EFFICIENT MOLECULAR CONFORMER GENERATION WITH SO(3) AVERAGED FLOW-MATCHING AND REFLOW

Zhonglin Cao[1,*] **Mario Geiger**[1,*] **Allan dos Santos Costa**[1,2,†]
Danny Reidenbach[1] **Karsten Kreis**[1] **Tomas Geffner**[1]
Franco Pellegrini[3] **Guoqing Zhou**[1] **Emine Kucukbenli**[1]

[1]NVIDIA [2]Massachusetts Institute of Technology [3]SISSA

ABSTRACT

Fast and accurate generation of molecular conformers is desired for downstream computational chemistry and drug discovery tasks. In this work, we propose two mechanisms for accelerating the training and inference of flow-based generative model for 3D molecular conformer generation. For fast training, we introduce the SO(3)-*Averaged Flow* training objective, which we show to converge faster and generate better conformer ensembles compared to conditional optimal transport and Kabsch alignment-based optimal transport flow. For fast inference, we demonstrate that reflow methods and distillation of these models enable few-steps or even one-step molecular conformer generation with high quality. Using these two techniques, we demonstrate a model that can match the performance of strong transformer baselines with only a fraction of the number of parameters and generation steps.

1 INTRODUCTION

Molecular conformer generation is the task to predict the ensemble of 3D conformations of molecules given the 2D molecular graphs (Hawkins, 2017). In the domain of drug discovery, molecular conformer generation is a prerequisite for both structure-based and ligand-based compound virtual screening applications. For established computational chemistry molecular conformer generation tools, there is a trade-off between generation speed and the quality/diversity of generated conformers (Axelrod & Gomez-Bombarelli, 2022). Deep generative models are being sought as a potential solution to overcome such trade-off and bring fast, diverse, and high-quality molecular conformer generation. Many earlier works are based on generative models (Simm & Hernández-Lobato, 2019; Luo et al., 2021; Xu et al., 2022). However, established cheminformatics tools still has better generation quality with faster sampling speed compared with early deep-learning based methods. Torsional diffusion (Jing et al., 2022) is the first diffusion model that achieves better generation quality than cheminformatics model. More recent works such as Molecular conformer field (MCF) (Wang et al., 2024) and ET-Flow (Hassan et al., 2024) perform diffusion/flow-matching directly on the Cartesian coordinates of the atoms. With more scalable transformer architecture, They have achieved the state-of-the-art conformer generation quality. However, iterative ODE or SDE solving with large transformer model to generate every conformer can still be computationally infeasible when the library to be virtually screened contains billions of compounds (Bellmann et al., 2022).

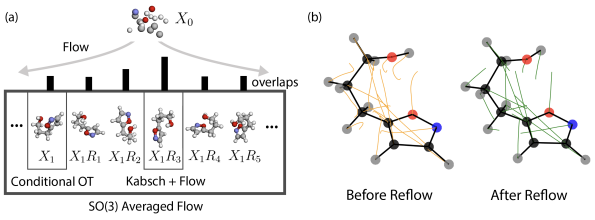

Figure 1: **SO(3)-Averaged Flow and Reflow (a)** We illustrate a comparison between our approach *Averaged Flow*, conditional OT, and Kabsch+Flow. While conditional OT randomly assigns any rotation of the data, Kabsch+Flow assigns the rotation of largest overlap. Our method instead computes the expected flow across all rotations. **(b)** The flow trajectories (100 Euler steps) are effectively straightened after reflow.

In this work, we propose **(i)** a novel flow-matching training objective called SO(3)-*Averaged Flow* (Fig. 1a). As an objective, *Averaged Flow* avoids the need to rotationally align prior and data distribution by *analytically* computing the averaged probability flow from the prior to all the rotations of the data sample. Model trained with *Averaged Flow* is experimentally shown to converge faster to better performance. **(ii)** To improve the sampling efficiency, we adopt the *reflow* and distillation technique (Liu et al., 2022) to straighten the flow trajectories (Fig. 1b). Straightened trajectories

[*] Equal Contribution; [†] Work performed during internship at NVIDIA

allow high quality molecular conformer generation with few-step or even one-step ODE solving, thus significantly relieving the computational cost.

2 METHOD

2.1 SO(3)-*Averaged Flow*

The concept of *Averaged Flow* involves recognizing that the data distribution q may exhibit group symmetries, which can be explicitly integrated out. A symmetry group G of q consists of transformations $g : x \mapsto g \cdot x$ that leave the distribution q unchanged, meaning $q(x) = q(g \cdot x)$.

If we focus on Lie groups with a Haar measure(Zee, 2016; Nachbin & Bechtolsheim, 1965; Chirikjian & Kyatkin, 2000), we can express q as $q(x) = \int d\hat{x}\ \hat{q}(\hat{x}) \int dg\ \delta_{g\cdot\hat{x}}(x)$, where $\hat{q}$ represents the distribution over the group orbits, $\hat{x}$ is a representative point of the orbit, and the integral over G uses the Haar measure.

By substituting this into equation equation FM8, we obtain:

$$u_t(x) = \int d\hat{x}\ \hat{q}(\hat{x}) \int dg\ u_t(x|g \cdot \hat{x}) \frac{p_t(x|g \cdot \hat{x})}{p_t(x)} \tag{1}$$

Notice that $p_t(x) = \int d\hat{x}\ \hat{q}(\hat{x}) \int dg\ p_t(x|g \cdot \hat{x})$ is the partition function.

Let's consider the case of conformer generation: **(i)** x is a $N\times 3$ matrix representing the 3D coordinates of N atoms. **(ii)** The group G is the rotation group $SO(3)$. We will use R to denote the rotation matrix, which acts on x as $x \mapsto xR^T$. **(iii)** The orbits $\hat{x}$ in this case corresponds to the different low-energy conformers of a given molecule and their permutations that leave the 2D molecular graph invariant. Therefore, the integral $\int d\hat{x}\ \hat{q}(\hat{x})$ in Eq.1 representing the entire conformer ensemble can be written as $\sum_{\hat{x}\in\text{conformers}} \hat{q}(\hat{x})$, where $\hat{q}(\hat{x})$ is the weight associated to that conformer. **(iv)** $p_t(x|x_1)$ is a Gaussian of the form: $p_t(x|x_1) \propto \exp\left(\frac{1}{2}\frac{1}{(1-t)^2}\sum_{ij\delta}(x-tx_1)_{i\delta}\Sigma_{ij}(x-tx_1)_{j\delta}\right) \equiv \exp\left(\frac{1}{2}\frac{\|x-tx_1\|^2_\Sigma}{(1-t)^2}\right)$ where Σ is a $\mathbb{R}^{N\times N}$ matrix. We will use the notation $\|A\|^2_\Sigma = \mathrm{tr}(A^T\Sigma A)$.

