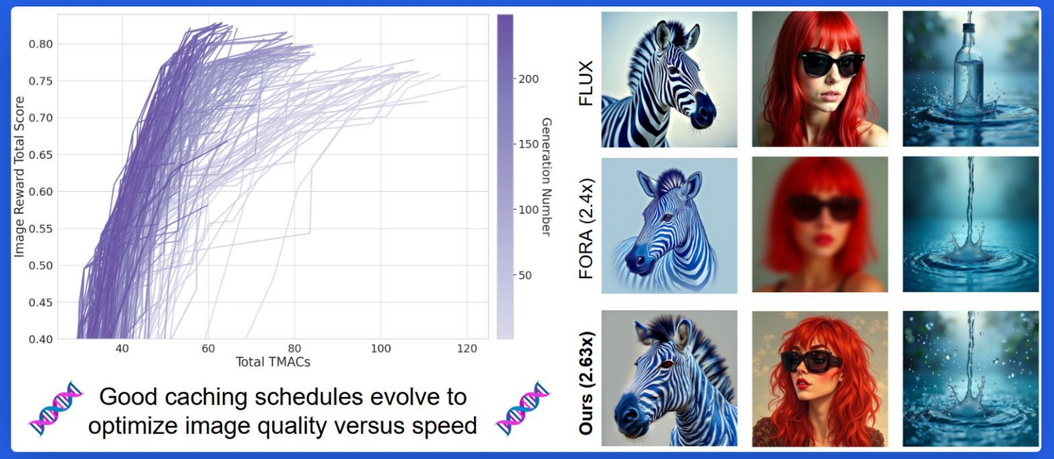
ICLR 2026 CONFERENCE PAPER

Evolutionary Caching to Accelerate Your Off-the-Shelf Diffusion Model

A genetic algorithm discovers optimal caching schedules — no retraining, no reference images, no architectural changes.

Anirud Aggarwal · Abhinav Shrivastava · Matthew Gwilliam

University of Maryland, College Park



Project Page

https://research.aniaggarwal.com/ecad

Caching as Pareto Optimization

THE PROBLEM

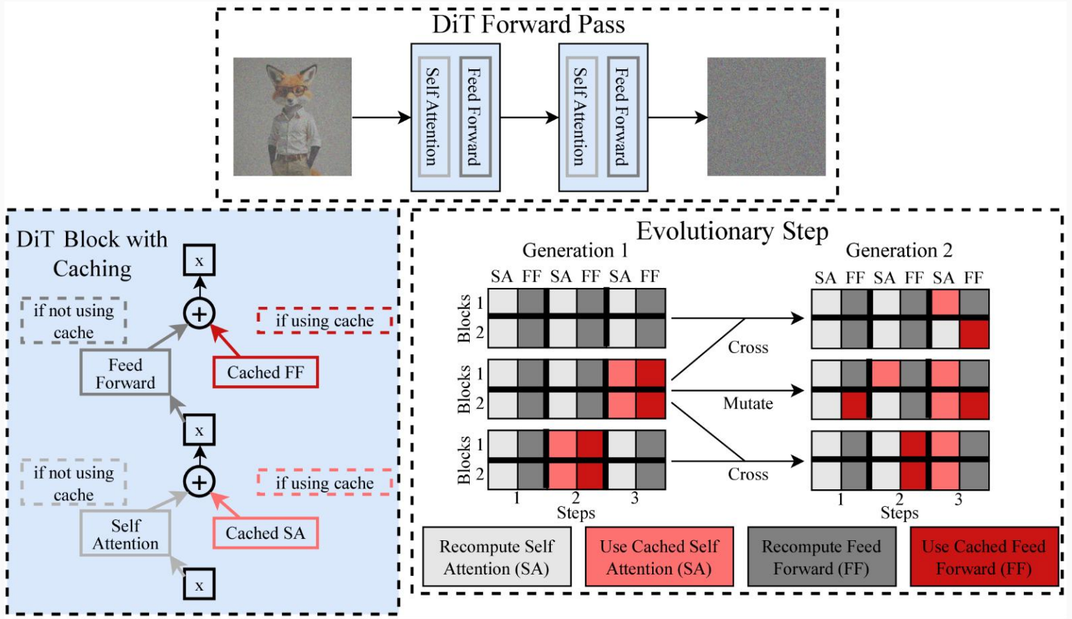
- Diffusion models require **20–50 denoising steps**, each a full transformer forward pass
- Feature caching reuses intermediate outputs to skip computation
- Prior methods use **rigid heuristics** — limited speedup, poor generalization
- Search space is vast: $>2^{1680}$ configurations for a 28-block DiT

KEY INSIGHT

> **Formulate caching as multi-objective optimization:** minimize compute (MACs) *and* maximize quality (Image Reward) simultaneously.

