Trajectory Commitments

Cheap Verifiable Inference for Diffusion and Codec Models

Andrew Lee
andrew@joseon.com

Abstract

A provider paid to run a 3.5-billion-parameter diffusion model has an obvious move: run a smaller one, or fewer denoising steps, and keep the difference. The consumer receives a plausible waveform or image and cannot tell. This is the incentive that motivates verifiable inference for language models, yet the language-model defenses do not transfer, for a structural reason. They commit a per-token output distribution and spot-check it; a diffusion model produces no such distribution. It integrates an ODE/SDE from noise over N steps, and the only thing the consumer sees is the last latent.

We give a verification primitive for this setting and measure it. The provider commits a *trajectory*: a Merkle root over (step index, latent digest) at sampled steps plus the final latent. A verifier re-runs *one* reference denoising step from a committed latent z_i and checks $\hat{z}_{i+1} \approx z_{i+1}$ within a tolerance, at cost $\rho \approx 1/N$ of the generation. On an honest re-run the step reproduces *exactly* (rel-$L_2 = 0$); a substituted computation diverges far outside any reasonable tolerance. We implemented this on three independent engines and modalities (a 3.5B diffusion-transformer audio flow model, a 1.5B Euler latent-diffusion image model, and a latent video-diffusion model), and the primitive holds on all three. We are precise about the one thing it does not give for free: the accept tolerance is a *measured* quantity, set from the honest cross-hardware reproduction tail rather than a proven constant, and we say exactly where that leaves the guarantee.

1 The problem, and why the obvious fix does not apply

Outsourced inference has a verification problem that is really an incentive problem. The party running the model is paid per request and can cut its own cost by serving something cheaper than advertised: a distilled or quantized model, a smaller one, or, for an iterative model, fewer sampling steps. The output still looks like an output. Nothing in a returned image or audio clip reveals that it was produced by a 1.5B model billed as a 7B one, or by 8 denoising steps billed as 40.

For autoregressive language models there is now a line of cheap defenses (Ong et al., 2025; Inference.net, 2025; Arun et al., 2025; Karvonen et al., 2025; Wang et al., 2025). They share a shape. Generating token t produces a probability distribution over the vocabulary; that distribution is a fingerprint of the exact computation, because re-running the model on the same prefix yields the same distribution (up to hardware noise). So the provider commits something derived from the per-token distributions (top-k log-probabilities (Inference.net, 2025), or a sketch of the hidden state that produced them (Ong et al., 2025)), and a verifier re-derives it on a sample of positions, cheaply, by a single teacher-forced pass rather than re-generation. The cost of checking is a small fraction ρ of the cost of producing.

Why this does not transfer. A diffusion or flow model (Ho et al., 2020; Song et al., 2021; Lipman et al., 2023) hands you nothing of that kind. It starts from Gaussian noise z_N and integrates

a learned velocity/score field for N steps down to z_0, then decodes z_0 to pixels or audio. There is no per-token distribution. There is no "position" to teacher-force. The only natural observable is the final latent, and committing only the final latent is useless: a provider that ran a cheaper model simply commits whatever it produced, and the commitment certifies nothing about *how* it was produced. The honest defenses for text are not strong enough here, and they are not weak; they are inapplicable. The first question is not "how do we verify" but "what is there to check."

The answer is the trajectory. A diffusion model's computation is not a single output; it is a sequence $z_N \to z_{N-1} \to \cdots \to z_0$, and each transition $z_{i+1} \to z_i$ is one forward evaluation of the same network the provider claims to be running. That sequence is checkable in exactly the way a single output is not.

It is worth stating the principle plainly, because it is more general than diffusion. An output is not evidence of the computation that produced it; a process is. An autoregressive model is checkable because it exposes its process one distribution at a time. A diffusion model looked uncheckable only because we kept looking at its output, the final latent, and an output is precisely the thing a cheaper computation can forge. Its process was never hidden: N deterministic steps, each a forward evaluation of the network the provider claims to run. Commit the process instead of the output and the iterative structure inverts from a liability into the strongest guarantee available: the model does the same checkable thing N times, and any one of those steps stands witness for the rest. The cost asymmetry $\rho \approx 1/N$ that follows is not a trick; it is a restatement of that fact.

Contributions.

1. A verification primitive for diffusion and codec inference (a *trajectory commitment* with a *single-step re-check*) that detects a substituted computation at cost $\rho \approx 1/N$, where prior cheap-verification work covers only autoregressive text (§3).
2. An implementation and measurement on *three independent engines and modalities* (audio, image, video), each instrumented at the denoising loop, showing exact honest reproduction and clean separation of a cheat (§5).
3. A precise account of the sampler-state subtlety (guidance with cross-step momentum is not single-step-reproducible inside the guidance window, and the verifier must select steps accordingly), and of the tolerance as a *measured cross-hardware* quantity rather than a proven constant (§4, §6).