Let's rewrite $u_t(x)$, the flow-matching objective that averaged over SO(3) group, in the conformer generation case:

$$u_t(x) = \frac{1}{Z_t(x,0)} \sum_{\hat{x}\in\text{conformers}} \hat{q}(\hat{x}) \int_{SO(3)} dR\ \frac{\hat{x}R^T - x}{1-t} e^{-\frac{1}{2}\frac{\|x-t\hat{x}R^T\|^2_\Sigma}{(1-t)^2}} \tag{2}$$

and define $Z_t(x,\alpha)$ as:

$$Z_t(x,\alpha) = \sum_{\hat{x}\in\text{conformers}} \hat{q}(\hat{x}) \int_{SO(3)} dR\ e^{-\frac{1}{2}\frac{\|x-t\hat{x}R^T\|^2_\Sigma}{(1-t)^2} + \mathrm{tr}(\alpha^T\hat{x}R^T)} \tag{3}$$

with α being an $N \times 3$ matrix that will be needed in the following steps.

Note $u_t(x)$ can be calculated via the derivative of $\log Z_t(x,\alpha)$ with respect to α evaluated at $\alpha = 0$,

$$u_t(x_t) = \left([\partial_\alpha \log Z_t(x_t,\alpha)]_{\alpha=0} - x_t\right)/(1-t) \tag{4}$$

The integral over R can be computed using the formula from Mohlin et al. (2020), which provides a closed-form solution for $F \mapsto \log \int_{SO(3)} dR \exp(\mathrm{tr}(FR^T))$, where F can be any 3×3 matrix. In our case, we have

$$\log Z_t(x,\alpha) = \log \sum_{\hat{x}\in\text{conformers}} \hat{q}(\hat{x}) \exp\left(\underbrace{\log \int_{SO(3)} dR e^{\mathrm{tr}((\alpha^T + \frac{t}{(1-t)^2}x^T\Sigma)\hat{x}R^T)}}_{\text{closed-form solution using } F = \alpha^T\hat{x} + \frac{t}{(1-t)^2}x^T\Sigma\hat{x}} - \frac{\mathrm{tr}(x^T\Sigma x) + t^2\mathrm{tr}(\hat{x}^T\Sigma\hat{x})}{2(1-t)^2}\right) \tag{5}$$

Then we can directly learn the analytically solved SO(3)-*Averaged Flow* $u_t(x_t)$:

$$\mathcal{L}_{\text{AvgFlow}}(\theta) = \mathbb{E}\Big[\|v_t^\theta(x_t) - u_t(x_t)\|^2\Big], \text{with } t \in [0, 1]. \tag{6}$$

We provide the python implementation of this formula in Appendix A.4.1. We note that while our *Averaged Flow* implementation is capable of handling multiple conformer states in the summation in Eq 5. In practice, we approximate the expectation of the conformer ensemble through sampling one conformer in each training epoch. The benchmark of computation time (Table A.4.2) shows that only a small overhead is added when using the *Averaged Flow* objective.

2.2 Reflow and Distillation

One effective technique to reduce the sampling steps without significantly sacrificing the generation quality is to straighten the trajectory. Inspired by the success of such technique in point-cloud generation (Wu et al., 2023) and text-image generation (Esser et al., 2024; Liu et al., 2023b), we finetune our model v_t^θ trained with Averaged Flow using the *reflow* algorithm proposed in previous rectified flow works (Liu et al., 2022; Liu, 2022). Specifically, we first randomly sample atom coordinates X_0' from standard Gaussian and generates the corresponding conformer X_1' using the Tsitouras' 5/4 solver (Tsitouras, 2011). The coupling (X_0', X_1') is then used in the rectified flow objective to finetune the model:

$$\mathcal{L}_{\text{Reflow}}(\theta) = \mathbb{E}\Big[\|v_t^\theta(X_t', t) - (X_1' - X_0')\|^2\Big], \text{with } t \in [0, 1] \tag{7}$$

Liu et al. (2022) proved that the coupling (X_0', X_1') yields equal or lower transport cost than (X_0, X_1) where X_0 is sampled from noise distribution and X_1 from data distribution. Therefore, applying the reflow algorithm to fine-tune model with Eq. 7 can effectively reduce the transport cost and straighten the trajectory. We empirically find that the transport trajectories bridging Gaussian noise and molecular conformers demonstrates high curvature when t is closer to 0 (Fig. 1b). Therefore, inspired by Lee et al. (2024), we sample t from a exponential distribution with the probability density function as $p(t) \propto \text{Exp}(\lambda t)$, where λ is -1.2 by selection to focus the training more on $t < 0.5$. The distribution of t is visualized in Fig. 4. After reflow, the sampling speed can be further reduced by distilling the relation of the coupling (X_0', X_1') into model v_θ to enable 1-step transport and eliminate the need of ODE solving. During the distillation stage, we fine-tune the reflowed model v_θ with the following 7 with $t = 0$. We use a SE(3)-equivariant networks based on Batzner et al. (2022) for predicting the time-dependent vector field. Details of our model are provided in Sec. A.2.3 and Fig.5. Overall, the model is trained and fine-tuned using *Averaged Flow* + reflow + distillation following the Algorithm 1. Details of model sampling are included in Sec. A.2.6.

3 Experiments

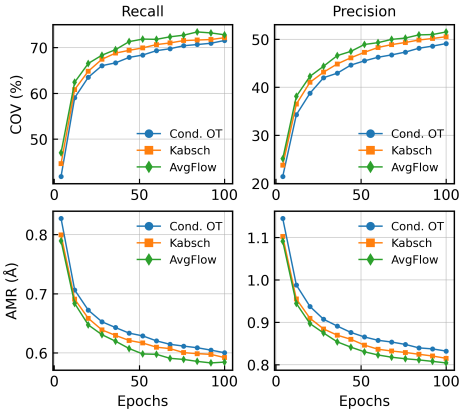

Figure 2: **Model trained with *Averaged Flow* consistently converge to better performance on GEOM-Drugs.** The two objective we compared *Averaged Flow* to are: *(i)* Conditional OT and *(ii)* Kabsch alignment of noise X_0 with conformer X_1 before conditional OT. Values are the average of a 300-molecule test subset.

Following previous works, we train and evaluate our model on the GEOM-QM9 and GEOM-Drugs dataset (Axelrod & Gomez-Bombarelli, 2022). We followed the splitting strategy proposed by Ganea et al. (2021) and test our model on the same test set containing 1000 molecules for both QM9 and Drugs dataset. Dataset and splitting details are included in Sec. A.2.4. The major model evaluation metrics are the average minimum RMSD (AMR, the lower the better) and coverage (COV, the higher the better). Both AMR and coverage are reported for precision (AMR-P and COV-P) and recall (AMR-R and COV-R). The definition of metrics are specified in Sec.A.3.1. Intuitively, coverage measures the percentage of ground truth conformers being generated (recall) or the percentage of generated conformers being close enough to ground truth (precision), while AMR measures the average RMSD between each ground truth and its closest generated conformer (recall) or vice versa (precision).