THE RESULT

- A caching schedule is a binary tensor $S \in \{0,1\}^{N \times B \times C}$ encoding *when*, *where*, and *what* to cache
- ECAD evolves a **Pareto frontier** of schedules — a full spectrum of quality-speed trade-offs, not just isolated operating points



Left: Component-level caching in DiT blocks. **Right:** Crossover and mutation evolve caching schedules.

ECAD: The Method

OPTIMIZATION LOOP

- Generate images using each candidate schedule on **100 calibration prompts**
- Evaluate quality (Image Reward) and compute cost (MACs)
- Evolve schedules via **selection, crossover, and mutation**
- Repeat for G generations — Pareto frontier improves each generation

MULTI-OBJECTIVE OPTIMIZATION WITH 4 COMPONENTS

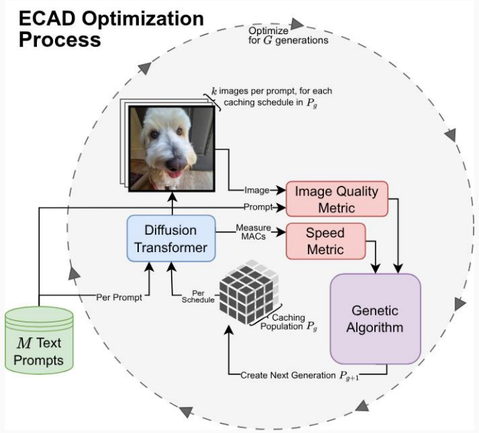
BINARY CACHING TENSOR
Defines which blocks, steps, and components are cacheable

QUALITY METRIC
Image Reward (or any metric). Bring your own reward function

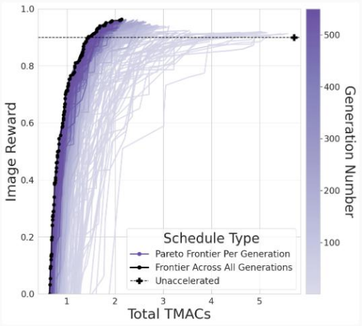
COST METRIC
TMACs (total multiply-accumulate operations) — measures computational cost per schedule

CALIBRATION PROMPTS & INITIAL POPULATION
Just 100 text prompts — no image data needed. Population seeded with heuristic-based schedules or random initialization.

Training-free: no gradients, no weight updates, no memory overhead. Runs asynchronously on consumer GPUs.



ECAD optimization loop. Text prompts drive image generation per schedule; NSGA-II evolves the population toward the Pareto frontier.



Evolution over generations. Competitive schedules emerge within ~50 generations. The Pareto frontier improves steadily with continued optimization.

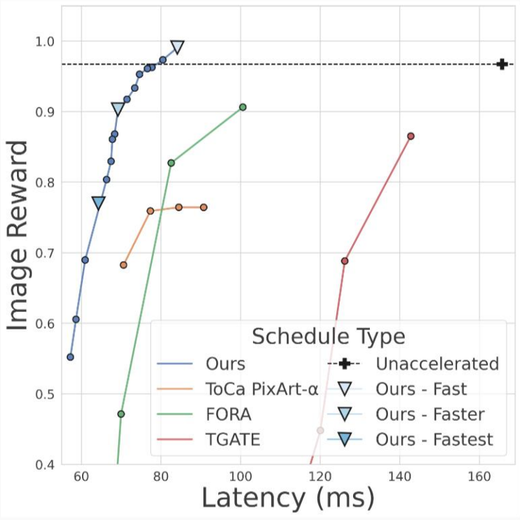
Results: New State of the Art

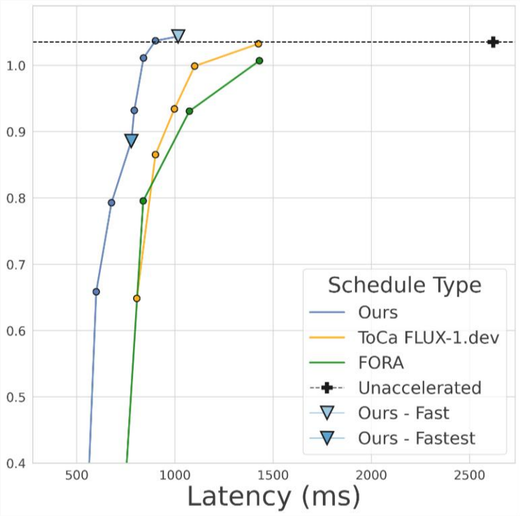
PixArt-α: improves COCO FID by **4.47** over prior SOTA while increasing speedup from 2.35× to **2.58×**

