Thank you for the constructive feedback. We address concerns raised by the reviewers 1 and 2 regarding comparative results by an update with additional baselines in the literature.

Updated results with baselines Table 1 shows an updated version of the Glow-based experiments. Our initial demonstration at submission timeline was an ablation study without full convergence. This could potentially result in misleading impressions of our method. After the submission, we trained each models for longer duration up to 3000 epochs where each model reached the saturation. Thus, we can now accurately compare the models to the results in the literature. † indicates that the numbers are taken from existing literature under uniform dequantization regime for fairness (Finlay et al, How to train your neural ODE, ICML 2020). Reviewer 1 and 2 raised a valid concern regarding to the baseline Glow model that it is hard to identify the gain in bpd at the cost of the larger architecture. Here, we accurately show that the baseline Glow does scale with the higher capacity, at the cost of the increased parameters and diminishing return. NanoFlow outperformed the recently proposed models with less capacity, even compared to models with more complex non-affine coupling, including neural spline flows (RQ-NSF (C)), Flow++, and ResidualFlow.

Table 1: Update of Table 4 with converged models and additional baselines.

Model (converged)	Params (M)	bpd
Glow (256 channels)	15.973	3.40
Glow (512 channels) †	44.235	3.35
Glow-large	287.489	3.30
RQ-NSF (C) †	11.8	3.38
FFJORD †	0.801	3.40
Flow++ †	32.3	3.28
ResidualFlow †	25.174	3.28
NanoFlow-naive	9.263	3.40
NanoFlow-decomp	9.935	3.32
NanoFlow	10.113	**3.27**
NanoFlow (K=48)	10.718	**3.25**

Filling in missing results Reviewer 1 pointed out that the results should also include NanoFlow's method applied to the reference topology of Glow to fully compare the results for better understanding of the method. Table 2 shows the results when applied to the reference Glow topology. Original Glow uses a total of 3 layers: 3×3 conv $\rightarrow 1 \times 1$ conv $\rightarrow 3 \times 3$ projection conv. NanoFlowAlt shares the first two layers and use separate 3×3 projection conv. Results show that the model performed significantly worse than the baseline architecture, even though NanoFlowAlt (K=48) has similar network size (10 M) to our main result. This is reasonable and expected, because the shared estimator's capacity is too restricted (0.7 M) to model multistep densities. Thus, one should be careful about allocating the parameters under NanoFlow framework, where the shared estimator should have sufficient capacity, while keeping the non-shared projection layers slim. We hope that these addional results combined with our main updated result would alleviate the concern about "moving the goal post". Reviewer 1 also mentioned that NanoFlow-decomp is missing in WaveFlow results, thus it is hard to identify how much the gain is achieved from our two strategies (decomposition and embedding). Table 3 shows that NanoFlow-decomp stays between NanoFlow-naive and NanoFlow, consistent with the Glow-based models. Thus, we emphasize that both methods do contribute to the performance. We will add the converged results. We also attach preliminary results applying RQ-NSF to WaveFlow-based models in Table 3, and show that NanoFlow is applicable beyond affine coupling.

Table 2: Results when applying the method to the reference topology of Glow model (NanoFlowAlt), evaluated at 600 epochs.

Model (600 epochs)	Params (M)	bpd
Glow (256 channels)	15.973	**3.44**
NanoFlowAlt-naive	0.778	3.75
NanoFlowAlt-decomp	6.783	3.54
NanoFlowAlt	6.961	3.53
NanoFlowAlt (K=48)	10.319	3.51

Implementation details and choice of embedding We will precisely add implementation details of the embedding strategies we used to improve clarity. The input at the start of each flow is $h^{k,0} = concatenate(x, \epsilon^k)$. for l-th layer inside k-th flow, we get hidden state for the next layer $h^{k,l+1}$ as follows:

$$h^{k,l+1} = \exp(\delta^{k,l}) * (g^{k,l}(h^{k,l}; \hat{\theta}^{\cdot,l}) + g^{k,l}_{embed}(\epsilon^k; \eta^{k,l})) \tag{1}$$

where $\delta^{k,l} \in \mathbb{R}^H$ serves as the multiplicative gating and H is the number of hidden channels. It is initialized to zero to initially perform identity. $g^{k,l}_{embed}(\epsilon^k; \eta^{k,l}) \in \mathbb{R}^H$ is the additive bias obtained by $\eta^{k,l}$ with one fully-connected layer (1×1 convolution). $\eta^{k,l}$ is discarded after training and caching the bias vectors from $g^{k,l}_{embed}$. We observed that these implementation choices ensured stability during training. We partially dropped the methods that showed no improvements during preliminary experiments, depending on the architecture.

Table 3: Additional WaveFlow experiment, now with NanoFlow-decomp evaluated at 100K steps.

Model	Params (M)	LL
WaveFlow	22.336	5.1164
NanoFlow-naive	2.792	5.0341
NanoFlow-decomp	2.794	5.048
NanoFlow	2.818	5.0774
WaveFlow + RQ-NSF	22.432	5.1262
NanoFlow + RQ-NSF	2.915	5.0862

Improving related work In line with the updated results in Table 1, we will describe other classes of flows beyond the affine coupling in related work and background. Notably, continuous-time flows (CNF) can utilize a "shared" neural network f, as commented by reviewer 2. The central difference between CNF and NanoFlow (and non-continuous NFs in general) is that CNFs use numerical ODE solvers that iteratively evaluate f to reach below tolerance, whereas NFs directly model pre-defined steps of transformation with f_k (or f in NanoFlow) with a single pass. The effectiveness and potential benefits of the shared f outside the ODE-based CNFs are yet to be studied in the literature, where NanoFlow aims to systematically address, reaching to non-trivial solutions as proposed.