3.1 AVERAGED FLOW LEADS TO FASTER CONVERGENCE TO BETTER PERFORMANCE

To showcase the advantage of the *Averaged Flow* over other training objectives, we evaluate the performance of model trained on different objectives using a randomly sampled GEOM-Drugs test subset containing 300 molecules. The two other objectives to be compared are conditional OT and Kabsch alignment. The Kabsch alignment objective is to rotationally align the sampled noise X_0 with conformer X_1 before training with the conditional OT objective. Fig. 2 demonstrates that model trained with *Averaged Flow* is consistently better than with both conditional OT and Kabsch alignment on all four metrics. With only 68 epochs of training, *Averaged Flow* has COV-R higher and AMR-R lower than the other two objectives trained for 100 epochs. The COV-P (49.3%) and AMR-P (0.831) of *Averaged Flow* trained for 52 epochs are better than conditional OT (COV-P= 49.1% and AMR-P= 0.832Å) trained for 100 epochs. Also, *Averaged Flow* outperforms Kabsch trained for 100 epochs on AMR-P (*Averaged Flow* $= 0.814$Å and Kabsch= 0.815Å) and on COV-P (*Averaged Flow* $= 50.9\%$ and Kabsch= 50.5%) after 76 and 84 epochs, respectively. Overall, model trained with *Averaged Flow* converges with less epochs to better performance in molecular conformer generation.

3.2 GEOM-QM9 AND GEOM-DRUGS

On the GEOM-QM9 dataset, we compared our model with two prevailingly used cheminformatics tools: RDKit and OMEGA, along with GeoMol (Ganea et al., 2021), Torsional Diffusion (Jing et al., 2022), ET-Flow-SS (Hassan et al., 2024), and MCF (Wang et al., 2024). We denote our model trained with *Averaged Flow* as AvgFlow, the model finetuned with reflow as $\text{AvgFlow}_{\text{Reflow}}$, and the model further finetuned with distillation as $\text{AvgFlow}_{\text{Distill}}$. The number of sampling steps required by diffusion and flow-matching model are also noted. Table. 3 shows that AvgFlow outperforms all other models in the COV-R metrics and almost matching the AMR-R of ET-Flow-SS, indicating it is capable of generating very diverse conformers on the GEOM-QM9 dataset. More importantly, the $\text{AvgFlow}_{\text{Reflow}}$ and $\text{AvgFlow}_{\text{Distill}}$ achieve higher COV-R than other models with only 2-step and 1-step ODE sampling, respectively. $\text{AvgFlow}_{\text{Reflow}}$ also outperforms all cheminformatics tools and GeoMol in all metrics. The benchmark on GEOM-QM9 shows that our model can match the performance of state-of-the-art models with only much less trainable parameters on smaller scale molecule. Table. 3 also shows that reflow+distillation can effectively maintain the conformer generation quality with only 1 or 2 steps of ODE solving.

Table 1: Quality of generated conformer ensembles for GEOM-DRUGS ($\delta = 0.75$Å) test set in terms of Coverage (COV) and Average Minimum RMSD (AMR). Bolded results are the best. Baseline values are taken from the corresponding papers. *Due to the use of adaptive step size, the number of steps of AvgFlow is an average value over all test set molecules.

		Recall				Precision			
		COV (%) ↑		AMR (Å) ↓		COV (%) ↑		AMR (Å) ↓	
Method	Step	Mean	Med	Mean	Med	Mean	Med	Mean	Med
RDKit	-	38.4	28.6	1.058	1.002	40.9	30.8	0.995	0.895
OMEGA	-	53.4	54.6	0.841	0.762	40.5	33.3	0.946	0.854
GeoMol	-	44.6	41.4	0.875	0.834	43.0	36.4	0.928	0.841
Tor. Diff.	20	72.7	80.0	0.582	0.565	55.2	56.9	0.778	0.729
ET-Flow-SS (8.3M)	50	79.6	84.6	0.439	0.406	**75.2**	**81.7**	**0.517**	**0.442**
MCF-S (13M)	1000	79.4	87.5	0.512	0.492	57.4	57.6	0.761	0.715
MCF-B (64M)	1000	84.0	91.5	0.427	0.402	64.0	66.2	0.667	0.605
MCF-L (242M)	1000	**84.7**	**92.2**	**0.390**	**0.247**	66.8	71.3	0.618	0.530
AvgFlow (4.7M)	102*	76.8	83.6	0.523	0.511	60.6	63.5	0.706	0.670
$\text{AvgFlow}_{\text{Reflow}}$ (4.7M)	2	64.2	67.7	0.663	0.661	43.1	38.9	0.871	0.853
$\text{AvgFlow}_{\text{Distill}}$ (4.7M)	1	55.6	56.8	0.739	0.734	36.4	30.5	0.912	0.888

We then trained and benchmarked our model on GEOM-Drugs, which is a larger dataset containing conformers of drug-like molecules. Table. 1 shows that AvgFlow has good performance on GEOM-Drugs by outperforming torsional diffusion on all metrics. Compared with MCF-S which has approximately 3 times more parameters, our model achieves better COV-P and AMR-P, indicating more AvgFlow-generated conformers are close to ground truth conformers. $\text{AvgFlow}_{\text{Reflow}}$ can outperform cheminformatics tools and GeoMol on all metrics, with large margin specifically on the recall metrics. With only 4.7M parameters and 2 ODE steps, $\text{AvgFlow}_{\text{Reflow}}$ pushes the limit of the quality-speed trade-off of molecular conformer generations and bears the potential to be adopted

for large-scale virtual screen use cases. The $\text{AvgFlow}_{\text{Distill}}$ is also shown to achieve better COV-R and AMR-R than cheminformatics tools and GeoMol, showing the our model can maintain high generation diversity even with a single ODE step.

3.3 WHEN IS REFLOW REALLY NECESSARY?

From the benchmark results on GEOM-Drugs and GEOM-QM9, we understand that our $\text{AvgFlow}_{\text{Reflow}}$ model can achieve better performance than cheminformatics methods on all metrics. However, it is obvious that the model's performance drops after reflow especially for the precision metrics. Flow-matching models generally have high generation quality with less steps compared to denoising diffusion model (Lipman et al., 2023) thanks to the ODE sampling process. In this section, we are trying to answer the question: when is reflow really necessary to generate high-quality molecular conformers?

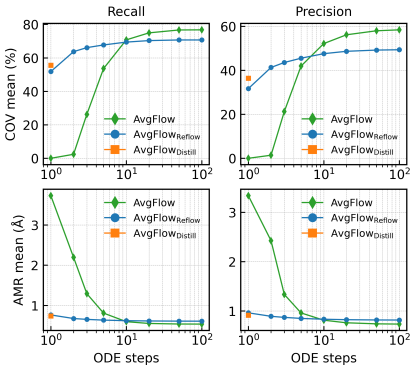

Figure 3: **Effect of the number of ODE steps to model's performance** Comparison between model performance before and after reflow with different number of ODE steps

Fig. 3 shows the the performance of our model using Euler sampling method with number of ODE steps $N_{\text{step}} \in \{1, 2, 3, 5, 10, 20, 50, 100\}$. The performance of models are evaluated with the same four metrics on a subset of the GEOM-Drugs test set containing 300 molecules. Overall, AvgFlow has better performance when $N_{\text{step}} \geq 10$ than $\text{AvgFlow}_{\text{Reflow}}$. When $N_{\text{step}} < 10$, the performance of AvgFlow has start to collapse and eventually reaches 0% coverage for both recall and precision when $N_{\text{step}} = 1$. The performance gap becomes significant for all metrics when $N_{\text{step}} < 5$. $\text{AvgFlow}_{\text{Reflow}}$, on the other hand, has minimal loss in performance until $N_{\text{step}} = 2$ thanks to the straightened flow trajectory. The 1-step generation quality of the model still suffers even after reflow. Distillation can effectively reduce the RMSD of 1-step generated conformers and improve both the COV-R and COV-P. In summary, reflow is critical when generating molecular conformers with very few ODE steps ($N_{\text{step}} < 5$).

