on the endpoint (Section 4). The sizer is a heuristic, not a bound, so nothing guarantees how close a FLARE subset comes to full data.

Variance and resolution. Realized budget has a high variance across repeats and sweeps, so we quote medians and show the spread. Two independent sets of repeats of an identical configuration, differing only in seed, put the median budget at 0.77% and 1.16% of the pool (Table 1) at indistinguishable quality. FLARE reads the budget off the critic's gradients, so the budget inherits the variability of an adversarial training trajectory, in which small differences between seeds change which conditions the generator fits well and how the critic's spectrum evolves. On CaloShowerGAN, quality also varies roughly ten times as much across seeds with reduced data as with full data (Table 2), likely because each run trains on its own sequence of small random subsets at a batch of 9 to 16 rows. At three repeats that spread leaves the domain's tilt and estimator contrasts unresolved.

Typical-condition fidelity. With the tilt on, FLARE favors the hardest conditions, matching full data on the worst quartile at $\alpha = 1$ and trailing it slightly on the median and mean condition, precision, recall and realism, by about 2% of full data's value or less (Table 7). Every configuration in both optiGAN sweeps scales its batch with its budget, so neither sweep separates the cost of data reduction from that of a smaller batch (Section 4). At 4x the budget, the mean condition is equivalent to full data, so the budget limit is a trade-off between speed and practitioner-defined requirements.

5 Future Work

The immediate next step is to compare FLARE with full data at equal wall clock rather than at equal optimizer steps. Holding the batch at full data's size and letting the smaller subset shorten each epoch instead would remove the batch confound on optiGAN and CaloShowerGAN. Reading quality against wall clock under that design would characterize the trade-off between the budget and the achievable speedup.

On CTGAN, giving each class an equal share of FLARE's budget would separate spectral sizing from plain class rebalancing. Repartitioning one domain into coarser or finer conditions would measure how the budget scales with the number of conditions. A tilt that orders conditions by the task's own endpoint, such as per-class F1, may carry the optiGAN gain to domains whose endpoints weigh conditions differently. The rank trajectory may also serve as an auto-stop if its decay over training tracks the point where more training stops improving quality.

6 Conclusion

For the conditional GANs studied here, the data budget matters more than membership, since uniform random membership nearly always ties or beats the tested membership rules at a matched budget. FLARE derives that budget from the effective rank of each condition's per-sample critic gradients, so there is no size parameter to tune. On optiGAN it selects under 2% of the pool and trains 19x faster on training plus selection, equivalent to full data on the worst quartile of conditions and within about 2% of full data's value on precision, recall and the mean and median condition.

AI USE STATEMENT

In this work, we used generative AI tools for code writing, brainstorming methods, fetching/summarizing literature, initial paper drafting, formalizing the mathematical claims in Appendix C, and proposing feedback on adding statistical rigor to our analysis (hence the addition of the two one-sided tests). We did *not* use generative AI tools for synthetic data generation, cleaning/reformatting datasets, proposing hypotheses, designing research methodology, or interpreting results. All AI-assisted work has been thoroughly reviewed, and every design choice and result interpretation was made by the authors. AI-written code runs under a unit-test suite, every results table/figure is generated by script from logged runs, and every reference was checked against its primary source. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 2.1 and Appendix F define every endpoint, and Appendix B describes the three models, their data and their conditions. Appendix C derives the closed-form per-sample gradient and checks it against autograd. Appendix E lists the training and selection settings of each domain and how each run's seed derives from a base seed, its configuration and its repeat index. Appendix F gives the sweep definitions, repeat counts, the rule behind every reported verdict and the hardware and software the timings come from. Every result comes from a configuration recorded in full alongside the run. GPU kernels ran in PyTorch's default non-deterministic mode, so a rerun reproduces the spread across repeats rather than identical subsets. CaloChallenge dataset 1 (Krause et al., 2025) and the KDD Cup 1999 data (Stolfo & Fan, 1999) are public. The optiGAN data cannot be made publicly available upon publication because they are not available in a format that is sufficiently accessible or reusable by other researchers. They are available upon reasonable request from the authors. Code will be released upon publication.

REFERENCES

S. Agostinelli et al. GEANT4 – a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A*, 506(3):250–303, 2003. doi: 10.1016/S0168-9002(03)01368-8.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. In *International Conference on Machine Learning (ICML)*, 2017.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In *International Conference on Machine Learning (ICML), PMLR v70*, pp. 224–232, 2017.

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1027–1035, 2007.

Fadhel Ayed and Soufiane Hayou. Data pruning and neural scaling laws: Fundamental limitations of score-based algorithms. *Transactions on Machine Learning Research (TMLR)*, 2023.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015. doi: 10.1007/s10851-014-0506-3.

Jorge Cadima and Ian Jolliffe. On relationships between uncentred and column-centred principal component analysis. *Pakistan Journal of Statistics*, 25(4):473–503, 2009.

Yutian Chen, Max Welling, and Alexander J. Smola. Super-samples from kernel herding. In *Uncertainty in Artificial Intelligence (UAI)*, pp. 109–116, 2010.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9268–9277, 2019.

Felix Dangel, Frederik Kunstner, and Philipp Hennig. BackPACK: Packing more into backprop. In *International Conference on Learning Representations (ICLR)*, 2020.

Terrance DeVries, Michal Drozdzal, and Graham W. Taylor. Instance selection for GANs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. doi: 10.1007/BF02288367.

Michele Faucci Giannelli and Rui Zhang. CaloShowerGAN, a generative adversarial networks model for fast calorimeter shower simulation. *European Physical Journal Plus*, 139(7):597, 2024. doi: 10.1140/epjp/s13360-024-05397-4.

Matan Gavish and David L. Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014. doi: 10.1109/TIT.2014.2323359.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2013.

Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. doi: 10.1016/0304-3975(85)90224-5.

Ian Goodfellow. Efficient per-example gradient computations. *arXiv preprint*, 2015.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Siegfried Graf and Harald Luschgy. *Foundations of Quantization for Probability Distributions*, volume 1730 of *Lecture Notes in Mathematics*. Springer, 2000. doi: 10.1007/BFb0103945.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint*, 2018.

Baran Hashemi and Claudius Krause. Deep generative models for detector signature simulation: A taxonomic review. *Reviews in Physics*, 12:100092, 2024. doi: 10.1016/j.revip.2024.100092.

HEP Software Foundation. HEP software foundation community white paper working group – detector simulation. *arXiv preprint*, 2018.

Paul Honeine. An eigenanalysis of data centering in machine learning. *arXiv preprint*, 2014.

Kaiyi Ji and Yingbin Liang. Minimax estimation of neural net distance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2nd edition, 2002. doi: 10.1007/b98835.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International Conference on Machine Learning (ICML), PMLR v80*, 2018.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. GRAD-MATCH: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning (ICML), PMLR v139*, 2021.

