Meta Flow Matching: Integrating Vector Fields on the Wasserstein Manifold

Lazar Atanackovic [1 2 *] **Xi Zhang** [3 4 *] **Brandon Amos** [5] **Mathieu Blanchette** [3 4] **Leo J. Lee** [1 2] **Yoshua Bengio** [3 6 7]
Alexander Tong [3 6] **Kirill Neklyudov** [3 6]

Abstract

Numerous biological and physical processes can be modeled as systems of interacting samples evolving continuously over time, e.g. the dynamics of communicating cells or physical particles. Learning the dynamics of such systems is essential for predicting the temporal evolution of populations across novel samples and unseen environments. Flow-based models allow for learning these dynamics at the population level — they model the evolution of the entire distribution of samples. However, standard flow-based models are limited to a single initial population and a set of predefined conditions which describe different dynamics. We propose *Meta Flow Matching* (MFM), a practical approach to integrating along vector fields on the Wasserstein manifold by amortizing the flow model over the initial populations. We demonstrate that MFM improves the prediction of individual treatment responses on a large scale multi-patient single-cell drug screen dataset.

1. Introduction

Understanding and modeling the dynamics of many-body problems is a central challenge across the natural sciences. In the field of cell biology, a central focus is the understanding of the dynamic processes that cells undergo in response to their environment, and in particular their response and interaction with other cells. Cells communicate with one other in close proximity using *cell signaling*, exerting influence over each other's trajectories (Armingol et al., 2020; Goodenough and Paul, 2009). This signaling presents an obstacle for modeling, because cell dynamics cannot be modeled in isolation, and must take into account the interaction between populations of cells, but is nonetheless essential for understanding and eventually controlling cell dynamics during development (Gulati et al., 2020; Rizvi et al., 2017), in diseased states (Molè et al., 2021; Binnewies et al., 2018; Zeng and Dai, 2019; Chung et al., 2017), and in response to perturbations (Ji et al., 2021; Peidli et al., 2024).

Single-cell sequencing has been used to great effect to understand the heterogeneity in cell systems, they are also destructive, making longitudinal measurements—measurements that track a single cell over time—extremely difficult. Instead, most approaches model cell dynamics at the population level (Hashimoto et al., 2016; Weinreb et al., 2018; Schiebinger et al., 2019; Tong et al., 2020; Neklyudov et al., 2022; Bunne et al., 2023a). These approaches involve the formalisms of optimal transport (Villani, 2009; Peyré and Cuturi, 2019) and generative modeling (De Bortoli et al., 2021; Lipman et al., 2023) methods, which learn a map between empirical measures. While these methods are able to model the dynamics of the population, they are fundamentally limited in that they model the evolution of cells as independent particles evolving according to a shared dynamical system. Furthermore, these models can be trained to match any given set of measures, but they are restricted to modeling of a single population and can at best condition on a number of different dynamics that is available in the training data.

To address this we propose *Meta Flow Matching* (MFM) — the amortization of the Flow Matching generative modeling framework (Lipman et al., 2023) over the input measures. In practice, our method can be used to predict the time-evolution of distributions from a given dataset of the time-evolved examples. Namely, we assume that the collected data undergoes a universal developmental process, which depends only on the population itself as in the setting of the interacting particles or communicating cells.

MFM was evaluated on two applications. First on a synthetic task of denoising letters. For this task, MFM is able to generalize the denoising process to letters in unseen orientations where a standard flow matching approach cannot. Next, we explore how MFM can be applied to model single-

[*]Joint first authorship [1]University of Toronto [2]Vector Institute [3]Mila - The Quebec AI Institute [4]McGill University [5]Meta FAIR [6]Université de Montréal [7]CIFAR Fellow. Correspondence to: Lazar Atanackovic <l.atanackovic@mail.utoronto.ca>.

International Conference on Machine Learning Workshop on Geometry-grounded Representation Learning and Generative Modeling, Vienna, Austria. 2024.

cell perturbation data (Ji et al., 2021; Peidli et al., 2024). MFM was evaluated on predicting the response of patient-derived cells to chemotherapy treatments in a recently published large scale single-cell drug screening dataset where there are known to be patient-specific responses (Ramos Zapatero et al., 2023). We demonstrate that Meta Flow Matching can successfully predict the development of cell populations on replicated experiments, and, most importantly, it generalizes to previously unseen patients, thus, capturing the patient-specific response to the treatment.

2. Meta Flow Matching

For a detailed description of Flow Matching, Conditional Flow Matching and related work, see appendix B and C.

In this paper, we propose the amortization of the Flow Matching framework over the marginal distributions. Our model is based on the outstanding ability of the Flow Matching framework to learn the push-forward map for any joint distribution $(x_0; x_1)$ given empirically. For the given joint $(x_0; x_1)$, we denote the solution of the Flow Matching optimization problem as follows (L_{GFM} defined in appendix B).

$$v_t(\ ;\ ) = \operatorname*{argmin}_{v_t} L_{GFM}(v_t(\ );\ (x_0; x_1)): \quad (1)$$

Analogously to the amortized optimization (Chen et al., 2022; Amos et al., 2023), we aim to learn the model that outputs the solution of Eq. (1) based on the input data sampled from , i.e.

$$v_t(\ ; '(\ )) = v_t(\ ;\ ); \quad (2)$$

where $'(\ )$ is the embedding model of and the joint density $(\ j\ c)$ is generated using some unknown measure of the conditional variables c $p(c)$.

2.1. Modeling Process in Natural Sciences as Vector Fields on the Wasserstein Manifold

We believe that numerous biological and physical processes cannot be modeled via the vector field propagating the population samples independently. Thus, we propose to model these processes as families of conditional vector fields where we amortize the conditional variable by embedding the population via a Graph Neural Network (GNN).

To provide the reader with the necessary intuition, we are going to use the geometric formalism developed by Otto (2001). That is, time-dependent densities $p_t(x_t)$ define absolutely-continuous curves on the 2-Wasserstein space of distributions $P_2(X)$ (Ambrosio et al., 2008). The tangent space of this manifold is defined by the gradient flows $S_t = \{\nabla s_t \mid s_t : X \to R\}$ on the state space X. In the Flow Matching context, we are going to refer to the tangent vectors as vector fields since one can always project the vector field onto the tangent space by parameterizing it as a gradient flow (Neklyudov et al., 2022).