3.4 SAMPLING TIME

To demonstrate the sampling efficiency of our model, we compared the sampling wall time of our model with MCF and torsional diffusion. Table. 4 shows the sampling time comparison between models*. The average sampling time of $\text{AvgFlow}_{\text{Reflow}}$ for each conformer in the GEOM-Drugs test set is 2.68 microseconds, which is 21 to 50$\times$ faster than different variants of MCF sampled with DDIM for 3 steps. It is also 48$\times$ faster than torsional diffusion sampled with 5 steps. $\text{AvgFlow}_{\text{Reflow}}$ outperforms MCF-B on precision metrics and reached comparable performance on the recall metrics. $\text{AvgFlow}_{\text{Reflow}}$ also outperforms torsional diffusion and MCF-S by large margin with only a fraction of the sampling time. The major speed-up of the our model is due to the `JAX` implementation and less number of parameters. With reflow ensuring high-quality generation with only 2 ODE steps, our model achieves extraordinary sampling efficiency.

4 CONCLUSION

We have presented SO(3)-*Averaged Flow* as a new objective to accelerate the training of flow-matching models for molecular conformer generation. *Averaged Flow* leads to faster convergence to better performance compared with conditional OT and Kabsch alignment. We have also experimented reflow and distillation to straighten the flow trajectory and enable few-step molecular conformer generation. Our model reaches the state-of-the-art performance on the coverage-recall metric of the GEOM-QM9 dataset. It is also matching the performance of transformer-based model which have several times more parameters on the GEOM-Drugs dataset. By analyzing the effect of number of ODE steps to the model generation quality, we find out that reflow and distillation are necessary when very few steps ($N_{\text{step}} < 5$) of conformer generation is desired. Finally, by comparing the sampling time, we demonstrate that our model is approximately 21 to 50 times faster than the other state-of-the-art models, while achieving second to the best generation quality and diversity. Overall, given that the *Averaged Flow* and reflow training scheme can be applied to any models, our method bridges the gap between multi-step flow-matching models and practical molecular conformer generation application by pushing the boundary of quality-speed trade-off.

* MCF and Torsional Diffusion sampling time values are adopted from Fig.6 of Wang et al. (2024)

REFERENCES

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Simon Axelrod and Rafael Gomez-Bombarelli. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):185, 2022.

Christoph Bannwarth, Sebastian Ehlert, and Stefan Grimme. Gfn2-xtb—an accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions. *Journal of chemical theory and computation*, 15 (3):1652–1671, 2019.

Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453, 2022.

Louis Bellmann, Patrick Penner, Marcus Gastreich, and Matthias Rarey. Comparison of combinatorial fragment spaces and its application to ultralarge make-on-demand compound catalogs. *Journal of Chemical Information and Modeling*, 62(3):553–566, 2022.

Gregory S Chirikjian and Alexander B Kyatkin. *Engineering applications of noncommutative harmonic analysis: with emphasis on rotation and motion groups*. CRC press, 2000.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24 (43):1–48, 2023.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

Octavian Ganea, Lagnajit Pattanaik, Connor Coley, Regina Barzilay, Klavs Jensen, William Green, and Tommi Jaakkola. Geomol: Torsional geometric generation of molecular 3d conformer ensembles. *Advances in Neural Information Processing Systems*, 34:13757–13769, 2021.

Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks, 2022. URL `https://arxiv.org/abs/2207.09453`.

Mario Geiger, Tess Smidt, Alby M., Benjamin Kurt Miller, Wouter Boomsma, Bradley Dice, Kostiantyn Lapchevskyi, Maurice Weiler, Michał Tyszkiewicz, Simon Batzner, Dylan Madisetti, Martin Uhrin, Jes Frellsen, Nuri Jung, Sophia Sanborn, Mingjian Wen, Josh Rackers, Marcel Rød, and Michael Bailey. Euclidean neural networks: e3nn, April 2022. URL `https://doi.org/10.5281/zenodo.6459381`.

Majdi Hassan, Nikhil Shenoy, Jungyoon Lee, Hannes Stark, Stephan Thaler, and Dominique Beaini. Equivariant flow matching for molecular conformer generation. In *ICML 2024 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*, 2024.

Paul CD Hawkins. Conformation generation: the state of the art. *Journal of chemical information and modeling*, 57(8):1747–1756, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. *Advances in Neural Information Processing Systems*, 35:24240–24253, 2022.

Bowen Jing, Ezra Erives, Peter Pao-Huang, Gabriele Corso, Bonnie Berger, and Tommi Jaakkola. Eigenfold: Generative protein structure prediction with diffusion models. *arXiv preprint arXiv:2304.02198*, 2023.

G Landrum. Rdkit: open-source cheminformatics http://www. rdkit. org. *Google Scholar There is no corresponding record for this reference*, 3(8), 2016.

Sangyun Lee, Zinan Lin, and Giulia Fanti. Improving the training of rectified flows. *arXiv preprint arXiv:2405.20320*, 2024.

Yi-Lun Liao, Brandon Wood, Abhishek Das, and Tess Smidt. Equiformerv2: Improved equivariant transformer for scaling to higher-degree representations. *arXiv preprint arXiv:2306.12059*, 2023.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=PqvMRDCJT9t`.

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023a.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*, 2023b.

Shitong Luo, Chence Shi, Minkai Xu, and Jian Tang. Predicting molecular conformation via dynamic graph score matching. *Advances in Neural Information Processing Systems*, 34:19784–19795, 2021.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14297–14306, 2023.

David Mohlin, Josephine Sullivan, and Gérald Bianchi. Probabilistic orientation estimation with matrix fisher distributions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4884–4893. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/33cc2b872dfe481abef0f61af181dfcf-Paper.pdf`.

Leopoldo Nachbin and Lulu Bechtolsheim. The haar integral. *(No Title)*, 1965.

Philipp Pracht, Fabian Bohle, and Stefan Grimme. Automated exploration of the low-energy chemical space with fast quantum chemical methods. *Physical Chemistry Chemical Physics*, 22(14):7169–7192, 2020.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. In *International conference on machine learning*, pp. 9558–9568. PMLR, 2021.

Gregor NC Simm and José Miguel Hernández-Lobato. A generative model for molecular distance geometry. *arXiv preprint arXiv:1909.11459*, 2019.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Ch Tsitouras. Runge–kutta pairs of order 5 (4) satisfying only the first column simplifying assumption. *Computers & Mathematics with Applications*, 62(2):770–775, 2011.

Yuyang Wang, Ahmed AA Elhag, Navdeep Jaitly, Joshua M Susskind, and Miguel Ángel Bautista. Swallowing the bitter pill: Simplified scalable conformer generation. In *Forty-first International Conference on Machine Learning*, 2024.