Claudius Krause, Michele Faucci Giannelli, Gregor Kasieczka, Benjamin Nachman, Dalila Salamani, David Shih, Anna Zaborowska, et al. CaloChallenge 2022: A community challenge for fast calorimeter simulation. *Reports on Progress in Physics*, 88(11):116201, 2025. doi: 10.1088/1361-6633/ae1304.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations (ICLR)*, 2022.

Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. PacGAN: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Simon Mak and V. Roshan Joseph. Support points. *The Annals of Statistics*, 46(6A):2562–2592, 2018. doi: 10.1214/17-AOS1629.

V. A. Marchenko and L. A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Matematicheskii Sbornik*, 72(114)(4):507–536, 1967.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint*, 2014.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning (ICML), PMLR v119*, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Brian B. Moser, Arundhati S. Shanbhag, Stanislav Frolov, Federico Raue, Joachim Folz, and Andreas Dengel. A coreset selection of coreset selection literature: Introduction and recent advances. *arXiv preprint*, 2025.

Stephan Naunheim, Brandon Pardi, Guneet Mummaneni, Carlotta Trigila, and Emilie Roncali. optiGAN for crystal arrays: Physics-informed generative modeling of optical photon transport in PET detector arrays. *Physics in Medicine and Biology*, 71(16):165022, 2026. doi: 10.1088/1361-6560/ae95d7.

Deanna Needell, Nathan Srebro, and Rachel Ward. Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Dang Nguyen, Wenhan Yang, Rathul Anand, Yu Yang, and Baharan Mirzasoleiman. Mini-batch coresets for memory-efficient language model training on data mixtures. In *International Conference on Learning Representations (ICLR)*, 2025.

Michela Paganini, Luke de Oliveira, and Benjamin Nachman. CaloGAN: Simulating 3d high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. *Physical Review D*, 97(1):014021, 2018. doi: 10.1103/PhysRevD.97.014021.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994. doi: 10.1162/neco.1994.6.1.147.

Jeff M. Phillips and Wai Ming Tai. Near-optimal coresets of kernel density estimates. In *Symposium on Computational Geometry (SoCG), LIPIcs v99*, 2018. doi: 10.4230/LIPIcs.SoCG.2018.66.

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision (SSVM)*, volume 6667 of *Lecture Notes in Computer Science*, pp. 435–446. Springer, 2011. doi: 10.1007/978-3-642-24785-9_37.

Markus Ring, Daniel Schlör, Dieter Landes, and Andreas Hotho. Flow-based network traffic generation using generative adversarial networks. *Computers & Security*, 82:156–172, 2019. doi: 10.1016/j.cose.2018.12.012.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *European Signal Processing Conference (EUSIPCO)*, pp. 606–610, 2007.

Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. Technical Report CS-2008-08, Technion, Computer Science Department, 2008.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

David Sarrut, Nicolas Arbor, Thomas Baudier, Julien Bert, et al. GATE 10 Monte Carlo particle transport simulation: I. Development and new features. *Physics in Medicine & Biology*, 71(1): 015042, 2026. doi: 10.1088/1361-6560/ae237b.

Donald J. Schuirmann. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics*, 15(6):657–680, 1987. doi: 10.1007/BF01068419.

Mohammadreza Sharifi and Ahad Harati. Efficient training of deep networks using guided spectral data selection: A step toward learning what you need. *Data Mining and Knowledge Discovery*, 40 (4):45, 2026. doi: 10.1007/s10618-026-01215-w.

Salvatore Stolfo and Wei Fan. KDD Cup 1999 data. UCI Machine Learning Repository, 1999.

Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pp. 1–6, 2009. doi: 10.1109/CISDA.2009.5356528.

Carlotta Trigila, Anirudh Srikanth, and Emilie Roncali. A generative adversarial network to speed up optical Monte Carlo simulations. *Machine Learning: Science and Technology*, 4(2):025005, 2023. doi: 10.1088/2632-2153/acc782.

Artem Vysogorets, Kartik Ahuja, and Julia Kempe. DRoP: Distributionally robust data pruning. In *International Conference on Learning Representations (ICLR)*, 2025.

Xiaobo Xia, Jiale Liu, Shaokun Zhang, Qingyun Wu, Hongxin Wei, and Tongliang Liu. Refined coreset selection: Towards minimal coreset size under model performance constraints. In *International Conference on Machine Learning (ICML), PMLR v235*, 2024.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Ashkan Yousefpour et al. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint*, 2021.

Hanyu Zhang, Zhen Xing, Ruian He, Wenxuan Yang, Chenxi Ma, Weimin Tan, and Bo Yan. Non-uniform class-wise coreset selection for vision model fine-tuning. *arXiv preprint*, 2025.

Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *International Conference on Machine Learning (ICML), PMLR v37*, 2015.

Yingxue Zhou, Zhiwei Steven Wu, and Arindam Banerjee. Bypassing the ambient dimension: Private SGD with gradient subspace identification. In *International Conference on Learning Representations (ICLR)*, 2021.

A WHY MEMBERSHIP FAILS TO BEAT RANDOM

We compared three membership families with uniform random membership on optiGAN at fixed budgets, difficulty scoring by GraNd at 1% and 3% of the pool and four coverage objectives together with orthogonal matching pursuit (OMP) gradient matching at 1% and 5%, each read on worst-condition similarity at the last epoch all of its runs reached (Appendix F). None improves on uniform random membership, and the best of them tie it (Figures 3 and 5).

Gradient matching saturates because a condition's atoms are close to collinear, with an effective rank of roughly 12 to 30 out of about 750, so the OMP residual stops falling long before the budget is spent. The saturation is not the float32 artifact of Appendix D, since refit in float64 the residual falls monotonically and still flattens early. Sample-level membership also barely reproduces across two runs of identical code, so there is little row-level signal for any rule to exploit in the first place.

Coverage matching ties uniform random membership at best, and so does OMP gradient matching. Against a fixed random subset of the same size, sliced-Wasserstein coresets and OMP tie it at both budgets, while maximum mean discrepancy (MMD) herding and the two principal component analysis (PCA) variants sit below it at 1%, the PCA variants furthest (Figure 5). Difficulty scoring sits below random whether the highest-scoring or the lowest-scoring rows are kept, and falls further as the score becomes more reliable (Figure 4). A noisy score would tie random, so a sharp ranking that lowers quality points to a systematic bias in what the score prefers.