FLUX.1-dev: matches baseline quality at **2.58×** speedup; competitive quality at **3.37×**

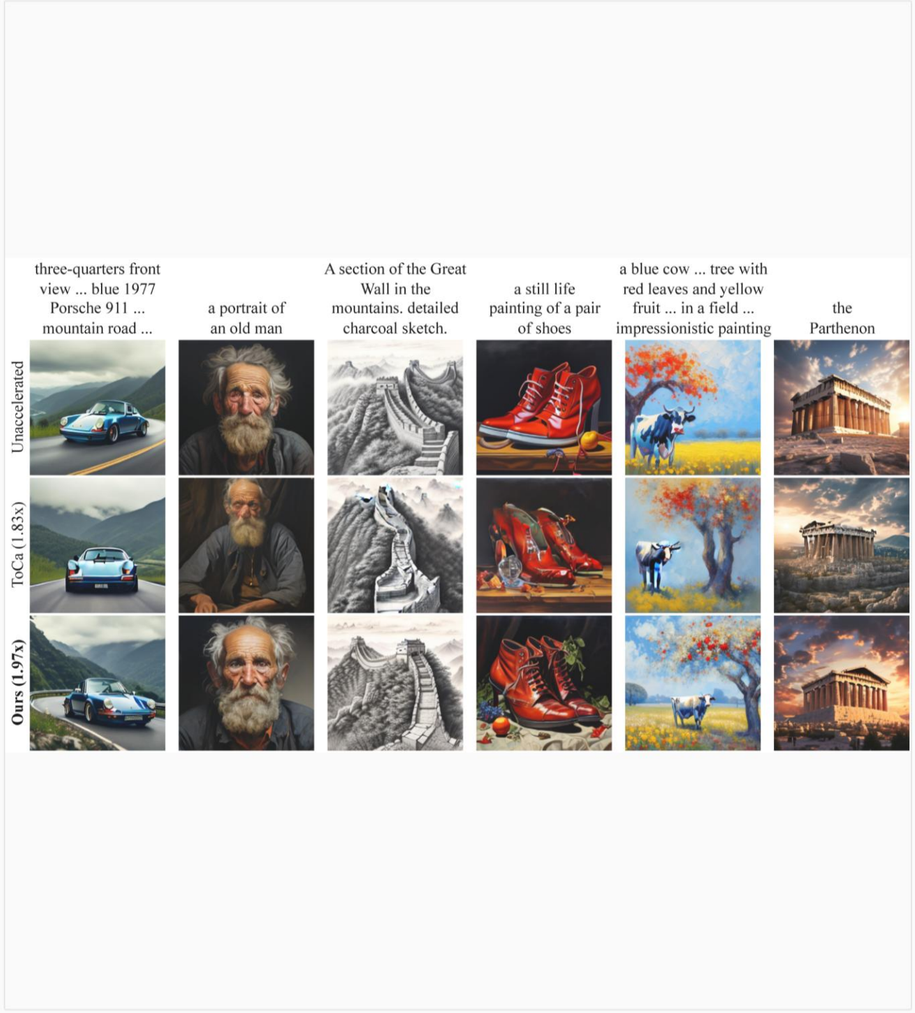
ECAD produces an **entire frontier**, allowing choice of quality-speed trade-off at deployment