Lemeng Wu, Dilin Wang, Chengyue Gong, Xingchao Liu, Yunyang Xiong, Rakesh Ranjan, Raghuraman Krishnamoorthi, Vikas Chandra, and Qiang Liu. Fast point cloud generation with straight flows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9445–9454, 2023.

Minkai Xu, Wujie Wang, Shitong Luo, Chence Shi, Yoshua Bengio, Rafael Gomez-Bombarelli, and Jian Tang. An end-to-end framework for molecular conformation generation via bilevel programming. In *International conference on machine learning*, pp. 11537–11547. PMLR, 2021.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

Anthony Zee. *Group theory in a nutshell for physicists*, volume 17. Princeton University Press, 2016.

A APPENDIX

A.1 BACKGROUND AND RELATED WORK

A.1.1 GENERATIVE MODELS FOR CONFORMER GENERATION

The task of molecular conformer generation in its core is to sample from the intractable conformer distribution conditioned on the 2D molecular graph. Therefore, generative deep learning model is well-suited for such task and many methods have been proposed. Deep learning model are usually trained on datasets containing molecular conformers generated by CREST (Pracht et al., 2020) using computationally expensive semi-empirical quantum chemistry method (Bannwarth et al., 2019) under the hood. Earliest works in this field uses variational autoencoder to generate the intrinsic inter-atomic distance (Simm & Hernández-Lobato, 2019; Xu et al., 2021). Shi et al. (2021) proposed a score-matching method that learns the gradient of intrinsic atom coordinates in molecular graph. Ganea et al. (2021) started to tackle molecular conformer generation by designing a message passing neural network to predict the local 3D structure and torsion angles. Xu et al. (2022) adopted diffusion model and equivariant graph neural network to generate molecular conformers by iteratively denoising the Euclidean atom coordinates from sampled noise. Torsional diffusion (Jing et al., 2022) reduced the degree-of-freedom by refining the torsion angles of RDKit-generated (Landrum, 2016) initial conformers with a diffusion process on the hypertorus. Such design allowed torsional diffusion to significantly reduce sampling steps. One drawback of torsional diffusion is that it relies on an RDKit-generated conformer as the starting point of diffusion, which adds computational overhead to generation process. The generation quality of RDKit, especially for atom coordinates in rings, can also impact the sample quality of torsional diffusion. Molecular conformer field (MCF) proposed by Wang et al. (2024) is a recent work that leverages the scaling power of the transformer architecture (Jaegle et al., 2021) and diffusion model. MCF achieves state-of-the-art performance in molecular conformer generation by training models with tens to hundreds million of parameters to denoise the atoms' Euclidean coordinates using DDPM paradigm (Ho et al., 2020). Equivariant Transformer Flow (ET-Flow) is a concurrent work that trains a equivariant flow-matching model to generate conformers from prior distribution. By combining harmonic prior (Jing et al., 2023), flow-matching, and Kabsch alignment that reduces transport cost, ET-Flow is reported to outperform MCF on several metrics with less ODE steps.

Overall, the trade-off between conformer generation quality and speed is a prevailing issue. Specifically, semi-empirical quantum chemistry can sample very high quality conformers with high computational cost. Diffusion or flow-matching models can generate high quality conformers but the iterative ODE/SDE solving process can be slow, making them less practical for large-scale virtual screening. Cheminformatics tools such as RDKit and OMEGA are very fast but generate conformers with underwhelming diversity.

A.1.2 FLOW-MATCHING

Averaged Flow is based on Flow Matching (Lipman et al., 2023; Liu et al., 2023a; Albergo & Vanden-Eijnden, 2023), which models a probability density path $p_t(\mathbf{x}_t)$ that gradually transforms an analytically tractable noise distribution ($t = 0$) into a data distribution ($t = 1$), following a time variable $t \in [0, 1]$. Formally, the path $p_t(\mathbf{x}_t)$ corresponds to a *flow* ψ_t that pushes samples from p_0 to p_t via $p_t = [\psi]_t * p_0$, where $*$ denotes the push-forward. In practice, the flow is modelled via an ordinary differential equation (ODE) $dx_t = v_t^\theta(x_t)dt$, defined through a learnable vector field $v_t^\theta(x_t)$ with parameters θ. Initialized from noise $x_0 \sim p_0(x_0)$, this ODE simulates the flow and transforms noise into approximate data distribution samples. The probability density path $p_t(x_t)$ and the (intractable) ground-truth vector field $u_t(x_t)$ are related via the continuity equation $dp_t(x)/dt = -\nabla_x \cdot (p_t(x)u_t(x))$. To construct p_t Lipman et al. (2023) introduce a conditional probability $p_t(x|x_1)$ and conditional vector field $u_t(x|x_1)$ both related to their unconditional counterparts as follow:

$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1. \tag{FM6}$$

$$u_t(x) = \int u_t(x|x_1)\frac{p_t(x|x_1)q(x_1)}{p_t(x)}dx_1 \tag{FM8}$$

With the following simple choices of conditional probability and flow

$$p_t(x|x_1) = \mathcal{N}(x; \mu_t(x_1), \sigma_t^2(x_1)) \tag{FM10}$$

$$\psi_t(x) = \sigma_t(x_1)x + \mu_t(x_1) \tag{FM11}$$

they prove that

$$u_t(x|x_1) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)}(x - \mu_t(x_1)) + \mu_t'(x_1). \tag{FM15}$$

It is noteworthy that we refer to the linear interpolant $x_t = tx_1 - (1-t)x_0$ between the noise and data distribution as conditional optimal transport (OT) following Lipman et al. (2023).

A.1.3 Rectified flow and other distillation

With the success of denoising diffusion probabilistic models (Ho et al., 2020), many attention has been drawn to improve the sampling speed of diffusion models. DDIM (Song et al., 2020) shows that the sampling steps can be significantly reduced by formulating the sampling process as ODE solving. Knowledge distillation techniques (Meng et al., 2023; Salimans & Ho, 2022; Song et al., 2023; Song & Dhariwal, 2023) are also proposed to reduce sampling steps and accelerate generation. Rectified flow (Liu et al., 2022; Liu, 2022) is a method proposed to train the model to learn straight probability flow that bridges prior and data distribution. The *reflow* technique proposed in rectified flow can straighten the flow trajectory and reduce the transport cost, allowing very few-step generation with high quality. After reflow, the model can be further distilled to improve 1-step generation. The reflow and distillation technique has been proven effective in enabling few-step or even single-step text-to-image (Esser et al., 2024; Liu et al., 2023b) and point cloud (Wu et al., 2023) generation.

A.2 Experiments Details

A.2.1 Training algorithm

Training algorithm for *Averaged Flow* and Reflow+Distillation.