That bias is a distortion of the density the generator learns. We measure a configuration's selection footprint, how often each row is picked across a run, by its Jensen-Shannon divergence from the footprint of the uniform random configurations at the same budget, in standard deviations of a multinomial null drawn at that budget. At 3% of the pool every GraNd configuration sits about 2,600 null standard deviations away while the random configurations sit at or below the null, and the scored subsets' feature densities depart from the pool's where the fixed random subset's do not (Figure 6). The deviation separates scored from random configurations but does not order the scored ones among themselves, so what separates them lies in which features they distort rather than in how uneven their footprint is. This is the distribution shift Ayed & Hayou (2023) prove for score-based pruning, and a conditional generative adversarial network (GAN) learns the shifted density along with everything else. We keep uniform random membership even where a coverage objective ties it, since a tie does not justify a method that needs its own hyperparameters and a selection pass. Uniform random selection is also unbiased for each condition's distribution by construction, so it has no failure mode to tune around.

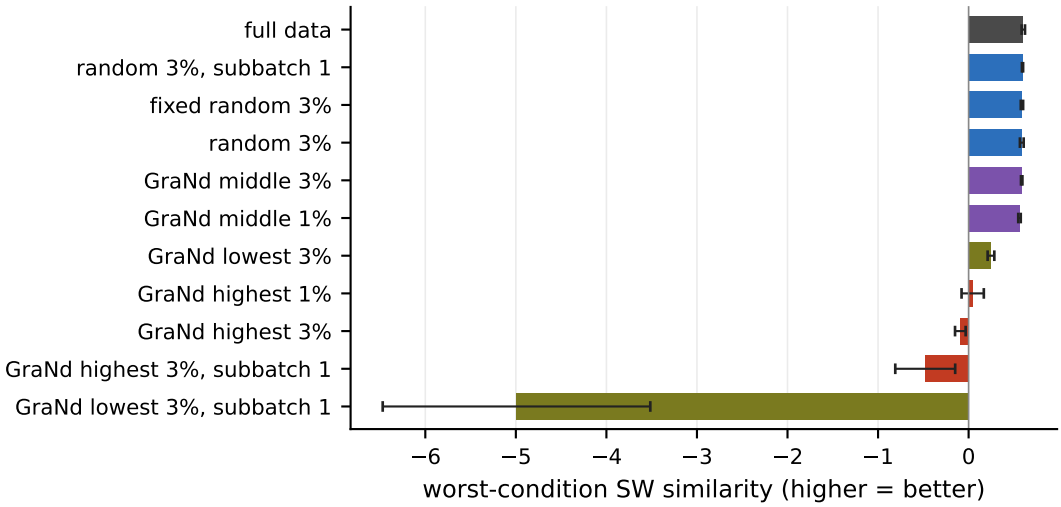

Figure 3: Worst-condition sliced-Wasserstein similarity on optiGAN for GraNd membership, uniform random membership and full data at epoch 2000, as the mean over the last tenth of evaluations and over three repeats (five for random 3% and fixed random 3%), with the across-repeat standard deviation as the bar. Highest, middle and lowest name the band of the gradient-norm ranking a configuration keeps, and percentages are its fixed budget. Subbatch 1 scores single rows where the rest score four-row subbatches, and fixed random keeps its first draw for the whole run.

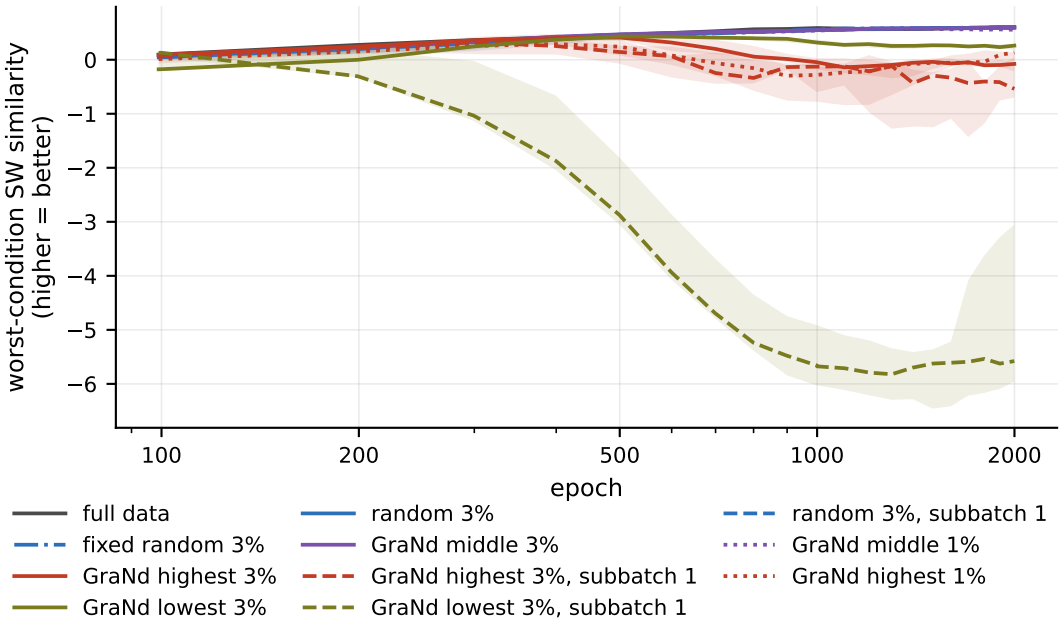

Figure 4: Worst-condition sliced-Wasserstein similarity over the first 2000 epochs for the configurations of Figure 3, median over repeats with the across-repeat range shaded.

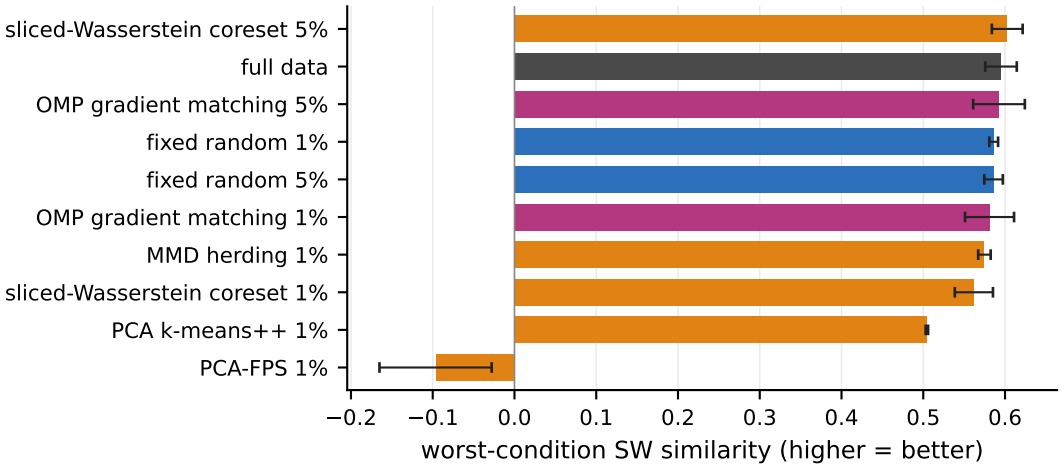

Figure 5: Worst-condition sliced-Wasserstein similarity on optiGAN for four coverage objectives, OMP gradient matching, a fixed uniform random subset and full data at epoch 1500, as the mean over the last tenth of evaluations and over three repeats, with the across-repeat standard deviation as the bar. Percentages are the fixed budget, and OMP gradient matching selects per condition with worst-first allocation at subbatch 64, where the rest use 256.