Under the geometric formalism of the 2-Wasserstein manifold, Flow Matching can be considered as learning the tangent vectors $v_t(\ )$ along the density curve $p_t(x_t)$ defined by the sampling process in Eq. (10) (see the left panel in Fig. 2). Furthermore, the conditional generation processes $p_t(x_t \mid c)$ would be represented as a finite set of curves if c is discrete (e.g. class-conditional generation of images) or as a family of curves if c is continuous (see the middle panel in Fig. 2).

Finally, one can define a vector field on the 2-Wasserstein manifold via the continuity equation with the vector field $v_t(x; p_t(x))$ on the state space X that depends on the current density $p_t(x)$ or its derivatives. Below we give two examples of processes defined as vector fields on the 2-Wasserstein manifold.

We believe that using the information about the current or the initial density is crucial for the modeling of time-evolution of densities in natural processes, to capture this type of dependency one can model the change of the density as the following Cauchy problem

$$\frac{\partial p_t(x)}{\partial t} = -\langle \nabla_x; p_t(x) v_t(x; p_t)\rangle; \quad p_{t=0}(x) = p_0(x); \quad (3)$$

where the state-space vector field $v_t(x; p_t)$ depends on the density p_t.

2.2. Integrating Vector Fields on the Wasserstein Manifold via Meta Flow Matching

Consider the dataset of joint populations $D = \{(x_0; x_1 \mid i)\}_i$, where, to simplify the notation, we associate every i-th population with its density $(\ j\ i)$ and the conditioning variable here is the index of this population in the dataset. We make the following assumptions regarding the ground truth sampling process (i) we assume that the starting marginals $p_0(x_0 \mid i) = \int dx_1\ (x_0; x_1 \mid i)$ are sampled from some unknown distribution that can be parameterized with a large enough number of parameters (ii) the endpoint marginals $p_1(x_1 \mid i) = \int dx_0\ (x_0; x_1 \mid i)$ are obtained as push-forward densities solving the Cauchy problem in Eq. (3), (iii) there exists unique solution to this Cauchy problem.

One can learn a joint model of all the processes from the dataset D using the conditional version of the Flow Matching algorithm (see Appendix B.2) where the population index i plays the role of the conditional variable. However, obviously, such a model will not generalize beyond the considered data D and unseen indices i. We illustrate this empirically in Section 3.

To be able to generalize to previously unseen populations with unknown density, we propose learning the density-dependent vector field motivated by Eq. (3). That is, we propose to use an embedding function $\psi : P_2(X) \to \mathbb{R}^m$ to embed the starting marginal density p_0, which we then input into the vector field model and minimize the following objective over θ

$$L_{MFM}(\theta, \psi) = \mathbb{E}\left[\frac{\partial}{\partial t} f_t(x_0, x_1) - v_t(f_t(x_0, x_1), t, \psi(p_0); \theta)\right]^2, \quad (4)$$

Where the expectation is taken with respect to t, i; and $(x_0, x_1 | i)$ with t, i being distributed uniformly with respect to their support. Note that the initial density p_0 is enough to predict the push-forward density p_1 since the Cauchy problem for Eq. (3) has a unique solution. The embedding function $\psi(p_0)$ can take different forms, e.g. it can be the density value $\psi(p_0) = p_0(\cdot)$, which is then used inside the vector field model to evaluate at the current point; a kernel density estimator; or a parametric model taking the samples from this density as an input.

Proposition 1. Meta Flow Matching recovers the Conditional Generation via Flow Matching when the conditional dependence of the marginals $p_0(x_0 | c) = \int dx_1\ \pi(x_0, x_1 | c)$ and $p_1(x_1 | c) = \int dx_0\ \pi(x_0, x_1 | c)$ and the distribution $p(c)$ are known, i.e. there exist $\psi : P_2(X) \to \mathbb{R}^m$ such that $L_{MFM}(\theta) = L_{CGFM}(\theta)$.

Proof. Indeed, sampling from the dataset $\mathcal{D}$ becomes sampling of the conditional variable $c \sim p(c)$ and the embedding function becomes $\psi(p_0(\cdot | c)) = c$. □

Furthermore, for the parametric family of the embedding models $\psi(p_t; \omega)$, we show that the parameters ω can be estimated by minimizing the objective in Eq. (4) in the joint optimization with the vector field parameters θ. We formalize this statement in the following theorem.

Theorem 1. Consider a dataset of populations $\mathcal{D} = \{(x_0, x_1 | i)\}_i$ generated from some unknown conditional model $\pi(x_0, x_1 | c)p(c)$. Then the following objective

$$L(\theta, \omega) = \mathbb{E}_{t; p(c); p_t(x_t | c)}\left[v_t(x_t | c) - v_t(x_t, t, \psi(p_0; \omega); \theta)\right]^2 \quad (5)$$

is equivalent to the Meta Flow Matching objective $L_{MFM}(\theta, \omega)$

$$\mathbb{E}_{t; i; \pi(x_0, x_1 | i)}\left[\frac{\partial}{\partial t} f_t(x_0, x_1) - v_t(f_t(x_0, x_1), t, \psi(p_0; \omega); \theta)\right]^2 \quad (6)$$

up to an additive constant.

Proof. We postpone the proof to Appendix E. □

2.3. Learning Population Embeddings via Graph Neural Networks (GNNs)

In many applications, the populations $\mathcal{D} = \{(x_0, x_1 | i)\}_{i=1}^N$ are given as empirical distributions, i.e. they are represented as samples from some

$$\{f(x_0^j, x_1^j)\}_{j=1}^{N_i}, \quad (x_0^j, x_1^j) \sim \pi(x_0, x_1 | i), \quad (7)$$

where N_i is the size of the i-th population. We use this model in our synthetic experiments in Section 3.1.