Algorithm 1 *Averaged Flow* with Reflow+Distillation Training

Require: Molecule Dataset $\mathcal{G} = [G_0, ..., G_D]$, each with conformers $\mathcal{X}^G = [X^{G,0}, ...X^{G,N}]$
Require: Learnable Velocity Field Network v^θ
1. Base SO(3) Averaged Flow Training
$t, X_0, G \sim \mathcal{U}(0,1), \mathcal{N}(0,1), \mathcal{G}$
$X_1 \sim \mathcal{X}^G$
$X_t \leftarrow t \cdot X_0 + (1-t) \cdot X_1$
$u_t(X_t) \leftarrow$ Solve closed-form Eq. 4 forX_t and t
Gradient Step -$\|v_t^\theta(X_t|G) - u_t(X_t)\|^2$
2. Reflow
$X_0' \sim \mathcal{N}(0,1)$
$X_1' \sim$ ODESolve$\big(v_t^\theta(\cdot|G), X_0'\big)$
Finetune model with coupled pair (X_0', X_1') through Eq. 7
3. Distillation
Train model with coupled pair (X_0', X_1') through Eq. 7 with $t = 0$

A.2.2 Trajectory and distribution of t

Here we are visualizing the trajectories of atoms in a molecules during 100-steps of ODE transport. Fig. 1a shows the trajectory before reflow, which demonstrate high curvature at the beginning of the transport (t close to 0). We observed such pattern in trajectory for most of molecules, leading us to sample t from exponential distribution which focus on the $t < 0$ region during the reflow. After reflow, the 100-step ODE trajectory of the same molecules much straighter (Fig. 1b).The distribution of t is visualized in Fig. 4.

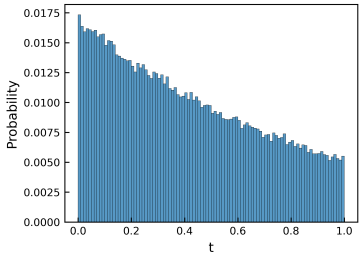

Figure 4: **The distribution of t during reflow**

A.2.3 MODEL ARCHITECTURE

The equivariant model used in this work is a modified variant (Fig. 5) of the NequIP model (Batzner et al., 2022). The model takes 4 inputs including the atomic features Z, relative distance vector between atoms $\vec{r}$, edge (bond) features e, and the flow-matching time-step t. The output model is a vector field corresponding to the probability flow at t. Compared to the original NequIP model, our variant has residue connection and equivariant layer normalization (Liao et al., 2023) after each interaction block, which we found to be highly effective in stabilizing the training of model with more than 4 layers. Bond information in the 2D molecular graph is critical inductive bias for the molecular conformer generation task. To add bond information into the model, we featurize the edges in the molecular graph and concatenate the edge features e with the radial basis embedding of relative distance vector $\vec{r}$. The concatenated message is then fed into the rotationally invariant radial function implemented as an multi-layer perceptron. To keep long-range information in the graph convolution during intermediate time-step t, we remove the envelop function from the radial basis and keep only the radial Bessel function.

For both the GEOM-Drugs and GEOM-QM9 dataset, we train a model with 6 interaction blocks. The multiplicity is set to 96 and maximum order of irreps l is 2. The radial function MLP has 2 layers and hidden dimension of 256. Molecular graph are fully-connected with non-bond as an specified bond type. The relative distance vectors are scaled down by a soft cutoff distance of 10Å and 20Å for QM9 and Drugs dataset, respectively. we used 12 Bessel radial basis functions in the model. The model is implemented using `e3nn-jax` (Geiger & Smidt, 2022; Geiger et al., 2022).

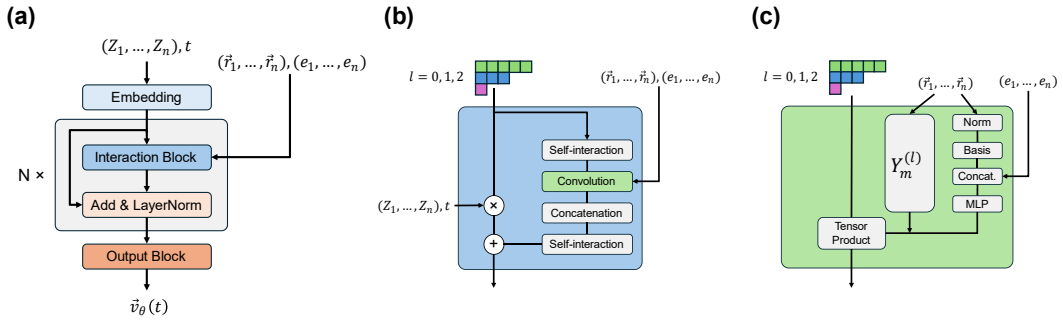

Figure 5: **Model architecture** *(a)* Overview of the modified NequIP architecture for the flow vector field prediction. *(b)* Details of the interaction block, where atomic features are mixed and refined with relative distance vectors $\vec{r}$ and edge features. *(c)* In the convolution block, a learnable radial function MLP incorporate basis embedding of $\vec{r}$ and edge features. Tensor product is used to combine the output of the MLP and the spherical harmonics $Y_m^{(l)}$ projection of $\vec{r}$.

A.2.4 DATASETS

The dataset we train and benchmark our model on are GEOM-Drugs and GEOM-QM9(Axelrod & Gomez-Bombarelli, 2022). We follow the exact splitting defined and used in previous works (Ganea et al., 2021; Jing et al., 2022; Wang et al., 2024). The train/val/test set of GEOM-Drugs contains 243473/30433/1000 molecules, respectively. The train/val/test set of GEOM-QM9 contains 106586/13323/1000 molecules, respectively.

A.2.5 MOLECULAR GRAPH FEATURIZATION

We followed the atomic featurization from GeoMol (Ganea et al., 2021). Details of the atomic featurization are included in Table. 2. Graph Laplacian positional encoding vector (Dwivedi et al., 2023) with size of 32 is concatenated with the atomic features for each atom in molecular graph. The edge features is the one-hot encoding of the bond types: {No Bond, Single Bond, Double Bond, Triple Bond, Aromatic Bond}.

Table 2: Atomic features as input to the model

Name	Description	Range
`atom_type`	Atom type	One-hot encoding of the atom type
`degree`	Number of bonded neighbors	$\{x : 0 \leq x \leq 6, x \in \mathbb{Z}\}$
`charge`	Formal charge of atom	$\{x : -1 \leq x \leq 1, x \in \mathbb{Z}\}$
`valence`	Implicit valence of atom	$\{x : 0 \leq x \leq 6, x \in \mathbb{Z}\}$
`hybridization`	Hybridization type	{sp, sp^2, sp^3, sp^3d, sp^3d^2, other}
`chirality`	Chirality Tag	{unspecified, tetrahedral CW, tetrahedral CCW, other}
`num_H`	Total number of hydrogens	$\{x : 0 \leq x \leq 8, x \in \mathbb{Z}\}$
`aromatic`	Whether on aromatic ring	{True, False}
`num_rings`	Number of rings the atom on	$\{x : 0 \leq x \leq 3, x \in \mathbb{Z}\}$
`ring_size_3-8`	Whether on ring size of 3-8	{True, False}

A.2.6 Training and sampling details

The model is trained with the *Averaged Flow* for 990 epochs on the GEOM-Drugs dataset and 1500 epochs on the GEOM-QM9 dataset using 2 NVIDIA A5880 GPUs. We used dynamic graph batching to maixmize the utilization of GPU memory and reduce JAX compilation time. The effective average batch size is 208 and 416 for Drugs and QM9 dataset, respectively. We used Adam optimizer with learning rate of $1\mathrm{e}{-2}$, which decays to $5\mathrm{e}{-3}$ after 600 epochs and to $1\mathrm{e}{-3}$ after 850 epochs. We selected the top-30 conformers for model training.