2 Threat model

The prover is rational and economically motivated, not arbitrarily malicious: it deviates only to lower its compute cost, and only in ways a consumer cannot detect from the output alone. Concretely it may substitute a smaller or more-aggressively-quantized model, run fewer sampling steps, or skip the conditioning it was paid to apply. It controls the bytes it returns and the commitment it publishes; it does not control the verifier's reference model or the public randomness that selects what to check.

The verifier holds the same model weights the provider *claims* to run (this is the declared-model setting; binding the declared model to a measured identity is a separate problem we do not solve here) together with a reference implementation of the sampler. It wants: if the prover deviated, to reject with probability at least s; if the prover was honest, to accept except with a small false-rejection rate ε; and to spend a small fraction $\rho \ll 1$ of the generation cost doing so. The asymmetry $\rho \ll 1$ is the entire point. A verifier that re-runs the whole trajectory has caught everything and saved no one anything; it is just a second provider.

We make no cryptographic assumption beyond a collision-resistant hash for the commitment (Merkle, 1988). The security of the check is statistical and computational: a prover that did not perform a step cannot produce its output, and cannot find a different cheap computation that lands on the committed latent, for the same reason it could not have produced that latent in the first place.

3 The trajectory commitment

Commit. Fix the request seed (so the initial noise z_N and the sampler schedule are public functions of the request, and the honest trajectory is reproducible). As it samples, the provider records the latent at a set of step indices $S \subseteq \{0, \dots, N\}$ (in the simplest case all of them) and publishes

$$\text{root} \;=\; \text{MerkleRoot}\big(\{(i,\ \text{H}(z_i)) : i \in S\} \cup \{(0, \text{H}(z_0))\}\big),$$

a Merkle root over (step index, latent digest) pairs at the sampled steps and the final latent. The root is small and is signed into the served record alongside the output. Publishing digests, not latents, keeps the commitment cheap and leaks nothing about the content.

Check. The verifier draws a step i from public randomness, asks the provider to open the committed z_i and z_{i+1} (with their Merkle paths), confirms both hash to the committed root, and then does the one expensive thing: it runs *one* reference denoising step,

$$\hat{z}_{i+1} \;=\; \text{Step}_\theta(z_i,\ i), \qquad \text{and accepts iff} \qquad \frac{\|\hat{z}_{i+1} - z_{i+1}\|_2}{\|z_{i+1}\|_2} \;\le\; \tau.$$

One forward evaluation of the network, against the N the provider performed: cost $\rho \approx 1/N$. Sampling k steps instead of one raises the cost to k/N and the per-request soundness toward $1 - (1 - s_1)^k$; the economics in §5 are reported for a single step.

Why it binds. The Merkle inclusion check runs *before* the tolerance check, and this ordering is the load-bearing part. It means the verifier never scores a step against a latent the provider did not commit: a prover cannot serve one trajectory, commit another, and reveal whichever is convenient when challenged. So the only question left is whether the committed transition $z_i \to z_{i+1}$ is one the claimed model actually produces. If the provider ran a different (cheaper) model, the committed z_{i+1} is that model's output, and the reference Step_θ does not reproduce it: the two networks disagree at that step, and the disagreement shows up in rel-L_2. If the provider ran fewer steps, the committed trajectory is short or its digests do not correspond to the declared schedule, and the opened pair fails either inclusion or the step relation. The provider's only escape is to find a cheap computation whose every committed step happens to match the reference model's step, which is to say, to cheaply reproduce the reference model, which is the thing it was trying to avoid.

4 Soundness, and the honest subtleties

Three things have to be true for the check to be sound, and we treat each as a measured claim, not an assumption.

(1) An honest step reproduces; a dishonest one does not. This is the empirical heart of the method and is the subject of §5. On the same build, an honest single step reproduces the committed next latent to rel-$L_2 = 0$: the computation is deterministic given $(z_i, i, \theta, \text{seed})$. A substituted computation (a perturbed conditioning, a different model, an unrelated step) lands far away, rel-L_2 from 0.27 up to 1.0 in our measurements. The gap between 0 and "far" is what a tolerance τ exploits.