Since the only available information about the populations is samples, we propose learning the embedding of populations via a parametric model $\psi(p_0; \omega)$, i.e.

$$\psi(p_0; \omega) = \psi\left(\{x_0^j\}_{j=1}^{N_i}; \omega\right), \quad (x_0^j, x_1^j) \sim \pi(x_0, x_1 | i). \quad (8)$$

For this purpose, we employ GNNs, which recently have been successfully applied for simulation of complicated many-body problems in physics (Sanchez-Gonzalez et al., 2020). To embed a population $\{x_0^j\}_{j=1}^{N_i}$, we create a k-nearest neighbour graph G_i based on the metric in the state-space X, input it into a GNN, which consists of several message-passing iterations (Gilmer et al., 2017) and the final average-pooling across nodes to produce the embedding vector. Finally, we update the parameters of the GNN jointly with the parameters of the vector field to minimize the loss function in Eq. (6).

3. Experiments

To show the effectiveness of MFM to generalize under previously unseen populations for the task population prediction, we consider the following experimental setting: a synthetic experiment with well defined coupled populations. To quantify model performance, we consider three distributional distances metrics: the 1-Wasserstein distance (W_1), 2-Wasserstein (W_2) distance, and the radial basis kernel maximum-mean-discrepancy (MMD) distance (Gretton et al., 2012). We parameterize all vector field models $v_t(\cdot | \psi(p_0); \theta)$ using a Multi-Layer Perceptron (MLP). For MFM, we additionally parameterize $\psi(p_t; \omega; k)$ using a Graph Convolutional Network (GCN) with k-nearest neighbour graph edge pooling layer. We include details regarding model hyperparameters, training/optimization, and implementation in Appendix F and Appendix F.2. The results for all the models are averaged over three random seeds. For more details on datasets, see Appendix D.

3.1. Synthetic Experiment

We trained FM, CGFM and 4 variants of MFM of varying k for the GCN population embedding model $\psi(p_t; \omega; k)$. When $k = 0$, $\psi(p_t; \omega; k)$ becomes identical to the DeepSets model (Zaheer et al., 2017). We compare MFM to Flow-Matching (FM) and Conditional Generation via Flow-Matching (CGFM). FM does not have access to conditional information; hence will only learn an aggregated lens of the distribution dynamics and will not be able to fit the training

Table 1: Experimental results on the organoid drug-screen dataset for population prediction of treatment response across patients and populations. Results shown in this table are broken out in Table 3. Results are reported for models trained on data embedded into 10 principle components. We report the the 1-Wasserstein (W_1), (2-Wasserstein (W_2), and the maximum-mean-discrepancy (MMD) distributional distances. We consider two settings for MFM with varying nearest-neighbours parameter.

	Train						Test					
	W_1		W_2		MMD (10^{-3})		W_1		W_2		MMD (10^{-3})	
FM	1.995	0.138	2246	0.193	687	2.65	2607	0.028	2947	0.050	2158	1.02
ICNN	2.163	0.067	2367	0.070	19267	4.22	2702	0.027	2996	0.033	45267	19.14
CGFM	1.773	0.072	1.954	0.092	3.03	0.69	2.675	0.019	2938	0.020	2375	0.61
MFM (k = 0)	1.863	0.056	2048	0.063	501	0.53	2393	0.160	2685	0.122	1666	1.99
MFM (k = 10)	1.881	0.071	2074	0.091	525	0.78	2.326	0.072	2.610	0.073	14.30	2.27

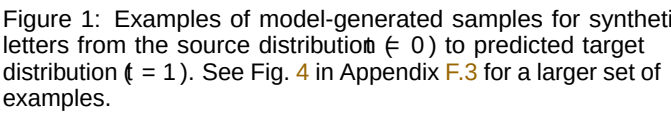

Figure 1: Examples of model-generated samples for synthetic letters from the source distribution ($t = 0$) to predicted target distribution ($t = 1$). See Fig. 4 in Appendix F.3 for a larger set of examples.

data, and consequently won't generalize to the test conditions. For the training data, CGFM vector field model takes in the distribution index x as a one-hot input condition. On the test set, since none of these indices is present, we input the normalized constant vector, which averages the learned embeddings of the indices. Because of this, CGFM will fit the training data, however, will not be able to generalize to the unseen condition in the test dataset. Note that the CGFM can be viewed as an *idealized* model for the train data since it gets perfect information regarding the population conditions. We use CGFM to assess if other models are fitting the data. For MFM, we expect to both fit the training data and generalize to unseen distributional conditions.

In Fig. 1, we observe that indeed FM fails to adequately learn to sample from $p_1(x_1 \mid i)$ in the training set, and likewise fails to generalize, while CGFM is able to effectively sample from $p_1(x_1 \mid i)$ in the training set, but fails to generalize. We report results for the synthetic experiment in Table 2. As expected, CGFM fits the training data, however, fails to generalize beyond its set of training conditions. In contrast, we see that MFM is able to both fit the training data (approaching the performance of CGFM) while also generalizing to the unseen test distributions. FM fails to fit the train data and fails to generalize under the test conditions. Interestingly, although MFM performs better for certain values of k versus others, overall performance does not vary significantly for the range considered.

3.2. Experiments on Organoid Drug-screen Data

We show results for generalization across patients in Table 1. Similar to the replicates data setting, we observe that CGFM fits the training data, but does not generalize to the test replicates. Likewise, the FM and ICNN models fail to fit the train data, relative to CGFM, and also fail to generalize. MFM (k = 10) performs best on generalization to unseen replicates. We include results reported for the separate cell cultures in Table 3 in Appendix F.3.

Through the biological and synthetic experiments, we have shown that MFM is able to generalize to unseen distributions/populations. The implication of our results suggest that MFM can learn population dynamics in unseen environments. This works towards a model where it is possible to predict and design an individualized treatment regimen for each patient based on their individual characteristics and tumor microenvironment.

4. Conclusion

Our paper highlights the significance of modeling dynamics based on the entire distribution. While flow-based models offer a promising avenue for learning dynamics at the population level, they were previously restricted to a single initial population and predefined conditions. In this paper, we introduce Meta Flow Matching (MFM) as a practical solution to address these limitations. MFM leverages graph neural networks to embed the initial population, enabling the model to generalize over various initial distributions. MFM opens up new possibilities for understanding complex phenomena that emerge from interacting systems in biological and physical systems. In practice, we demonstrate that MFM learns meaningful embeddings of single-cell populations along with the developmental model of these populations. Moreover, our empirical study demonstrates the possibility of modeling patient-specific response to treatments via meta-learning.