To sample coupled (X_0', X_1') for reflow and distillation, we generate 32 noise-sample pairs for each molecule in the Drugs and 64 for each molecule in the QM9 dataset. The reflow and distillation are done using 4 NVIDIA A100 GPUs and doubling the effective batch size of each dataset. During the reflow stage, the model is finetuned for 870 epochs on Drugs and 1530 epochs on QM9. We used Adam optimizer with learning rate of $5\mathrm{e}{-3}$, which decays to $2.5\mathrm{e}{-3}$ after 450 epochs for Drugs (500 epochs for QM9), and to $5\mathrm{e}{-4}$ after 650 epochs for Drugs (900 epochs for QM9). During the distillation stage, the model is finetuned for 450 epochs on Drugs and 1200 epochs on QM9. We used Adam optimizer with learning rate of $2\mathrm{e}{-3}$, which decays to $1\mathrm{e}{-3}$ after 300 epochs for Drugs (500 epochs for QM9), and to $2\mathrm{e}{-4}$ after 450 epochs for Drugs (900 epochs for QM9). We used exponential moving average (EMA) with a decay of 0.999 for all Averaged Flow, reflow, and distillation training.

To generate the benchmark results of AvgFlow (Table. 3, Table. 3, and Table. 4), we use the Tsitouras' 5/4 solver (Tsitouras, 2011) implemented in the `diffrax` package with adaptive stepping. The relative tolerance and absolute tolerance are set to $1\mathrm{e}{-5}$ and $1\mathrm{e}{-6}$ when sampling for GEOM-Drugs, respectively. The relative tolerance and absolute tolerance are both set to $1\mathrm{e}{-5}$ when sampling for GEOM-QM9. Euler solver is always used for $\text{AvgFlow}_{\text{Reflow}}$ and $\text{AvgFlow}_{\text{Distill}}$. When comparing the effect of ODE steps to models, Euler solver is used.

A.3 Evaluation Details and Results

A.3.1 Evaluation Mtrics

We report the average minimum RMSD (AMR) between ground truth and generated structures, and Coverage for Recall and Precision. Coverage is defined as the percentage of conformers with a minimum error under a specified AMR threshold. Recall matches each ground truth structure to its closest generated structure, and Precision measures the overall spatial accuracy of the each generated structure. Following Ganea et al. (2021); Jing et al. (2022), we generate two times the number of ground truth structures for each molecule. More formally, for $K = 2L$, let $\{C_l^*\}_{l\in[1,L]}$ and $\{C_k\}_{k\in[1,K]}$ respectively be the sets of ground truth and generated structures:

$$\begin{aligned}
\text{COV-Precision} &:= \frac{1}{K}\left|\{k \in [1..K] : \min_{l\in[1..L]} \mathrm{RMSD}(C_k, C_l^*) < \delta\}\right|, \\
\text{AMR-Precision} &:= \frac{1}{K}\sum_{k\in[1..K]} \min_{l\in[1..L]} \mathrm{RMSD}(C_k, C_l^*),
\end{aligned} \tag{8}$$

where δ is the coverage threshold. δ is set to 0.75Å for the Drugs and 0.5Å for the QM9 dataset. The recall metrics are obtained by swapping ground truth (K) and generated conformers (L) in the above equations.

A.3.2 GEOM-QM9 BENCHMARK RESULTS

Table 3: Quality of ML generated conformer ensembles for GEOM-QM9 ($\delta = 0.5$Å) test set in terms of Coverage (COV) and Average Minimum RMSD (AMR). Bolded results are the best. Baseline values are taken from the corresponding papers. *Due to the use of adaptive step size, the number of steps of AvgFlow is an average value over all test set molecules.

		Recall				Precision			
		COV (%) ↑		AMR (Å) ↓		COV (%) ↑		AMR (Å) ↓	
Method	Step	Mean	Med	Mean	Med	Mean	Med	Mean	Med
RDKit	-	85.1	100	0.235	0.199	86.8	100	0.232	0.205
OMEGA	-	85.5	100	0.177	0.126	82.9	100	0.224	0.186
GeoMol	-	91.5	100	0.225	0.193	87.6	100	0.27	0.241
Tor. Diff.	20	92.8	100	0.178	0.147	92.7	100	0.221	0.195
ET-Flow-SS (8.3M)	50	95.0	100	**0.083**	**0.035**	91.0	100	**0.116**	**0.047**
MCF-B (64M)	1000	95.0	100	0.103	0.044	**93.7**	100	0.119	0.055
AvgFlow (4.7M)	60*	**96.4**	100	0.089	0.042	92.8	100	0.132	0.084
$\text{AvgFlow}_{\text{Reflow}}$ (4.7M)	2	95.9	100	0.151	0.104	87.7	100	0.236	0.207
$\text{AvgFlow}_{\text{Distill}}$ (4.7M)	1	95.1	100	0.220	0.195	84.8	100	0.304	0.283

A.3.3 SAMPLING TIME BENCHMARK

Table 4: **Sampling time and performance comparison between models.** Bolded results are the best.

			Recall		Precision	
			COV (%) ↑	AMR (Å) ↓	COV (%) ↑	AMR (Å) ↓
Method	Step	Time (ms) ↓	Mean	Mean	Mean	Mean
Tor. Diff.	5	128	58.4	0.691	36.4	0.973
ET-Flow	5	106	**77.8**	**0.476**	**74.0**	**0.550**
MCF-S	3	57.3	56.9	0.725	30.8	1.014
MCF-B	3	102	66.5	0.665	39.9	0.951
MCF-L	3	134	71.6	0.636	45.3	0.686
$\text{AvgFlow}_{\text{Reflow}}$	2	**2.68**	64.2	0.663	43.1	0.871