(2) The sampler-state subtlety (guidance momentum). A single step is only single-step-reproducible if it is a function of the *committed* state alone. Plain Euler and DDPM steps are. But guidance schemes with cross-step momentum, such as adaptive projected guidance and its relatives (Ho and Salimans, 2022; Sadat et al., 2025), carry a running term between steps, so reproducing step i from z_i alone is not exact inside the guidance window; it requires the committed sampler state too. We handle this without committing extra state: the guidance momentum is active only on a known sub-interval of the schedule, and the verifier *selects its re-check step from outside that window*, where the step is a pure function of (z_i, i) and reproduces exactly. This is a real restriction on *which* steps are checkable, not on whether the method works (the checkable steps are a known, large fraction of the schedule), and it is the kind of detail that decides whether a verifier false-rejects honest providers in practice.

(3) The tolerance is cross-hardware, and that is where the honesty lives. The rel-$L_2 = 0$ above is a *same-build* number: identical GPU, identical kernels, identical accumulation order. A different GPU, a different attention kernel, or a different reduction order reproduces a latent to a small *nonzero* drift. So the deployed accept threshold τ is not 0; it is set from the measured honest cross-backend drift tail, and the soundness claim is exactly this: τ separates the honest drift tail from the cheat divergence. For diffusion this separation is comfortable, because the honest tail is small while the cheat divergence is 0.27–1.0, two to three orders of magnitude larger. That is precisely why diffusion is a *cleaner* verification target than the marginal near-lossless-quantization band that troubles text verification. We return to the limits of this in §6.

5 Implementation and measurement

We instrumented the denoising loop of three independent engines to emit the per-step latent, and measured both honest reproduction and the response to a substituted computation. None of the three shares code with the others; the primitive is not an artifact of one sampler.

Audio (a 3.5B diffusion-transformer flow model). Instrumenting the sampler of our own ggml audio engine, an honest re-run reproduces the trajectory *exactly*: per-step rel-$L_2 = 0$, so the same-build $\varepsilon = 0$. A small (5%) perturbation of the conditioning, standing in for a substituted computation, diverges progressively through the trajectory to rel-$L_2 = 0.27$, cleanly separable at any reasonable tolerance. We then implemented the single-step re-check itself: from a committed latent z_i, one reference step reproduces z_{i+1} to rel-$L_2 = 0$, against rel-$L_2 = 0.52$ for an *unrelated* step. That confirms the $\rho \approx 1/N$ economics on the real engine: the verifier re-runs one of N steps, not the trajectory.

Image (a 1.5B Euler latent-diffusion model). On an independent engine (a vendored Stable Diffusion implementation (Rombach et al., 2022) with an Euler sampler (Karras et al., 2022)), an

Engine (independent)	Modality	honest rel-L_2	cheat rel-L_2	cheat type
3.5B diffusion-transformer (flow)	audio	0	0.27	5% conditioning perturb.
single-step re-check	audio	0	0.52	unrelated step
final-decode	audio	0	changed	2% latent perturb.
1.5B latent diffusion (Euler)	image	0	1.0	changed prompt
Wan latent video diffusion	video	0	rejected	fabricated step

Table 1: Same-build measurements across three independent engines. Honest re-runs reproduce each committed step exactly; substituted computations diverge by 0.27–1.0, well outside any reasonable tolerance. Deployed thresholds are set from the honest *cross-backend* drift tail (§6), not the same-build 0 shown here.

honest re-run again reproduces the trajectory exactly (rel-$L_2 = 0$ per step), while changing the prompt diverges progressively to rel-$L_2 = 1.0$.

Video (a latent video-diffusion model). A third engine, a Wan latent video-diffusion model running on Apple Silicon, behaves identically: an honest re-run reproduces each committed step exactly, and a fabricated step is rejected.

The final-decode check. Committing the trajectory binds the *computation*; committing the final latent z_0 binds the *output*. We measured the decode: decoding a committed z_0 is deterministic and reproduces exactly (rel-$L_2 = 0$), while a 2% perturbation of z_0 changes the decoded audio. So committing z_0 ties the served bytes to the verified trajectory: a provider cannot run the honest trajectory and then return a different output.

Adjudication. The full check is implemented, not just the distances: the verifier's single-step re-check is bound to the committed trajectory root by Merkle inclusion before the tolerance is applied, so a step scored against an uncommitted latent is rejected as tampered rather than silently accepted. The audit loop runs end to end in the reference system.

6 Limitations

We are explicit about the boundary of the claim.

The tolerance is measured, not proven. The clean separation above is same-build. The deployed guarantee depends on the honest cross-hardware drift tail sitting strictly below the cheat divergence at the chosen τ. For diffusion the margin is large (honest tail small; cheat 0.27–1.0), so we expect this to hold comfortably across hardware, but we report it as a measured expectation, not a theorem. A fully adversarial study would characterize the worst-case honest drift across the GPU/kernel matrix; we have not done that here.