Acknowledgments

The authors acknowledge funding from UNIQUE, CIFAR, NSERC, Intel, and Samsung. The research was enabled in part by by the Province of Ontario and companies sponsoring the Vector Institute (http://vectorinstitute.ai/partners/), the computational resources provided by the Digital Research Alliance of Canada (https://alliancecan.ca), Mila (https://mila.quebec), and NVIDIA.

References

Albergo, M. S. and Vanden-Eijnden, E. (2022). Building normalizing flows with stochastic interpolants. arXiv preprint arXiv:2209.15571.

Ambrosio, L., Gigli, N., and Savaré, G. (2008). Gradient flows: in metric spaces and in the space of probability measures. Springer Science & Business Media.

Amos, B., Cohen, S., Luise, G., and Redko, I. (2022). Meta optimal transport. arXiv preprint arXiv:2206.05262.

Amos, B. et al. (2023). Tutorial on amortized optimization. Foundations and Trends® in Machine Learning, 16(5):592–732.

Armingol, E., Officer, A., Harismendy, O., and Lewis, N. E. (2020). Deciphering cell–cell interactions and communication from gene expression. Nature Reviews Genetics, 22(2):71–88.

Binnewies, M., Roberts, E. W., Kersten, K., Chan, V., Fearon, D. F., Merad, M., Coussens, L. M., Gabrilovich, D. I., Ostrand-Rosenberg, S., Hedrick, C. C., Vonderheide, R. H., Pittet, M. J., Jain, R. K., Zou, W., Howcroft, T. K., Woodhouse, E. C., Weinberg, R. A., and Krummel, M. F. (2018). Understanding the tumor immune microenvironment (time) for effective therapy. Nature Medicine, 24(5):541–550.

Bunne, C., Krause, A., and Cuturi, M. (2022a). Supervised training of conditional monge maps. Advances in Neural Information Processing Systems, 35:6859–6872.

Bunne, C., Krause, A., and Cuturi, M. (2022b). Supervised training of conditional monge maps.

Bunne, C., Stark, S. G., Gut, G., Del Castillo, J. S., Levesque, M., Lehmann, K.-V., Pelkmans, L., Krause, A., and Rätsch, G. (2023a). Learning single-cell perturbation responses using neural optimal transport. Nature Methods, 20(11):1759–1768.

Bunne, C., Stark, S. G., Gut, G., del Castillo, J. S., Levesque, M., Lehmann, K.-V., Pelkmans, L., Krause, A., and Rätsch, G. (2023b). Learning single-cell perturbation responses using neural optimal transport. Nature Methods, 20(11):1759–1768.

Chen, S., Rivaud, P., Park, J. H., Tsou, T., Charles, E., Haliburton, J. R., Pichiorri, F., and Thomson, M. (2020). Dissecting heterogeneous cell populations across drug and disease conditions with popalign. Proceedings of the National Academy of Sciences, 117(46):28784–28794.

Chen, T., Chen, X., Chen, W., Heaton, H., Liu, J., Wang, Z., and Yin, W. (2022). Learning to optimize: A primer and a benchmark. Journal of Machine Learning Research, 23(189):1–59.

Chung, W., Eum, H. H., Lee, H.-O., Lee, K.-M., Lee, H.-B., Kim, K.-T., Ryu, H. S., Kim, S., Lee, J. E., Park, Y. H., Kan, Z., Han, W., and Park, W.-Y. (2017). Single-cell rna-seq enables comprehensive tumour and immune cell profiling in primary breast cancer. Nature Communications, 8(1).

Dao, Q., Phung, H., Nguyen, B., and Tran, A. (2023). Flow matching in latent space. arXiv preprint arXiv:2307.08698.

De Bortoli, V., Thornton, J., Heng, J., and Doucet, A. (2021). Diffusion schrödinger bridge with applications to score-based generative modeling. Advances in Neural Information Processing Systems, 34:17695–17709.

Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds.

Frangieh, C. J., Melms, J. C., Thakore, P. I., Geiger-Schuller, K. R., Ho, P., Luoma, A. M., Cleary, B., Jerby-Arnon, L., Malu, S., Cuoco, M. S., Zhao, M., Ager, C. R., Rogava, M., Hovey, L., Rotem, A., Bernatchez, C., Wucherpfennig, K. W., Johnson, B. E., Rozenblatt-Rosen, O., Schadendorf, D., Regev, A., and Izar, B. (2021). Multimodal pooled perturb-cite-seq screens in patient models define mechanisms of cancer immune evasion. Nature Genetics, 53(3):332–341.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In International conference on machine learning, pages 1263–1272. PMLR.

Goodenough, D. A. and Paul, D. L. (2009). Gap junctions. Cold Spring Harb Perspect Biol, 1(1):a002576.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. The Journal of Machine Learning Research, 13(1):723–773.

Gulati, G. S., Sikandar, S. S., Wesche, D. J., Manjunath, A., Bharadwaj, A., Berger, M. J., Ilagan, F., Kuo, A. H.,

Hsieh, R. W., Cai, S., Zabala, M., Scheeren, F. A., Lobo, N. A., Qian, D., Yu, F. B., Dirbas, F. M., Clarke, M. F., and Newman, A. M. (2020). Single-cell transcriptional diversity is a hallmark of developmental potential. Science, 367(6476):405–411.

Hashimoto, T. B., Gifford, D. K., and Jaakkola, T. S. (2016). Learning population-level diffusions with generative recurrent networks. In Proceedings of the 33rd International Conference on Machine Learning, pages 2417–2426.

Hetzel, L., Boehm, S., Kilbertus, N., Günnemann, S., Lotfollahi, M., and Theis, F. (2022). Predicting cellular responses to novel drug perturbations at a single-cell resolution. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, Advances in Neural Information Processing Systems, volume 35, pages 26711–26722. Curran Associates, Inc.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. Computing in science & engineering, 9(3):90.

Isobe, N., Koyama, M., Hayashi, K., and Fukumizu, K. (2024). Extended flow matching: a method of conditional generation with generalized continuity equation. arXiv preprint arXiv:2402.18839.

Ji, Y., Lotfollahi, M., Wolf, F. A., and Theis, F. J. (2021). Machine learning for perturbational single-cell omics. Cell Systems, 12(6):522–537.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. (2023). Flow matching for generative modeling. In The Eleventh International Conference on Learning Representations.