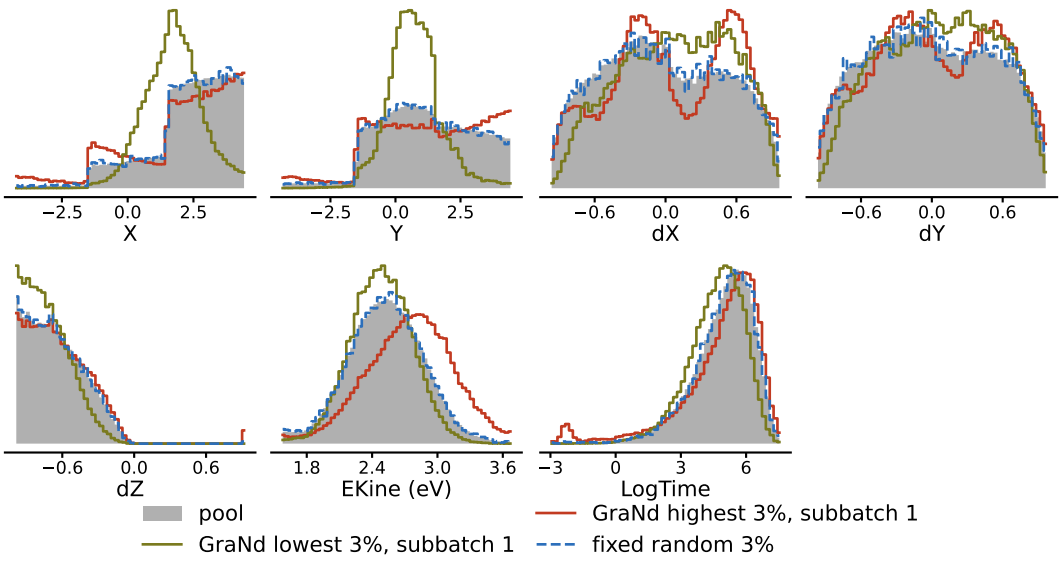

Figure 6: Density of each photon feature over the optiGAN pool, and over the rows selected in one run of two GraNd configurations and of the fixed random subset of Figure 3, each row weighted by how many selection events picked it. X and Y are the exit position on the photodetector face, dX, dY and dZ the direction, EKine the energy (Trigila et al., 2023) and LogTime the logarithm of the arrival time.

B DOMAINS, MODELS AND DATA

B.1 OPTIGAN

The primary model is optiGAN, a conditional Wasserstein GAN with gradient penalty (WGAN-GP) (Mirza & Osindero, 2014; Arjovsky et al., 2017; Gulrajani et al., 2017) with parameter weight scaling and a generator exponential moving average (EMA), trained on GATE (Sarrut et al., 2026) output that replaces optical photon tracking for a scintillation detector (Trigila et al., 2023; Naunheim et al., 2026). Its pool is a simulation of a series of gamma interactions with a 3x3 bismuth germanate crystal array containing 513,042 training rows of seven photon features conditioned on 108 gamma emission points taken from one octant of the crystal volume (Figure 7). Evaluation runs on a held-out set of 846 emission-point conditions spread densely across all nine crystals, which reaches well beyond the octant the training conditions come from, and on a control set restricted to points not present in training.

B.2 CALOSHOWERGAN ON CALOCHALLENGE DATASET 1

CaloShowerGAN (Faucci Giannelli & Zhang, 2024) trains on CaloChallenge dataset 1 pions (Krause et al., 2025) containing 533-voxel ATLAS calorimeter showers over 15 incident energies, using WGAN-GP with a dense spectral-norm critic (Miyato et al., 2018). It is the dimensionality axis, roughly two orders of magnitude above optiGAN's seven features, with 15 conditions against optiGAN's 108. We ported their model code to PyTorch and achieved their reported baselines on full-data training for an accurate comparison.

Setup. The pool is the 120,800 showers of the dataset's first pion file at 15 incident energies spaced by factors of two from 256 MeV to 4.2 TeV. Conditions are the incident energies and the subbatch size is 1. Every configuration keeps the published step count by scaling its training batch with its realized budget, so batch is a deterministic function of budget and the two cannot be told apart within this sweep. The reduced-data configurations therefore run a critic and a gradient penalty on roughly two orders of magnitude fewer rows per step than the full-data baseline does, and any quality difference the sweep shows is a joint statement about how much data was kept and how large the batch was. The membership contrast is exempt, because the membership control runs at the same budget and batch as the default configuration it modifies, so a loss there cannot be a small-batch effect.

Endpoint. Chi-squared per degree of freedom compares histograms of total deposited energy between generated and reference showers at each incident energy and pools the 15 comparisons, lower is better. For each run, the 40 checkpoints with the best in-loop value are re-measured with the generator conditioned on the reference's exact incident energies, and the one that scores best on the training file is reported on the challenge evaluation file, so the reported number never comes from the data it was selected on. The reduced-data configurations vary across repeats roughly ten times as much as full data does (Table 2), so two of them must differ by about 1.1 to 1.7 to be called apart.

Secondary endpoints. Worst-condition similarity and recall are logged in the training loop and averaged over the same 40 checkpoints without re-measurement. The tilt raises the energy configuration's worst-condition similarity by 0.020 (2.4 standard errors), and on erank it ties while lowering recall from 0.091 to 0.054. Recall falls from 0.50 on full data to between 0.05 and 0.11 on every configuration with uniform membership.

What the numbers rest on. The full-data baseline reproduces our earlier full-data measurement of the port. The in-loop and re-measured chi-squared agree to within 0.09 on average over each run's shortlist, except on the membership control, where they differ by about 1% of its far larger value. The membership contrast is budget-matched to within 4%, and the two differ in exactly one configuration field.