Declared-model setting. The verifier checks that the trajectory is consistent with *the weights it was told to expect*. Binding those weights to a measured hardware/software identity (so a provider cannot lie about which model it declared) is a separate mechanism, orthogonal to this one.

Step coverage under momentum guidance. As in §4, samplers with cross-step momentum restrict the single-step-checkable steps to those outside the guidance window. This shrinks the pool of checkable steps; it does not weaken the check on the steps that are checkable, and the verifier's step selection must respect it or it will false-reject honest providers.

What is covered. The primitive covers continuous-latent iterative generators (diffusion, flow, codec decode). Non-autoregressive *text* diffusion, meaning models that denoise a discrete token canvas in parallel, emits neither a left-to-right distribution nor a continuous latent, and would need a hybrid commitment (a token-trajectory) we sketch but have not instrumented.

7 Related work

Cheap verification of outsourced inference for autoregressive language models is recent and active. Ong et al. (2025) commits a locality-sensitive sketch of the hidden state at sampled positions; Inference.net (2025) commits top-k log-probabilities and checks them by a teacher-forced pass; Arun et al. (2025) resolves disputes by refereed re-execution; Karvonen et al. (2025); Wang et al. (2025) address nondeterminism and public verifiability. All require a per-token output distribution and therefore an autoregressive decoder. The sketch literature we borrow from for activation comparison is Johnson–Lindenstrauss random projection (Johnson and Lindenstrauss, 1984; Achlioptas, 2003). Verifying image inference is not itself new: heavyweight zero-knowledge proof systems verify model execution cryptographically (Kang et al., 2022), at orders-of-magnitude overhead and mostly for smaller networks; generic *optimistic* re-execution runs image models such as Stable Diffusion on-chain behind a fraud-proof challenge (Conway et al., 2024); and statistical fingerprinting detects a substituted generative model from its outputs (Yao and Juarez, 2025). What we have not found is a lightweight, non-zk construction *specific to the denoising process*: a trajectory commitment whose single-step re-check catches both step-skipping and substitution at cost $\rho \approx 1/N$, exploiting the one structure a diffusion model has that an output fingerprint discards. The contribution here is that mechanism together with the *measurement on real engines* (that an honest step reproduces exactly and a cheat does not, across three independent modalities).

If there is a lesson past diffusion, it is that the verifiability of a computation is a question of what you commit to, not of what the model happens to output. For any generator that works by iterating, and most of the capable ones do, the process is already there, waiting to be committed.

References

Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.

Arasu Arun, Adam St. Arnaud, Alexey Titov, Brian Wilcox, Viktor Kolobaric, Marc Brinkmann, Oguzhan Ersoy, Ben Fielding, and Joseph Bonneau. Verde: Verification via refereed delegation for machine learning programs, 2025. Gensyn.

KD Conway et al. opML: Optimistic machine learning on blockchain, 2024. ORA / Hyper Oracle.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. NeurIPS 2021 Workshop on Deep Generative Models.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Inference.net. LOGIC: Trustless inference through log-probability verification. Technical report, `https://inference.net/blog/logic`, 2025.

William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

Daniel Kang, Tatsunori Hashimoto, Ion Stoica, and Yi Sun. Scaling up trustless DNN inference with zero-knowledge proofs, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Adam Karvonen, Daniel Reuter, Roy Rinberg, Luke Marks, Adrià Garriga-Alonso, and Keri Warr. DiFR: Inference verification despite nondeterminism, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology — CRYPTO '87*. Springer, 1988.

Jack Min Ong, Matthew Di Ferrante, Aaron Pazdera, Ryan Garner, Sami Jaghouar, Manveer Basra, Max Ryabinin, and Johannes Hagemann. TOPLOC: A locality sensitive hashing scheme for trustless verifiable inference, 2025. Prime Intellect.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

Seyedmorteza Sadat, Manuel Kansy, Otmar Hilliges, and Romann M. Weber. Eliminating oversaturation and artifacts of high guidance scales in diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025. Adaptive Projected Guidance (APG).

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Ke Wang, Zishuo Zhao, Xinyuan Song, Zelin Li, Libin Xia, Chris Tong, Bill Shi, Wenjie Qu, Eric Yang, and Lynn Ai. VeriLLM: A lightweight framework for publicly verifiable decentralized inference, 2025.

Kai Yao and Marc Juarez. AuthPrint: Fingerprinting generative models against malicious model providers, 2025.