Reviewer #1: Thank you for your questions. We will incorporate our answers into the paper.

- We used a deep and thin network and a modified objective that emphasized the reconstruction aspect of the VAE. This led to very stiff models, but are of no practical use as they have to be trained for an extremely long time.
- We only explored 2d SDEs for this initial proof of concept as we found that the literature consists mostly of 2d synthetic SDEs when comparing different inference methods. We would appreciate it if the reviewer could suggest a suitable dataset if they have one in mind.
- With a full diffusion matrix of d*m, where m is the dimension of W, we would require m DiffOpNets, one for each row of the matrix. Our approach should still reduce computation by a factor of d, but we did not pursue this.
- The SDE training is very fast and can be trained in a few minutes due to its parallelism. The DiffOp-CNF is slower due to having to sequentially solve an ODE, and can take days for convergence on MNIST. Our method should be around the same time cost as FFJORD (which also reported to be slower than alternatives due to the sequential nature of solving an ODE).
- We will reference the original IWAE paper, "Importance Weighted Autoencoders" by Burda et al.

Reviewer #2:

Comparing fairly to regular networks. This is difficult to carry out because (i) it's unclear how fair comparisons should be constructed and (ii) the exact dimension-wise derivative (as used for training e.g. CNFs) is only feasible in very low dimensions or otherwise would require a significant amount of time. For instance, FFJORD reported computing the exact trace to be infeasible and as such could not provide NLL estimates using the IWAE estimator. We did not come up with a satisfactory answer to this during the rebuttal, but we thank the reviewer for bringing this up.

DiffOpNets are not composable. This is a very good point, and we will add this to the paper. For a function composition $f \circ g$, the Jacobian would be $J_f J_g$ whose diagonal cannot be computed using just the diagonals of each Jacobian. However, the networks that make up a DiffOpNet can have arbitrary depth and hidden state width.

We did not explore higher-dimensional SDE problems as we could not identify a suitable dataset, and we believe our experiment results would carry over to simple synthetic datasets. We would appreciate it if the reviewer could suggest a suitable dataset if they have one in mind. We hope to explore this combination of cheap derivatives and SDEs more in the future, as dimension-wise derivatives show up in many equations involving SDEs.

We also thank the reviewer for their minor corrections/suggestions.

Reviewer #3: We thank the reviewer for taking the time to read our paper, and we apologize that the reviewer found our paper to be difficult to comprehend. We understand that we choose a non-standard format of having one core idea and three stand-alone applications which may as of now be only of a niche interest. However, we believe our contributions are useful to machine learning research, and we hoped to convey its wide applicability (ranging from numerical integration, to modeling, to optimization) using our format. We believe there are many areas and applications that can make use of dimension-wise derivatives but are yet unexplored due to computational infeasibility.

We plan on including more background information in each section using the extra page allowance should this paper be accepted. The reason we chose these applications is because the CNF experiments are on standard benchmark datasets, while the SDE experiments motivate the use of second-order dimension-wise derivatives.

Our main contribution lies in reducing the computational cost by a factor of d. Comparison is difficult, as naïve computation would be infeasible in but all but very low dimensions. Instead, we focused on motivating why computing the dimension-wise derivatives are useful in the first place as this has not been explored in previous works due to its computational cost. We compared against a *stochastic* trace estimator which has the same time complexity as our method (and is less general), but has higher variance gradients, which can impede optimization.

It was shown [1] that for ODE models, the number of function evaluations is linear with the wall-clock time. As such, we only reported number of function evaluations as this is more reproducible and comparable between implementations.

i) While we still have d networks, our comment is only meant to illustrate that the expressiveness would be equal to a standard network, i.e. it can express the same set of functions. If the bottleneck is large enough, one could set $\tau_i(x)$ to simply be the i-th output of a standard network.

ii) You're correct that g is τ. This was missed during a change of notation. We will correct this.

[1] "Neural Ordinary Differential Equations" Chen et al. (2018)