B.3 CTGAN ON THE KDD99 INTRUSION BENCHMARK

CTGAN (Xu et al., 2019) trains on the KDD Cup 1999 intrusion records (Stolfo & Fan, 1999) with mixed discrete and continuous columns, mode-specific normalization, conditional by default. It is the non-physics axis. Its log-frequency training-by-sampling, a published hand-designed rebalancer of

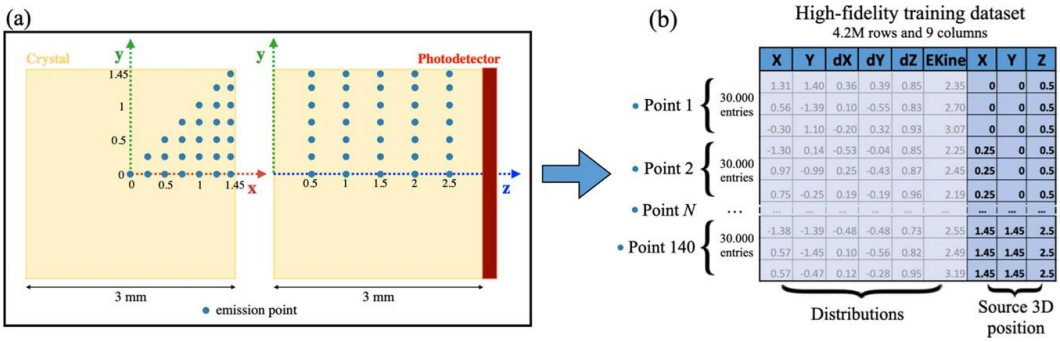

Figure 7: Adapted from Trigila et al. (2023), showing the original single-crystal dataset rather than our 3x3 array pool. (a) Emission points inside one eighth of the crystal volume, with the depth-of-interaction layout toward the photodetector. (b) The packed training matrix the critic sees, photon distributions on the left and source position on the right.

Table 3: Every contrast registered before the CaloShowerGAN sweep ran. Differences are in chi-squared per degree of freedom, first configuration minus second, so a negative difference favors the first. The interval is two standard errors of the across-repeat spread at three repeats, and a contrast is called only when it excludes zero.

Contrast	$\Delta\,\chi^2$/NDF	2 SE interval	Verdict
energy, tilt off vs. full data	$+3.33$	$[+2.33, +4.32]$	worse
energy, tilt on (default) vs. full data	$+3.83$	$[+2.83, +4.83]$	worse
erank, tilt off vs. full data	$+4.38$	$[+3.83, +4.93]$	worse
erank, tilt on vs. full data	$+5.31$	$[+3.93, +6.70]$	worse
OMP membership control vs. full data	$+24.51$	$[+20.50, +28.51]$	worse
OMP vs. random membership	$+20.68$	$[+16.55, +24.80]$	worse
tilt on vs. off, energy	$+0.50$	$[-0.90, +1.91]$	unresolved
tilt on vs. off, erank	$+0.93$	$[-0.55, +2.42]$	unresolved
energy vs. erank, tilt off	-1.05	$[-2.19, +0.08]$	unresolved
energy vs. erank, tilt on	-1.48	$[-3.19, +0.22]$	unresolved

the discrete columns, runs in every configuration, full data included, so worst-first allocation adds to it rather than replacing it.

Setup. The pool is a stratified 394,021-row split of the 494,021-row 10% file, seeded at 42, and the other 100,000 rows are held out, matching the benchmark's train and test sizes (Xu et al., 2019, Table 4). It has 41 columns, with the label collapsed to the five standard groups the benchmark table uses. Conditions are those five groups, which makes this the smallest condition count of the three domains and therefore the smallest budget: the total is a sum of per-condition ranks, so it scales with how many conditions there are. The subbatch size is 8, which is the only lever that moves the budget at five conditions. The critic runs without PacGAN packing (Lin et al., 2018) in every configuration, full data included, because at the published pac of 10 its first layer mixes ten samples and no per-sample atom exists. The CTGAN ablation scores WGAN-GP without packing slightly above the published model (Xu et al., 2019, Table 3). Every configuration holds the published batch and the published step count and cycles the subset to fill an epoch, so the full-data configuration is unchanged from our own full-data reference run and the comparison carries no batch confound.

Endpoint. A decision tree and a multilayer perceptron (MLP) are trained on synthetic rows and scored on the real held-out split, and we report the mean of their macro F1. Two of the five classes sit at ceiling and contribute almost no variance, so a difference on this endpoint is always a difference on the three rare classes. The benchmark resolves nothing finer than 0.033 macro F1, which is its own evaluation spread measured over the second half of training. Each run is pooled over its last third, and the error bar the contrasts use is the across-repeat spread at three repeats, not the much smaller pooled-evaluation spread.

Table 4: Every contrast registered before the CTGAN sweep ran, in registration order, as a difference in macro F1 with a two-standard-error interval over the three repeats. A verdict is called when the interval excludes zero and the difference clears the benchmark resolution floor of 0.033.

Contrast	Δ macro F1	2 SE interval	Verdict
energy, tilt off vs full data	$+0.0890$	$[+0.0742, +0.1037]$	higher
energy, tilt on (default) vs full data	$+0.0517$	$[+0.0326, +0.0707]$	higher
erank, tilt off vs full data	$+0.0862$	$[+0.0711, +0.1012]$	higher
erank, tilt on vs full data	$+0.0450$	$[+0.0254, +0.0646]$	higher
OMP membership control vs full data	$+0.0900$	$[+0.0632, +0.1168]$	higher
random vs OMP membership	-0.0383	$[-0.0637, -0.0129]$	lower
tilt on vs off, energy	-0.0373	$[-0.0494, -0.0252]$	lower
tilt on vs off, erank	-0.0412	$[-0.0545, -0.0279]$	lower
energy vs erank, tilt off	$+0.0028$	$[-0.0004, +0.0060]$	unresolved
energy vs erank, tilt on	$+0.0067$	$[-0.0110, +0.0244]$	unresolved

What the numbers rest on. The full-data baseline reproduces our own earlier full-data measurement on this benchmark. The allocation rule demonstrably fires, since the per-condition budget equals the per-condition rank exactly on the untilted configurations and departs from it on the tilted ones. The membership contrast is budget-matched to within 1.5%, and the two differ in exactly one configuration field.

C PER-SAMPLE GRADIENTS AND THE GATHER

The sizer needs one gradient atom per sample per selection event, and the naive way to get them is a loop over micro-batches (Yousefpour et al., 2021). That loop is the large majority of the cost of a selection event, however that cost can be all but removed entirely. A subbatch's gradient is the mean of its samples' gradients, because every loss term is a mean over the subbatch and the gradient is linear, so per-sample gradients give every atom as a segment mean. At output-layer scope, the per-sample gradient of a Wasserstein critic has a closed form in the penultimate activations. Write the output layer as $D(x) = w^\top h(x) + b$, where $h(x) \in \mathbb{R}^d$ is that vector, $w \in \mathbb{R}^d$ is the output weight and b the scalar bias, so the parameter dimension of Section 2.2 is $p = d + 1$. A Wasserstein GAN critic is trained to raise D on real samples and lower it on generated ones, giving the loss $\mathbb{E}_{\text{fake}}[D] - \mathbb{E}_{\text{real}}[D]$ plus a gradient penalty (Arjovsky et al., 2017; Gulrajani et al., 2017), and a training row's atom is the gradient of its own term in that loss. For a real row x_i that term is $L_i = -D(x_i)$, and since h does not depend on the output parameters, differentiating $L_i = -(w^\top h(x_i) + b)$ gives

$$\frac{\partial L_i}{\partial w} \;=\; -h(x_i), \qquad \frac{\partial L_i}{\partial b} \;=\; -1. \tag{5}$$

Without the penalty, row i's atom is therefore its own negated penultimate activation with a constant appended, so the whole per-condition atom matrix comes out of one forward pass and the spectrum measured in Section 2.2 is that of the critic's centered penultimate activations.

