

Peer Review

Review of: "Enhancing Sample Generation of Diffusion Models using Noise Level Correction"

Evgenii Arsentev¹

1. Independent researcher

1. What the work claims

The paper starts from a known geometric reading of diffusion sampling: each denoising step is an approximate projection onto the data manifold, and the scheduled noise level σ_t stands in for the distance of the current sample to that manifold. The authors argue that this stand-in drifts during sampling, especially in the last steps and especially after constraint projections, and that the drift hurts sample quality. Their fix is a small network that predicts a residual correction to the noise level (§3.2). It reuses the frozen encoder of the pretrained UNet, adds two residual blocks and one attention block (Table 7), and is trained against the forward-process distance between a noisy sample and its clean source. The corrected level then replaces the scheduled one inside DDPM, DDIM, EDM, and DDNM (Algorithms 1, 3, 4).

Two further contributions follow. IterProj-NLC (§3.4, Algorithm 2) is an alternating-projection sampler for linear inverse problems that alternates a corrected denoising step with a pseudo-inverse projection onto the constraint set. LT-NLC (§4.4) replaces the network with a lookup table built from the average correction curve in Figure 5. Evidence comes from a sphere toy problem (§4.1, Figure 4, Appendix C), unconstrained CIFAR-10 generation (Tables 1, 2, 5, 6), and five restoration tasks on ImageNet and CelebA-HQ (Tables 3, 4, 10).

2. Whether the conclusion holds on the data

I like this paper. The idea is simple, the geometric motivation in §3.1 and Figure 2 is easy to follow, and the method is cheap: the correction network is trained on 19.2 M samples against hundreds of millions or billions for the denoiser (Table 8, row "# Samples"), and I checked that 150 K iterations $\times$ batch 128 and

300 K × 64 both give exactly 19.2 M. The unconstrained gains are large and consistent. I recomputed every DDIM figure quoted in §4.2 from Table 1: 14.21 → 9.61 at 10 steps is 32.4%, 8.25 → 5.66 at 20 steps is 31.4%, 5.17 → 4.04 at 50 steps is 21.9%, 4.66 → 3.12 at 100 steps is 33.0%, 4.32 → 3.11 at 300 steps is 28.0%. All five match the text. The Euler rows of Table 2 also show a clear effect (12.28 → 8.17 at 13 steps).

Where I would ask for more care is in how strongly the remaining numbers are read.

Variance. No table reports a standard deviation, a number of seeds, or the number of generated images behind each FID. For the large effects, this hardly matters. For the small ones, it does. Table 2, row Heun, 35 steps: 1.98 versus 1.95 for Heun-NLC; Table 6, same column: 1.98 versus 1.97 for Heun-LT-NLC. Differences of 0.01 to 0.03 FID are within the run-to-run spread one usually sees on CIFAR-10, so I would not count them as improvements without repeated runs. The same applies to several restoration cells, for example, Table 4, 4× SR, PSNR 31.63 (DDNM) versus 31.78 (DDNM-NLC).

"Best performance across all benchmarks" (§4.3). The tables do not quite support the word "all." In Table 3, column Inpainting, SSIM is 0.968 for RePaint and 0.966 for IterProj-NLC. In Table 4, the same column ties at 0.981. More interestingly, NLC sometimes lowers PSNR: Table 3, column CS 25%, PSNR is 21.62 for DDNM and 21.27 for DDNM-NLC, and Table 10 shows 21.48 for DDNM-LT-NLC in that cell. FID and SSIM improve in those same cells, so this looks like a perception–distortion trade-off rather than a failure. It would be worth saying so.

Quoted percentages. Most are right; three are not. §4.3 gives a 6% FID gain for SR on ImageNet, but Table 3, column 4× SR, FID 39.56 (DDNM) → 37.48 (IterProj-NLC) is $(39.56 - 37.48)/39.56 = 5.3\%$. §4.2 says NLC improves Heun by 10% at 13 steps; Table 2 gives $(7.22 - 6.56)/7.22 = 9.1\%$. And in §4.4 the LT-NLC gains are stated as 14%, 20%, and 15% for 10, 20, and 50 steps, while Table 5 gives $(14.21 - 11.21)/14.21 = 21.1\%$ at 10 steps, $(8.25 - 6.54)/8.25 = 20.7\%$ at 20, and $(5.17 - 4.37)/5.17 = 15.5\%$ at 50. The 10-step figure understates the authors' own result. The other quoted gains check out: Deblurring 1.17 → 0.48 is 59.0%, CS 25% 64.68 → 55.69 is 13.9%, Inpainting 9.79 → 4.90 is 49.9%, and colorization consistency 42.32 → 38.30 is 9.5% (Table 3, row IterProj-NLC against row DDNM).

Compute parity for IterProj-NLC. §4.3 states that all methods use the same 100-step process. §3.4, though, describes IterProj-NLC as permitting "an arbitrary number of refinement steps" with a geometric decay of the noise level. I could not find the decay factor, the stopping rule, or the number of denoiser evaluations actually used for Tables 3 and 4. Since IterProj-NLC is the row that wins most cells, this is the single number I would most like to see.