Liu, X., Gong, C., and Liu, Q. (2022). Flow straight and fast: Learning to generate and transfer data with rectified flow. arXiv preprint arXiv:2209.03003.

Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., and Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. Nature Methods, 15(12):1053–1058.

Lotfollahi, M., Wolf, F. A., and Theis, F. J. (2019). scgen predicts single-cell perturbation responses. Nature Methods, 16(8):715–721.

Makkuva, A. V., Taghvaei, A., Oh, S., and Lee, J. D. (2020). Optimal transport mapping via input convex neural networks. In ICML.

McKinney, W. (2012). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. "O'Reilly Media, Inc.".

Molè, M. A., Coorens, T. H. H., Shahbazi, M. N., Weberling, A., Weatherbee, B. A. T., Gantner, C. W., Sancho-Serra, C., Richardson, L., Drinkwater, A., Syed, N., Engley, S., Snell, P., Christie, L., Elder, K., Campbell, A., Fishel, S., Behjati, S., Vento-Tormo, R., and Zernicka-Goetz, M. (2021). A single cell characterisation of human embryogenesis identifies pluripotency transitions and putative anterior hypoblast centre. Nature Communications, 12(1).

Neklyudov, K., Brekelmans, R., Tong, A., Atanackovic, L., Liu, Q., and Makhzani, A. (2023). A computational framework for solving wasserstein lagrangian flows. arXiv preprint arXiv:2310.10649.

Neklyudov, K., Severo, D., and Makhzani, A. (2022). Action matching: A variational method for learning stochastic dynamics from samples.

Oliphant, T. E. (2006). A guide to NumPy, volume 1. Trelgol Publishing USA.

Oliphant, T. E. (2007). Python for scientific computing. Computing in Science & Engineering, 9(3):10–20.

Otto, F. (2001). The geometry of dissipative evolution equations: the porous medium equation.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, pages 8026–8037.

Peidli, S., Green, T. D., Shen, C., Gross, T., Min, J., Garda, S., Yuan, B., Schumacher, L. J., Taylor-King, J. P., Marks, D. S., et al. (2024). scperturb: harmonized single-cell perturbation data. Nature Methods, pages 1–10.

Peyré, G. and Cuturi, M. (2019). Computational Optimal Transport. arXiv:1803.00567.

Pooladian, A.-A., Ben-Hamu, H., Domingo-Enrich, C., Amos, B., Lipman, Y., and Chen, R. T. (2023). Multisample flow matching: Straightening flows with minibatch couplings. arXiv preprint arXiv:2304.14772.

Price, I., Sanchez-Gonzalez, A., Alet, F., Ewalds, T., El-Kadi, A., Stott, J., Mohamed, S., Battaglia, P., Lam, R., and Willson, M. (2023). Gencast: Diffusion-based ensemble forecasting for medium-range weather. arXiv preprint arXiv:2312.15796.

Ramos Zapatero, M., Tong, A., Opzoomer, J. W., O'Sullivan, R., Cardoso Rodriguez, F., Su, J., Vlckova, P., Nattress, C., Qin, X., Claus, J., Hochhauser, D., Krishnaswamy, S., and Tape, C. J. (2023). Trellis

tree-based analysis reveals stromal regulation of patient-derived organoid drug responses. Cell, 186(25):5606–5619.e24.

Rizvi, A. H., Camara, P. G., Kandror, E. K., Roberts, T. J., Schieren, I., Maniatis, T., and Rabadan, R. (2017). Single-cell topological rna-seq analysis reveals insights into cellular differentiation and development. Nature Biotechnology, 35(6):551–560.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10684–10695.

Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., and Norouzi, M. (2022a). Palette: Image-to-image diffusion models. In ACM SIGGRAPH 2022 conference proceedings, pages 1–10.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. (2022b). Photorealistic text-to-image diffusion models with deep language understanding. Advances in neural information processing systems, 35:36479–36494.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. (2020). Learning to simulate complex physics with graph networks. In International conference on machine learning, pages 8459–8468. PMLR.

Schiebinger, G., Shu, J., Tabaka, M., Cleary, B., Subramanian, V., Solomon, A., Gould, J., Liu, S., Lin, S., Berube, P., et al. (2019). Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. Cell, 176(4):928–943.

Tong, A., FATRAS, K., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Wolf, G., and Bengio, Y. (2024). Improving and generalizing flow-based generative models with minibatch optimal transport. Transactions on Machine Learning Research. Expert Certification.

Tong, A., Huang, J., Wolf, G., Van Dijk, D., and Krishnaswamy, S. (2020). Trajectorynet: A dynamic optimal transport network for modeling cellular dynamics. In International conference on machine learning, pages 9526–9536. PMLR.

Tong, A., Malkin, N., Fatras, K., Atanackovic, L., Zhang, Y., Huguet, G., Wolf, G., and Bengio, Y. (2023). Simulation-free schrödinger bridges via score and flow matching. arXiv preprint arXiv:2307.03672.

Uscidda, T. and Cuturi, M. (2023). The monge gap: A regularizer to learn all transport maps. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pages 34709–34733. PMLR.

Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. Computing in Science & Engineering, 13(2):22.

Van Rossum, G. and Drake Jr, F. L. (1995). Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam.

Verma, Y., Heinonen, M., and Garg, V. (2024). Climode: Climate and weather forecasting with physics-informed neural odes. arXiv preprint arXiv:2404.10024.

Villani, C. (2009). Optimal transport: old and new, volume 338. Springer.

Weinreb, C., Wolock, S., Tusi, B. K., Socolovsky, M., and Klein, A. M. (2018). Fundamental limits on dynamic inference from single-cell snapshots. 115(10):E2467–E2476.

Yadan, O. (2019). Hydra - a framework for elegantly configuring complex applications. Github.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. Advances in neural information processing systems, 30.

Zeng, T. and Dai, H. (2019). Single-cell rna sequencing-based computational analysis to describe disease heterogeneity. Frontiers in Genetics, 10.

Zheng, Q., Le, M., Shaul, N., Lipman, Y., Grover, A., and Chen, R. T. (2023). Guided flows for generative modeling and decision making. arXiv preprint arXiv:2311.13443.