The gradient penalty (Gulrajani et al., 2017) is the term that pushes the critic toward a unit input gradient, $\rho_i = (\|v_i\| - 1)^2$, where $v_i = \nabla_x D(\tilde{x}_i)$ is the critic's gradient with respect to its own input at an interpolate $\tilde{x}_i$. With $D = w^\top h + b$ that input gradient is $v_i = J_i^\top w$ for $J_i = \partial h / \partial x$ at $\tilde{x}_i$, so the penalty's parameter gradient differentiates a gradient and is a second derivative. It also has a closed form at the output layer,

$$\frac{\partial \rho_i}{\partial w} \;=\; 2\,\frac{\|v_i\| - 1}{\|v_i\|}\,J_i v_i, \qquad \frac{\partial \rho_i}{\partial b} \;=\; 0, \tag{6}$$

the bias term vanishing because b never enters $\nabla_x D$. Evaluating it needs only $J_i v_i$, a forward-mode directional derivative that the R-operator (Pearlmutter, 1994) supplies without ever forming J_i. Row i's atom is then the gradient of $L_i + \lambda \rho_i$, with λ the penalty weight of Table 5, that is $\big(-h(x_i) + \lambda\, \partial \rho_i / \partial w,\; -1\big)$, where the interpolate pairs x_i with a sample generated under x_i's own condition. The generated term of the loss does not depend on which real row is drawn, so it is left out of the atom. Batched per-sample gradients are established (Goodfellow, 2015; Dangel et al., 2020; Yousefpour et al., 2021; Li et al., 2022), as is reducing them to the last layer (Killamsetty et al.,

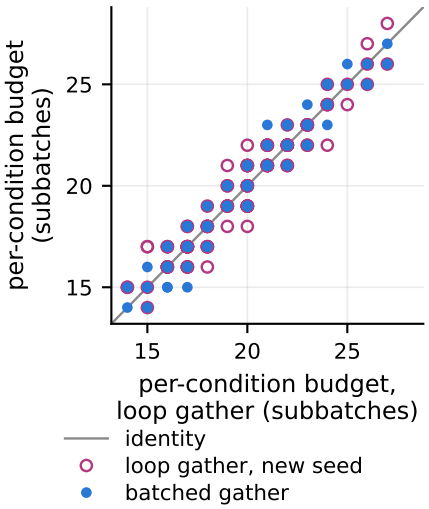

Figure 8: Per-condition budget from the batched gather against the budget from the micro-batch loop, on one selection event. Filled markers are the batched gather, open markers rerun the loop under a fresh seed, and the line is $y = x$.

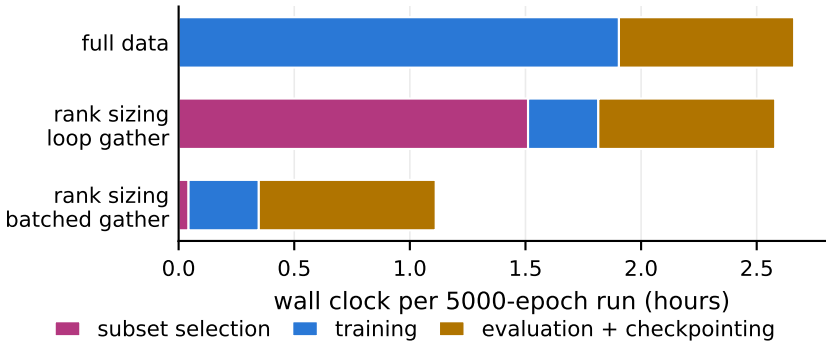

Figure 9: Where a 5000-epoch run spends its wall clock under full-data training, and under the same auto-sized method with the micro-batch loop gather and with the batched gather.

2021; Mirzasoleiman et al., 2020). Goodfellow (2015) gives the linear-layer factorization behind Equation 5, framed around gradient norms, and Li et al. (2022) extend it, but neither covers the gradient-penalty term.

Without the penalty the batched path is bit-identical to autograd, and the closed form of Equation 6 agrees with it to float32 round-off. The penalty is evaluated at points drawn between a real sample and a generated one, $\tilde{x}_i = \epsilon_i x_i + (1 - \epsilon_i) G(z_i)$ with the interpolation coefficient ϵ_i uniform on $[0, 1]$. The loop draws one ϵ per subbatch where the batched path draws one per row, so with the penalty the two paths evaluate it at different points and cannot agree exactly. Zero disagreement is therefore not the bar, since the loop does not reproduce itself across seeds either. Figure 8 plots the budget each condition receives under the batched gather against the budget it receives under the loop, so a point on the identity line $y = x$ is a condition the two gathers sized identically, with the loop rerun under a fresh seed as the control for how far apart identical code lands. The batched gather disagrees with the loop by less than the loop disagrees with itself.

On the micro-batch loop, selection is the largest single cost of a reduced-data run and gives back nearly all of the saving from reducing the data. The batched gather reduces the sizing/selection cost by a factor of ~ 30, and selection is roughly a tenth of a run (Figure 9). Substituting orthogonal matching pursuit for uniform random membership costs about 2.9x as much per event (Table 8) and lowers quality (Table 1). Batch-OMP (Rubinstein et al., 2008) precomputes the Gram matrix, which

is the standard implementation and has the same squaring hazard as computing the spectrum from the Gram (Appendix D).

D BUILDING AND READING THE SPECTRUM

Subbatch size. The subbatch size s is the lever for a pool too tall to decompose per condition at every selection event, and for a domain whose condition count alone makes the total small. CTGAN runs at $s = 8$ for both reasons, since its 394k-row pool has five conditions with most of the rows in one and its total would otherwise be a sum of five counts. The estimate itself is invariant to s while A_c stays taller than it is wide, because random grouping, unlike a structure-aware one, is independent of the gradient geometry, and averaging scales every direction alike. A subbatch large enough to push a condition's atom count below the parameter dimension caps its rank at that atom count, so the subbatch size rather than the spectrum then sets the budget.