A.4 AVERAGED FLOW DETAILS

A.4.1 PYTHON IMPLEMENTATION

Listing 1: Averaged Flow

```
def avg_harmonic_flow(
    t: jax.Array,  # []
    x: jax.Array,  # [num_nodes, 3]
    x1: jax.Array,  # [num_conformers, num_nodes, 3]
    edges: jax.Array,  # [2, num_edges]
    weights: jax.Array | None = None,  # [num_conformers]
    sigma0: jax.Array = 1.0,
    sigma1: jax.Array = 0.0,
) -> jax.Array:
    degree = jnp.bincount(edges[0], length=x.shape[0])

    def metric(x, y):
        # x and y have shape [num_nodes]
        sigma_t = (1 - t) * sigma0 + t * sigma1
        laplacian = (
            jnp.sum(degree * x * y)
            - jnp.sum(x[edges[0]] * y[edges[1]])
            - jnp.sum(x[edges[1]] * y[edges[0]])
        )
        return laplacian / sigma_t**2

    avg_x1 = avg_target(x, x1, t, metric, weights)
```

```
    return (avg_x1 - x) / (1 - t)


def avg_flow(
    t: jax.Array,  # []
    x: jax.Array,  # [num_nodes, 3]
    x1: jax.Array,  # [num_conformers, num_nodes, 3]
    weights: jax.Array | None = None,  # [num_conformers]
    sigma0: jax.Array = 1.0,
    sigma1: jax.Array = 0.0,
) -> jax.Array:
    def metric(x, y):
        # x and y have shape [num_nodes]
        sigma_t = (1 - t) * sigma0 + t * sigma1
        return jnp.dot(x, y) / sigma_t**2

    avg_x1 = avg_target(x, x1, t, metric, weights)

    return (avg_x1 - x) / (1 - t)


def avg_target(
    x: jax.Array,  # [n, 3]
    targets: jax.Array,  # [num_targets, n, 3]
    t: jax.Array,  # []
    metric: Callable[[jax.Array, jax.Array], jax.Array],
    weights: jax.Array | None = None,  # [num_targets]
) -> jax.Array:
    num_targets, n, _ = targets.shape
    assert x.shape == (n, 3)
    assert targets.shape == (num_targets, n, 3)
    assert t.shape == ()

    def outer(u, v):  # [n, 3] x [n, 3] -> [3, 3]
        return jax.vmap(jax.vmap(metric, (None, -1)), (-1, None))(u, v)

    def inner(u, v):  # [n, 3] x [n, 3] -> []
        return jnp.sum(jax.vmap(metric, (-1, -1))(u, v))

    def logZ(alpha):  # [n, 3] -> []
        def f(target):  # [n, 3] -> []
            return (
                logcF(t * outer(target, x) + target.T @ alpha)
                - (inner(x, x) + t**2 * inner(target, target)) / 2
            )

        return logsumexp(jax.vmap(f)(targets), weights)

    return jax.grad(logZ)(jnp.zeros_like(x))


def logsumexp(a: jax.Array, weights: jax.Array | None = None) -> jax.Array:
    assert a.ndim == 1
    assert weights is None or weights.shape == a.shape
    where = (weights > 0) if weights is not None else None

    amax = jnp.max(a, where=where, initial=-jnp.inf)
    amax = jax.lax.stop_gradient(
        jax.lax.select(jnp.isfinite(amax), amax, jax.lax.full_like(amax, 0))
    )
    if where is not None:
        a = jnp.where(where, a, amax)
    exp_a = jax.lax.exp(jax.lax.sub(a, amax))
    if weights is not None:
        exp_a = exp_a * weights
    sumexp = exp_a.sum(where=where)
    return jax.lax.add(jax.lax.log(sumexp), amax)


# All the code below is adapted from a PyTorch code from David Mohlin, Gerald Bianchi and Josephine Sullivan

def logcF(F: jax.Array) -> jax.Array:
    # \log \int_{SO(3)} \exp(\text{tr}(F^T R)) dR
    assert F.shape == (3, 3)
    return logcf(*signed_svdvals(F))


def signed_svdvals(F: jax.Array) -> jax.Array:
    u, s, vh = jnp.linalg.svd(F, full_matrices=False)
    u, vh = jax.lax.stop_gradient((u, vh))
    sign = jnp.sign(jnp.linalg.det(u @ vh))
    return s.at[-1].mul(sign)


@jax.custom_vjp
def logcf(s1: jax.Array, s2: jax.Array, s3: jax.Array) -> jax.Array:
    # assume s1 >= s2 >= s3
    s1, s2, s3 = jnp.asarray(s1), jnp.asarray(s2), jnp.asarray(s3)
    return s1 + s2 + s3 + jnp.log(factor(False, s1, s2, s3))


def _logcf_fwd(
```

```
    s1: jax.Array, s2: jax.Array, s3: jax.Array
) -> tuple[jax.Array, tuple[jax.Array, jax.Array]]:
    # s1 >= s2 >= s3
    f = factor(False, s1, s2, s3)
    return s1 + s2 + s3 + jnp.log(f), (s1, s2, s3, f)


def _logcf_bwd(res: tuple[jax.Array, ...], grad: jax.Array) -> tuple[jax.Array]:
    s1, s2, s3, f = res
    # s1 >= s2 >= s3
    assert s1.shape == ()
    assert f.shape == ()
    assert grad.shape == ()
    g1 = grad * factor(True, s1, s2, s3) / f
    g2 = grad * factor(True, s2, s1, s3) / f
    g3 = grad * factor(True, s3, s1, s2) / f
    return g1, g2, g3


logcf.defvjp(_logcf_fwd, _logcf_bwd)


def factor(add_x: bool, s1: jax.Array, s2: jax.Array, s3: jax.Array) -> jax.Array:
    def f(x):
        i0 = (1.0 - 2 * x) if add_x else 1.0
        i1 = bessel0((s2 - s3) * x)
        i2 = bessel0((s2 + s3) * (1 - x))
        return i0 * i1 * i2

    tiny = jnp.finfo(s1.dtype).tiny
    a = 2 * (s3 + s1)

    # a non zero:
    a_ = jnp.maximum(a, 0.5)
    y = jnp.linspace(tiny + jnp.exp(-a_), 1.0, 512)
    r1 = jnp.trapezoid(jax.vmap(f)(-jnp.log(y) / a_), y) / a_

    # a (close to) zero:
    x = jnp.linspace(0.0, 1.0, 512, dtype=s1.dtype)
    r2 = jnp.trapezoid(jax.vmap(f)(x) * jnp.exp(-a * x), x)

    return jnp.where(a > 1.0, r1, r2)


def bessel0(x: jax.Array) -> jax.Array:
    p = [1.0, 3.5156229, 3.0899424, 1.2067492, 0.2659732, 0.360768e-1, 0.45813e-2]
    bessel0_a = jnp.array(p[::-1], dtype=x.dtype)

    p = [0.39894228, 0.1328592e-1, 0.225319e-2, -0.157565e-2, 0.916281e-2]
    p += [-0.2057706e-1, 0.2635537e-1, -0.1647633e-1, 0.392377e-2]
    bessel0_b = jnp.array(p[::-1], dtype=x.dtype)

    abs_x = jnp.abs(x)
    x_lim = 3.75

    def w(x, y):
        return jnp.where(abs_x <= x_lim, x, y)

    abs_x_ = w(x_lim, abs_x)

    return w(
        jnp.polyval(bessel0_a, w(abs_x / x_lim, 1.0) ** 2) * jnp.exp(-abs_x),
        jnp.polyval(bessel0_b, w(1.0, x_lim / abs_x_)) / jnp.sqrt(abs_x_),
    )
```

A.4.2 Speed Benchmark

We benchmarked the time used by our Python implementation to solve the *Averaged Flow* objective for batched graphs. Each graph is set to have 50 nodes (the average number of atoms in GEOM-Drugs molecules is 44). The benchmark is done on a single NVIDIA A5880 GPU.

Table 5: Computation time of *Averaged Flow* on batched graphs (50 nodes per graph). Unit is in ms. N_{batch} is the number of graphs in a batch and $N_{\text{conformer}}$ is number of conformers used in *Averaged Flow* solving.

N_{batch} \ $N_{\text{conformer}}$	1	10	100	1000
1	0.6	0.5	0.5	0.6
10	0.5	0.5	0.6	1.0
100	0.5	0.6	1.1	7.6
1000	0.5	0.9	7.5	73.5