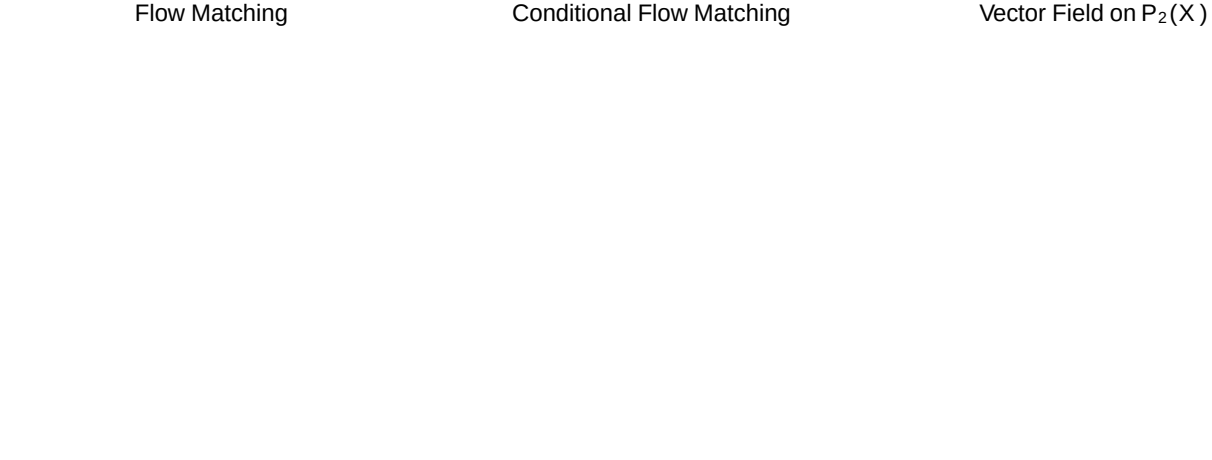

Figure 2: Illustration of ow matching methods on the 2-Wasserstein manifold $P_2(X)$, depicted as a two-dimensional sphere. Flow Matching learns the tangent vectors to a single curve on the manifold. Conditional generation corresponds to learning a nite set of curves on the manifold, e.g. classes c_1 and c_2 on the plot. Meta Flow Matching learns to integrate a vector eld on $P_2(X)$, i.e. for every starting density p_0 Meta Flow Matching de nes a push-forward measure that integrates along the underlying vector eld.

A. Figures

Here we include schematics comparing Flow Matching, Conditional Flow Matching and Meta Flow Matching, see Figure 2.

B. Background

B.1. Generative Modeling via Flow Matching

Flow Matching is an approach to generative modeling recently proposed independently in different works: Recti ed Flows (Liu et al., 2022), Flow Matching (Lipman et al., 2023), Stochastic Interpolants (Albergo and Vanden-Eijnden, 2022). It assumes a continuous interpolation between densities $p_0(x_0)$ and $p_1(x_1)$ in the sample space. That is, the sample from the intermediate density $p_t(x_t)$ is produced as follows

$$x_t = f_t(x_0; x_1); \quad (x_0; x_1) \qquad (x_0; x_1) ; \tag{9}$$

$$\text{where} \quad \int dx_1 \quad (x_0; x_1) = p_0(x_0) ; \qquad \int dx_0 \quad (x_0; x_1) = p_1(x_1) ; \tag{10}$$

where f_t is the time-continuous interpolating function such that $f_{t=0}(x_0; x_1) = x_0$ and $f_{t=1}(x_0; x_1) = x_1$ (e.g. linearly between x_0 and x_1 with $f_t(x_0; x_1) = (1 \quad t) \quad x_0 + t \quad x_1$); $(x_0; x_1)$ is the density of the joint distribution, which is usually

Figure 3: Organoid drug-screen dataset overview. Left: a given replica consists of a control distribution p_0 and corresponding treatment response distribution p_1 for treatment condition c_i. Right: train and test data splits for replica (top) and patients (bottom) splits, restively. For each experiment there are 11 treatments, 10 patients and 3 culture conditions.

taken as a distribution of independent random variables $(x_0; x_1) = p_0(x_0)p_1(x_1)$, but can also be generalized to formulate the optimal transport problems (Pooladian et al., 2023; Tong et al., 2024). The corresponding density can be de ned then as the following expectation

$$p_t(x) = \int dx_0 dx_1 \quad (x_0; x_1) \ (x \quad f_t(x_0; x_1)) : \tag{11}$$

The essential part of Flow Matching is the continuity equation that describes the change of this density through the vector eld on the state space, which admits vector eld $v_t(x)$ as a solution

$$\frac{@p_t(x)}{@t} = \ hr \ _x ; p_t(x)v_t(x)i ; \quad v_t(\ ) = \frac{1}{p_t(\ )} E_{(x_0;x_1)} \quad (f_t(x_0; x_1) \quad) \frac{@f_t(x_0; x_1)}{@t} \quad :$$

Relying on this formula, one can derive the tractable objective for learning $v_t(x)$, i.e.

$$L_{FM}(!) = \int_0^1 dt \ E_{p_t(x)} kv_t(x) \quad v_t(x; !)k^2 \tag{12}$$

$$= E_{(x_0;x_1)} \int_0^1 dt \ \frac{@}{@t} f_t(x_0; x_1) \quad v_t(f_t(x_0; x_1); !) \ ^2 + \text{constant} \tag{13}$$

Finally, the vector eld $v_t(\ ; !) \quad v_t(\ )$ de nes the push-forward density that approximately matches $p_{t=1}$, i.e. $T_{\#} \ p_0 \ p_{t=1}$, where T is the ow corresponding to vector eld $v_t(\ ; !)$ with parameters .

B.2. Conditional Generative Modeling via Flow Matching

Conditional image generation is one of the most common applications of generative models nowadays; it includes conditioning on the text prompts (Saharia et al., 2022b; Rombach et al., 2022) as well as conditioning on other images (Saharia et al., 2022a). To learn the conditional generative process with diffusion models, one merely has to pass the conditional variable (sampled jointly with the data point) as an additional input to the parametric model of the vector eld. The same applies for the Flow Matching framework.