Why the atom matrix is centered. Subtracting the mean atom is a rank-one update, so it can lower the algebraic rank by at most one. It changes where the energy sits. Interlacing (Honeine, 2014; Cadima & Jolliffe, 2009) gives $\sigma_{i+1}(A_c) \leq \sigma_i(\tilde{A}_c)$, so the centered spectrum sits at or above the uncentered one shifted up by one index, and on our atom matrices the count at a fixed θ rises accordingly. Uncentered, σ_1 is the mean direction and carries almost all of the squared energy, so Equation 2 measures the mean instead of the spread around it. Every atom carries that mean direction, so it is not a direction additional samples have to supply.

Why an SVD of the atom matrix and not an eigendecomposition of its Gram. $\kappa(A^\top A) = \kappa(A)^2$ (Golub & Van Loan, 2013), so forming the Gram squares the conditioning a singular value decomposition (SVD) would have kept, on matrices that are badly rank deficient by construction. In float32 this squaring produced a spurious OMP residual floor, and it is the same hazard as the Batch-OMP implementation in Appendix C.

Why float64. The OMP residual floor that rises with training is a round-off artifact of a rank-deficient Gram solve in float32, not a property of the data, since the same atoms refit in float64 give a floor that falls monotonically instead. A residual-elbow stopping rule computed the same way in float32 would read the artifact rather than the data, so we report it as a standalone finding. The spectrum of Section 2.2 is taken in float64 unconditionally for the same reason.

Why not the Gavish-Donoho threshold. The Gavish-Donoho optimal hard threshold (Gavish & Donoho, 2014) fails here because it assumes the non-signal part of the spectrum is an iid Marchenko-Pastur bulk (Marchenko & Pastur, 1967), and a matrix of gradient atoms has no such bulk. It lands between the two thresholds we trained (Figure 10).

E HYPERPARAMETERS AND SEEDS

Every configuration in a domain shares the settings of Table 5 except the selection fields its sweep varies (Appendix F). Per-sample gradients come from the critic's output layer in float32, and every decomposition runs in float64 (Appendix D). optiGAN follows Trigila et al. (2023) at base width 128, with a seven-dimensional noise input and a generator EMA of 0.999. CaloShowerGAN follows Faucci Giannelli & Zhang (2024) with a 200-400-800 generator, an 800-400-200 critic and a 200-dimensional latent, and CTGAN follows Xu et al. (2019) with two 256-unit layers in each network, a 128-dimensional noise input and packing off. On optiGAN each held-out evaluation draws 80,000 samples globally and 5,000 per condition, and uses 250 random projections for the sliced distance and $k = 5$ neighbors for precision, recall and realism. Each run seeds Python, NumPy and PyTorch with the SHA-256 hash of the base seed 42, its configuration tag and its repeat index, and derives the seeds of its membership draws and data-loader order the same way. GPU kernels run in PyTorch's default non-deterministic mode, so a rerun of a configuration reproduces the spread across its repeats rather than any single run.

Table 5: Training and selection settings of each domain's default configuration. A CaloShowerGAN epoch is 1000 iterations.

	optiGAN	CaloShowerGAN	CTGAN
Pool rows	513,042	120,800	394,021
Training conditions	108	15	5
Full-data batch	90,000	1024	500
Epochs	5000	1000	300
Learning rate	5×10^{-5}	10^{-4}	2×10^{-4}
Adam (β_1, β_2)	(0.5, 0.9)	(0.5, 0.999)	(0.5, 0.9)
Critic steps per generator step	3	5	1
Gradient-penalty weight	10	20	10
Selection interval (epochs)	100	50	15
Re-sizing interval (epochs)	500	50	15
Energy threshold θ	0.999	0.999	0.999
Rank multiple α	1	1	1
Subbatch size s	1	1	8
Tilt	on	on	on
Budget cap (% of pool)	25	none	none
Reduced-data batch	scaled	scaled	full-data
Evaluation interval (epochs)	100	1	10

F SWEEP DEFINITIONS, AGGREGATION AND TIMING

The first optiGAN sweep crosses the energy criterion at $\theta = 0.999$ and $\theta = 0.9999$ and erank with the tilt on and off, re-sizing at every selection event. The second repeats the tilted energy configuration at $\theta = 0.999$ and changes one field at a time, the resize interval, the membership rule or the rank multiple. The batch-scaling sweep runs the tilted erank configuration with its batch scaled to the budget, at full data's batch for one pass over the subset per epoch, and at full data's batch and step count with the subset cycled. Its six full-data runs are the baseline of every optiGAN comparison. The pool and cap sweeps run the three estimators with the tilt off, on a pool five times optiGAN's with a 25% cap and on optiGAN's own pool with a 0.5% cap. The cross-domain sweeps run both estimators with the tilt on and off and the OMP membership control, re-sizing at every selection event (Table 6). Figure 10 comes from a separate probe of 48 condition-by-repeat cells at epoch 5000. The membership comparisons of Appendix A come from two earlier 2000-epoch optiGAN sweeps at fixed budgets, difficulty scoring at 1% and 3% of the pool, and coverage and gradient matching at 1% and 5%. Each sweep is read at the last epoch all of its runs reached, 2000 and 1500 respectively.

On optiGAN the worst-quartile endpoint is a run's mean similarity over the worst quarter of the 846 held-out conditions at its final evaluation, and the mean, median and single-worst condition come from the same per-condition values. Worst-condition similarity is the single-worst condition's value in every domain (Figure 1 and Appendix A). Global and p99 similarity, precision, recall and realism are each run's mean over its last five held-out evaluations, from epoch 4600 to epoch 5000. A contrast's interval is a two-sample bootstrap of the difference in means over repeats with 10,000 resamples, and a configuration sits above or below a reference when the 95% interval excludes zero. Equivalence is a two one-sided test (TOST) at the 5% level with a margin of twice full data's across-repeat standard deviation on the same endpoint, 0.0138 on the worst quartile. Budgets in Table 1 are the median realized share of the pool over the final 1000 epochs. On CaloShowerGAN and CTGAN a contrast's interval is two across-repeat standard errors of the difference at three repeats. A verdict needs the interval to exclude zero and, on CTGAN, the difference to clear the resolution floor of 0.033 macro F1. CaloShowerGAN's chi-squared is read at a checkpoint selected on the training file, and CTGAN's macro F1 is pooled over the last third of training (Appendix B).

Timings come from one workstation with two NVIDIA RTX 4080 SUPER GPUs of 16 GB, an AMD Ryzen Threadripper 7960X CPU, PyTorch 2.10.0 and CUDA 12.9. Each timed optiGAN sweep ran its runs one at a time on one GPU with no other job queued on the machine, since a job on the second card contends for the same CPU cores and host memory bandwidth. The data path is tuned identically for every configuration before anything is timed, since the per-row fetch cost scales with batch size and our batch scales with the budget. An untuned full-data baseline would manufacture a speedup for

Table 6: The sweeps behind each reported result. Configuration counts exclude full data.