Baselines. The comparison is fair inside its scope: every method uses the same pretrained denoiser. What is missing is a comparison with other inference-time fixes for the same symptom. Ning et al. (ICML 2023, input perturbation against exposure bias) is in the reference list, and the time-shift line of work on exposure bias attacks a very similar mismatch between scheduled and actual noise level. One row of each in Table 1 would show whether a learned correction buys more than a cheaper heuristic.

"Parameter-free" lookup table. §4.4 builds the table from the average curve of the trained network (Figure 5). So LT-NLC is free of parameters at inference time, but it still requires training the NLC network first. The abstract and conclusion should say that.

None of this undoes the main finding. The CIFAR-10 gains in Tables 1 and 5 are well beyond noise, and the restoration gains on Deblurring and Inpainting FID in Table 3 are large. The claims mostly need tighter wording and error bars, not new science.

3. Reproducibility

Some things are well documented. Appendix D.1 names the pretrained checkpoints by reference ([23] and [24] for CIFAR-10, [47] for ImageNet, [46] for CelebA-HQ). Table 7 gives the correction network's shape per dataset, Table 8 gives batch size, learning rate (0.0003), and iteration counts, and §D.2 gives a sense of cost (about two weeks on 8 Tesla V100 GPUs for the ImageNet denoiser, about one day for the correction network). §3.2 also states the range of the training scaling factor (λ drawn from $U(1-\delta, 1+\delta)$ with $\delta = 0.5$). The restoration protocol follows DDNM [22], which is a sensible choice.

Several things are missing, and I list them so the authors can add them cheaply.

- I found no code or checkpoint link in the Qeios version.
- No random seeds, no number of runs, and no number of samples per FID evaluation. The reference statistics for FID (training set or test set) are not named.
- The size and identity of the ImageNet and CelebA-HQ evaluation subsets for Tables 3 and 4 are not stated in the text I read.
- §3.4 introduces a decay factor $\eta < 1$ for IterProj-NLC, but I could not find the value used in the experiments.
- Table 9 has no units. §D.2 says the times are for 10-step generation at batch size 1, and I assume seconds. The text puts the overhead at approximately 10%, but the table gives $0.38/0.32 = 1.19\times$ on CIFAR-10 and $1.19/0.93 = 1.28\times$ on ImageNet, that is about 19% and 28%.

A general rendering note. On Qeios, almost every equation appears as duplicated, broken MathML, so most numbered equations, Theorem 1, and the loss in §3.2 are hard to read as rendered. I reviewed the mathematics only where the surrounding prose makes it unambiguous. Theorem 1 is stated informally, and its proofs are deferred to [16]; I could not check the precise statement from this rendering.

4. *What specifically to fix*

1. Tables 1–6 and 10: report mean and standard deviation over at least three seeds, plus the number of images used for FID. Start with the cells where the gap is below 0.1 FID (Table 2 and Table 6, column 35).
2. §4.3, sentence "achieves the best performance across all benchmarks". Soften it. Table 3, Inpainting SSIM (RePaint 0.968 versus 0.966), and Table 4, Inpainting SSIM (tie at 0.981), are exceptions.
3. Correct the percentages: 6% $\rightarrow$ 5.3% for ImageNet SR FID (§4.3), 10% $\rightarrow$ 9.1% for Heun at 13 steps (§4.2), and 14% $\rightarrow$ 21% for LT-NLC at 10 steps (§4.4, from Table 5).
4. Discuss the PSNR drop in Table 3, CS 25% (21.62 $\rightarrow$ 21.27 for DDNM-NLC; 21.48 for DDNM-LT-NLC in Table 10) as a possible perception–distortion trade-off.
5. For Tables 3 and 4, state the number of denoiser evaluations, the decay factor, and the stopping rule used by IterProj-NLC, so readers can confirm that the "same 100-step" condition holds for it.
6. Add one or two inference-time baselines that target the same scheduled-versus-actual noise mismatch (for example, input perturbation from Ning et al., already cited, or a time-shift sampler) to Table 1.
7. Table 7, rows "# Parameters": the ratios of denoiser to correction network are $218/14 = 15.6$, $2109/234 = 9.0$, and $434/59 = 7.4$, so "approximately ten times smaller" holds only on average. I would also double-check the 218 M figure for the CIFAR-10 denoiser, which looks large for a 32×32 model; if it counts something other than parameters, say what.
8. Table 9: add units, and reconcile the "approximately 10%" overhead in §D.2 with the 19% and 28% that the table implies.
9. §4.4, abstract, and §5: say that the lookup table is derived from a trained NLC network.
10. Small typos: "256 $\times$ 255" appears twice in §D.1 (surely 256 $\times$ 256); Table 7 header "Heperparameter"; Table 6 caption "with and with LT-NLC"; "path-planningand" in §1.
11. Release code and the trained correction networks, or say why not.
12. Check how the equations render on Qeios and, if needed, upload a PDF or fix the MathML.

5. Recommendation

Minor revisions. The method is original in a modest, useful way: it turns the "denoising as projection" view into a concrete, cheap correction that plugs into several existing samplers and improves CIFAR-10 FID by roughly a fifth to a third in Table 1. The restoration results are also convincing on the tasks where gains are large. What I am asking for is mostly bookkeeping: error bars, three corrected percentages, a clear compute budget for IterProj-NLC, a consistent overhead figure, and one or two extra baselines. None of it should change the paper's main message.

Declarations

Potential competing interests: No potential competing interests to declare.