Conditional Generative Modeling via Flow Matching is independently introduced in several works (Zheng et al., 2023; Dao et al., 2023; Isobe et al., 2024) and it operates as follows. Consider a family of time-continuous distributions $p_t(x_t \mid c)$, which corresponds to the distribution of the following random variable

$$x_t = f_t(x_0; x_1); \quad (x_0; x_1) \quad (x_0; x_1 \mid c) : \tag{14}$$

For every c, the density $p_t(x_t \mid c)$ follows the continuity equation with the following vector eld

$$v_t(\ \mid c) = \frac{1}{p_t(\ \mid c)} E_{(x_0;x_1)} \quad (f_t(x_0; x_1) \quad) \frac{@f_t(x_0; x_1)}{@t} ; \tag{15}$$

which depends on c. Thus, the training objective of the conditional model becomes

$$L_{CGFM}(!) = E_{p(c)} E_{(x_0;x_1 \mid c)} \int_0^1 dt \ \frac{@}{@t} f_t(x_0; x_1) \quad v_t(f_t(x_0; x_1) \mid c; !) \ ^2 ; \tag{16}$$

where, compared to the original Flow Matching formulation, we rst have to sample c, then produce the samples from $p_t(x_t \mid c)$ and pass c as input to the parametric model of the vector eld.

C. Related Work

Meta-learning of probability measures. The meta-learning of probability measures was previously studied by Amos et al. (2022); Bunne et al. (2022a;b) where they demonstrate that the prediction of the optimal transport paths can be ef ciently amortized over the input marginal measures. The main difference with our approach is that we are trying to learn the push-forward map without embedding the second marginal.

Generative modeling for single cells. Single cell data has expanded to encompass multiple modalities of data pro ling cell state and activities (Frangieh et al., 2021; Bunne et al., 2023b). Single-cell data presents multiple challenges in terms of noise, non-time resolved, and high dimension, and generative models have been used to counter those problems. Autoencoder has been used to embed and extrapolate data Out Of Distribution (OOD) with its latent state dimension (Lotfollahi et al., 2019; Lopez et al., 2018; Hetzel et al., 2022). Orthogonal non-negative matrix factorization (oNMF) has also been used for dimensionality reduction combined with mixture models for cell state prediction (Chen et al., 2020). Other approaches have tried to use Flow Matching (FM) (Tong et al., 2023; 2024; Neklyudov et al., 2023) or similar approaches such as the Monge gap (Uscidda and Cuturi, 2023) to predict cell trajectories. Currently, the state of the art method uses the principle of Optimal Transport (OT) to predict cell trajectories with Input Convex Neural Network (ICNN) (Makkuva et al., 2020; Bunne et al., 2023b). What determines the signi cance of the method is its capability in generalizing out of distribution to a new population of cells, which may be from different culture or individuals. As of this time, our method is the only method that takes inter-cellular interactions into account.

Generative modeling for physical processes. The closest approach to ours is the prediction of the many-body interactions in physics (Sanchez-Gonzalez et al., 2020) via GNNs. However, the problem there is very different since these models use the information about the individual trajectories of samples, which are not available for the single-cell prediction. Neklyudov et al. (2022) consider learning the vector eld for any continuous time-evolution of a probability measure, however, their method is restricted to single curves and do not consider generalization to unseen data. Finally, the weather/climate forecast models generating the next state conditioned on the previous one (Price et al., 2023; Verma et al., 2024) are similar approaches to ours but operating on a much ner time resolution.

D. Data processing

Synthetic Data. We curate a synthetic dataset of the joint distribution $\{(p_0(x_0 \mid i), p_1(x_1 \mid i))\}_{i=1}^{N}$ by simulating a diffusion process applied to a set of pre-de ned target distributions $p_1(x_1 \mid i)$ for $i = 1, \ldots, N$. To get a paired population $p_0(x_0 \mid i)$ we simulate the forward diffusion process without drift $x_0 + \mathcal{N}(x_1, \sigma)$. After this setup, for reasonable values of σ, we assume that one can reverse the diffusion process and learn the push-forward map $p_\theta(x_0 \mid i)$ to $p_1(x_1 \mid i)$ for every index i. For this task, given the i-th population index we denote $p_0(x_0 \mid i)$ as the *source population* $p_1(x_1 \mid i)$ as the i-th *target population*.

To construct $p_1(x_1 \mid i)$, we discretize samples from a de ned silhouette; e.g. an image of a character, where i indexes the respective character. We use upper case letters as the silhouette and generate the corresponding samples $x_1 \sim p_1(x_1 \mid i)$ from the uniform distribution over the silhouette and run the diffusion process for samples to acquire x_0. We construct the *training data* using 10 random orientations of 24 letters, while only using the upright orientation for the remaining letters "X" and "Y". We construct the *test data* by using 10 random orientations of "X" and "Y" (validation and test, respectively) that differ from the upright orientations of the same letters in the training data. We do this to simplify the generalization task – the model will see the shapes of "X" and "Y" during training, but the same letters under different orientations remain unseen.

Biological Data. For experiments on biological data, we use the organoid drug-screen dataset from (Ramos Zapatero et al., 2023). This dataset is a single-cell mass-cytometry dataset collected over 10 patients. Somewhat unique to this dataset, unlike many prior perturbation-screen datasets which have a single control population, this dataset has matched controls to each experimental condition. Populations from each patient are treated with 11 different drug treatments of varying dose concentrations.[1] We use the term *replicate* to de ne control-treatment population pairs $p_0(x_0 \mid c_i)$ and $p_1(x_1 \mid c_i)$, respectively (see Fig. 3-left). In each patient, cell population are categorized into 3 cell types (i) cancer associated Fibroblasts, (ii) patient-derived organoid cancer cells (PDO), and (iii) patient-derived organoid cancer cells co-cultured broblasts (PDOF). We report results averaged over Fibroblast/PDO/PDOF cultures and results for the individual cultures (this is reported in Appendix F.3).

Pre-processing and data splits. We lter each cell population to contain at least 1000 cells and consider 43 bio-markers. We consider two data splits for the organoid drug-screen dataset (see Fig. 3-right): (1) *Replicate split*, here we leave-out replicates evenly across all patients for testing. (2) *Patients split*, here we leave-out replicates fully in one patients – in this setting, we are testing the ability of of model to generalize population prediction of treatment response for unseen patients. In both settings, we normalize the data and embed it into a lower dimensional principle components (PC) representation. We

[1] We consider only the highest dosage and leave exploration of dose-dependent response to future work.

do this to reduce the dimensionality of the data and to extract the relevant information from the 43 bio-markers (features) of the ambient space. We train and evaluate all models in the PC space. For all organoid drug-screen dataset experiments we use PC=10. Further details regarding data pre-processing and data splits are provided in Appendix F.2.