Sweep	Configurations	Repeats	Epochs	Reported in
optiGAN, first	6	6	5000	Table 1
optiGAN, second	5	6	5000	Tables 1, 7, 8, Figure 1
optiGAN, batch scaling	3	6	5000	full-data baseline, Section 4
optiGAN, pool and cap	6	5	2000	Figure 11
CaloShowerGAN	5	3	1000	Tables 2, 3, Figure 2a
CTGAN	5	3	300	Tables 2, 4, Figure 2b
optiGAN, difficulty scoring	10	3–5	2000	Figures 3, 4, 6
optiGAN, coverage matching	9	3	2000	Figure 5

Table 7: Full data's value on each endpoint, and the difference from it as the rank multiple α of FLARE (resize 100) is raised, six repeats each, higher is better. A dagger marks TOST equivalence with full data.

Endpoint	Full data (value)	1.16% ($\alpha = 1$)	2.25% ($\alpha = 2$)	4.62% ($\alpha = 4$)
Worst quartile	0.7844	$+0.0042^{\dagger}$	$+0.0056^{\dagger}$	$+0.0166$
Median condition	0.9009	-0.0110	-0.0081	-0.0065
Mean condition	0.8788	-0.0082	-0.0063	$-0.0022^{\dagger}$
Global	0.9764	$-0.0012^{\dagger}$	$+0.0008^{\dagger}$	$+0.0022$
Global, p99	0.9514	$+0.0010^{\dagger}$	$+0.0048$	$+0.0055$
Precision	0.7917	-0.0174	-0.0111	-0.0078
Recall	0.8398	-0.0163	-0.0162	-0.0148
Realism	1.3148	-0.0256	-0.0171	-0.0143

free, and we found and removed that artifact in our own harness. The latency probe of Section 4.1 times training iterations at a fixed budget against full data, on one GPU and on the CPU, outside the sweeps. Two evaluation-cadence changes landed during the project and are not bit-neutral, so no reported configuration mixes repeats from before and after them.

Table 8: Mean hours per run by phase over six repeats, with the speedup over full data on training plus selection and end to end. Other is wall clock outside the three phase timers, so each row sums to a run's wall clock.

	Hours				Speedup	
Configuration	Train	Select	Eval	Other	Train+sel	End-to-end
FLARE (resize 100)	0.067	0.047	0.270	0.049	13.0x	4.1x
FLARE (resize 500)	0.067	0.010	0.270	0.035	19.1x	4.7x
OMP membership control	0.071	0.138	0.272	0.053	7.1x	3.4x
Full data	1.486	—	0.270	0.037	1.0x	1.0x

G ADDITIONAL FIGURES

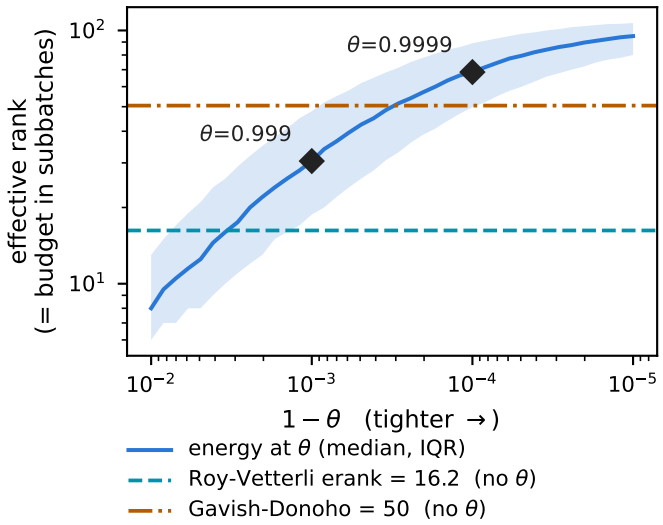

Figure 10: Effective rank, the per-condition budget in atoms, against $1-\theta$ over 48 condition-by-repeat cells, median with the interquartile band. Diamonds mark the two thresholds we trained, and the two threshold-free estimators are horizontal lines.

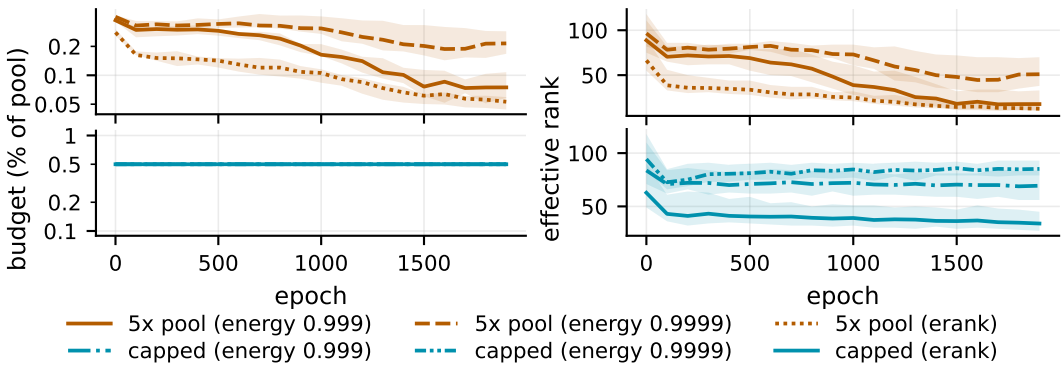

Figure 11: Budget demand over training for three estimators on two sweeps, median over five repeats with the across-repeat range shaded. The top row uses a 2.56M-row pool, five times optiGAN's, with a 25% cap it never reaches, and the bottom row optiGAN's own pool with a 0.5% cap below every estimator's demand. The left column is the realized budget as a percentage of that row's pool and the right column the per-condition effective rank that sets it.

H FURTHER RELATED WORK

Importance sampling and class reweighting. Non-uniform sampling reduces gradient variance at a fixed iteration count (Needell et al., 2014; Zhao & Zhang, 2015), and Katharopoulos & Fleuret (2018) apply it per example through a bound on the per-sample gradient norm. Used to choose samples rather than to weight them, that bound is a difficulty score in the sense of Paul et al. (2021). Worst-first allocation changes how many samples a condition contributes rather than the probability of drawing a fixed number, and membership within a condition stays uniform. Cui et al. (2019) share the words "effective number of samples" but reweight a loss by a closed form in the class count, where we set a count from a spectrum.

GAN sample complexity and generalization. Arora et al. (2017) show that the neural-net distance a critic measures generalizes once the sample is large relative to the critic's parameter count, and Ji & Liang (2018) give matching minimax bounds on estimating it that fall as the inverse square root of the sample size. Both hold uniformly over a class of critics, while FLARE reads a budget off one trained critic's gradients. Data-efficient GAN training addresses scarce data rather than training cost, augmenting what the discriminator sees so it does not overfit (Karras et al., 2020).