PARETO FRONTIERS



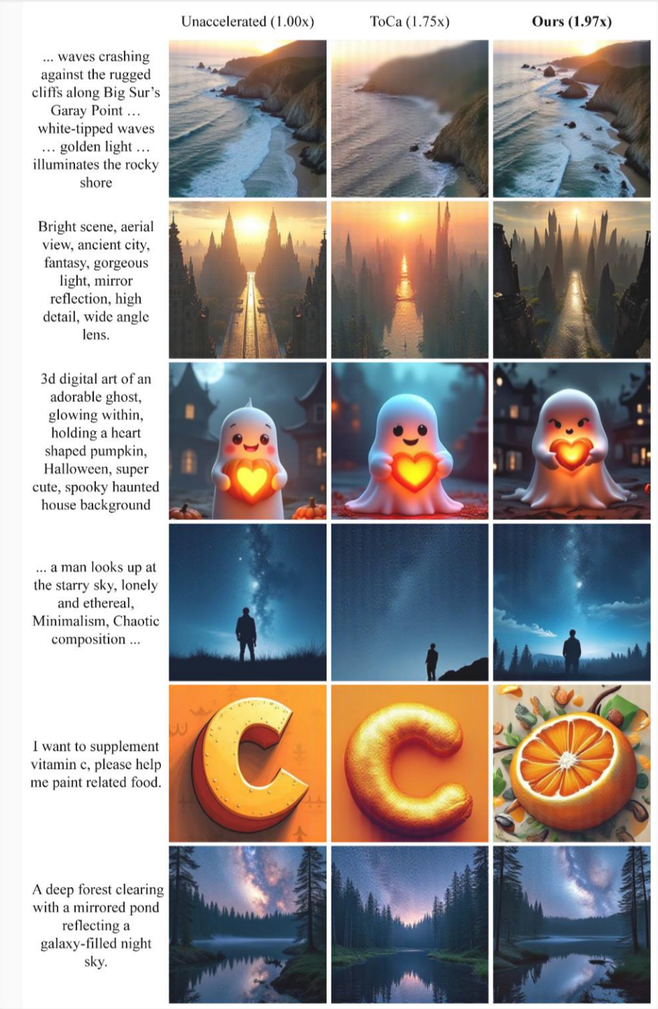


QUALITATIVE — PIXART-A



Top: unaccelerated. Mid: ToCa (1.83×). Bottom: **Ours (1.97×)**.

QUALITATIVE — FLUX.1-DEV



Left: unaccelerated. Mid: ToCa (1.75×). Right: **Ours (1.97×)**.

Generalization & Flexibility

EMERGENT GENERALIZATION

RESOLUTION TRANSFER
Schedules optimized at 256^2 transfer to **1024^2** without re-optimization. FLUX fast schedule: **2.63×** at 1024^2.

MODEL TRANSFER
PixArt-α → PixArt-Σ: direct transfer 1.76×; +50 generations → **1.98×** with better FID.

STEP COUNT TRANSFER
Schedules at 10 steps can be upscaled to 20 steps and vice versa.

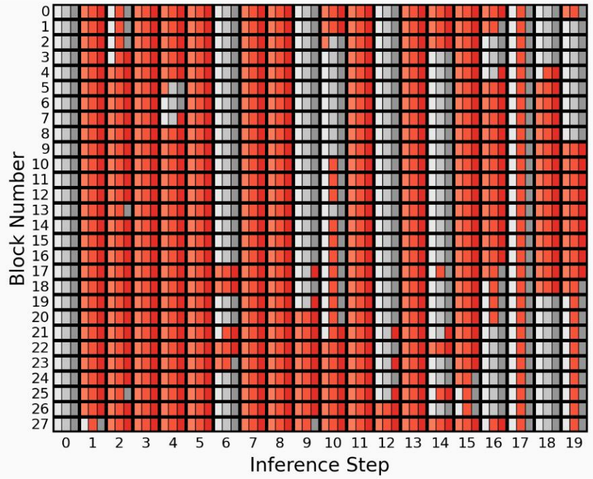
PROMPT ROBUSTNESS
ChatGPT-generated, domain-specific, even **5-word prompts** all yield competitive schedules.

A BRING-YOUR-OWN FRAMEWORK

- **Metrics:** Image Reward, CLIP Score + CLIP IQA, VisionReward, or human ratings
- **Prompts:** ~100 prompts of any source. Quantity matters more than specificity
- **Compute:** competitive schedules emerge in ~50 generations; optimized in ~44 GPU-hours

ECAD is **orthogonal** to distillation, quantization, pruning, and torch.compile — combine freely for further speedup.

LEARNED SCHEDULE — PIXART-A "FAST"



Rows: blocks 0–27, each with 3 sub-rows (self-attn, cross-attn, FFN). Columns: inference steps 0–19.
Red = cached (reuse previous output). **Gray = recomputed** (fresh computation).
Structured caching patterns **emerge from optimization**, not manual design. Early steps recompute more; later steps cache aggressively.

Conclusion

Reconceptualize diffusion caching as **Pareto optimization over binary schedules.** SOTA training-free acceleration with just 100 text prompts.

QUALITY-SPEED SOTA
PixArt-α: **2.58×** speedup, **+4.47 FID** over previous SOTA. FLUX.1-dev: up to **3.37×** with competitive quality.

FLEXIBLE FRAMEWORK
Choose *what* to cache, *how* to measure quality, and *which* model to target. Composes with quantization, distillation, and pruning.

GENERALIZES
Across resolutions, model variants, and calibration sets — no re-optimization needed

LIGHTWEIGHT
No gradients, no weight updates. Works on consumer GPUs. One-time cost, reusable results

Thank you!

Paper · Code · Interactive Demo
https://research.aniaggarwal.com/ecad

Scan for project page