For the organoid drug-screen experiments, we consider an ICNN architecture in addition to the Flow-matching models. The ICNN model is based on CellOT (Bunne et al., 2023a); a method for learning cell speci c response to treatments. The ICNN (and likewise CellOT) counterparts our FM model in that it does not take the population index as a condition. Therefore, it will neither be able to t the training data, nor generalize.

E. Proof of Theorem 1

Theorem 1. Consider a dataset of populations $\mathcal{D} = \{(x_0, x_1 \mid i)\}_i$ generated from some unknown conditional model $(x_0, x_1 \mid c)p(c)$. Then the following objective

$$L(\theta; \eta) = \mathbb{E}_{t, p(c), p_t(x_t \mid c)} \left\| v_t(x_t \mid c) - v_t(x_t \mid \phi(p_0; \theta); \eta) \right\|^2 \tag{5}$$

is equivalent to the Meta Flow Matching objective $L_{\text{MFM}}(\theta; \eta)$

$$\mathbb{E}_{t, i, (x_0, x_1 \mid i)} \left\| \frac{\partial}{\partial t} f_t(x_0, x_1) - v_t(f_t(x_0, x_1) \mid \phi(p_0; \theta); \eta) \right\|^2 \tag{6}$$

up to an additive constant.

Proof. The loss function

$$L(\theta; \eta) = \mathbb{E}_{p(c)} \int_0^1 dt\, \mathbb{E}_{p_t(x_t \mid c)} \| v_t(x_t \mid c) - v_t(x_t \mid \phi(p_t; \theta); \eta) \|^2 \tag{17}$$

$$= 2\mathbb{E}_{p(c)} \int dt dx \, \langle p_t(x \mid c) v_t(x \mid c), v_t(x \mid \phi(p_t; \theta); \eta) \rangle + \tag{18}$$

$$+ \mathbb{E}_{p(c)} \int_0^1 dt\, \mathbb{E}_{p_t(x_t \mid c)} \| v_t(x_t \mid \phi(p_t; \theta); \eta) \|^2 + \tag{19}$$

$$+ \mathbb{E}_{p(c)} \int_0^1 dt\, \mathbb{E}_{p_t(x_t \mid c)} \| v_t(x_t \mid c) \|^2 : \tag{20}$$

The last term does not depend on θ, the second term we can estimate, for the first term, we use the formula for the (from Eq. (15))

$$p_t(x \mid c) v_t(x \mid c) = \mathbb{E}_{(x_0, x_1)} \left(f_t(x_0, x_1) - x \right) \frac{\partial f(x_0, x_1)}{\partial t} : \tag{21}$$

Thus, the loss is equivalent (up to a constant) to

$$L(\theta; \eta) = 2\mathbb{E}_{p(c)} \mathbb{E}_{(x_0, x_1 \mid c)} \int dt \left\langle \frac{\partial f(x_0, x_1)}{\partial t}, v_t(f_t(x_0, x_1) \mid \phi(p_t; \theta); \eta) \right\rangle \tag{22}$$

$$+ \mathbb{E}_{p(c)} \mathbb{E}_{(x_0, x_1 \mid c)} \int_0^1 dt \, \| v_t(f_t(x_0, x_1) \mid \phi(p_t; \theta); \eta) \|^2 \tag{23}$$

$$\mathbb{E}_{p(c)} \mathbb{E}_{(x_0, x_1 \mid c)} \int_0^1 dt \left\| \frac{\partial f(x_0, x_1)}{\partial t} \right\|^2 \tag{24}$$

$$= \mathbb{E}_{c \sim p(c)} \mathbb{E}_{(x_0, x_1 \mid c)} \int_0^1 dt \left\| \frac{\partial}{\partial t} f_t(x_0, x_1) - v_t(f_t(x_0, x_1) \mid \phi(p_t; \theta); \eta) \right\|^2 : \tag{25}$$

Note that in the final expression we do not need access to the probabilistic model $p(c)$ if the joints $(x_0, x_1 \mid c)$ are already sampled in the data $\mathcal{D}$. Thus, we have

$$L(\theta; \eta) = \mathbb{E}_{c \sim p(c)} \mathbb{E}_{(x_0, x_1 \mid c)} \int_0^1 dt \left\| \frac{\partial}{\partial t} f_t(x_0, x_1) - v_t(f_t(x_0, x_1) \mid \phi(p_t; \theta); \eta) \right\|^2 \tag{26}$$

$$= \mathbb{E}_{i \sim \mathcal{D}} \mathbb{E}_{(x_0, x_1 \mid i)} \int_0^1 dt \left\| \frac{\partial}{\partial t} f_t(x_0, x_1) - v_t(f_t(x_0, x_1) \mid \phi(p_t; \theta); \eta) \right\|^2 \tag{27}$$

$$= L_{\text{MFM}}(\theta; \eta) : \tag{28}$$

Table 2: Results of the synthetic letters experiment for population prediction on seen train populations and unseen test populations. We report the the 1-Wasserstein (W_1), 2-Wasserstein (W_2), and the maximum-mean-discrepancy (MMD) distributional distances. We consider 4 settings for MFM with varying k.

	Train						Test					
	W_1		W_2		MMD ($\times 10^{-3}$)		W_1		W_2		MMD ($\times 10^{-3}$)	
FM	0.216	0.000	0.280	0.000	2.38	0.00	0.237	0.000	0.315	0.000	3.28	0.00
CGFM	0.093	0.000	0.112	0.000	0.34	0.00	0.317	0.000	0.397	0.000	6.67	0.00
MFM (k = 0)	0.099	0.000	0.128	0.000	0.25	0.00	0.221	0.000	0.267	0.000	3.77	0.00
MFM (k = 1)	0.096	0.003	0.124	0.004	0.22	0.04	0.217	0.003	0.261	0.003	3.80	0.28
MFM (k = 10)	0.096	0.003	0.124	0.003	0.23	0.04	0.213	0.008	0.256	0.008	3.68	0.45
MFM (k = 50)	0.099	0.003	0.127	0.003	0.25	0.05	0.226	0.005	0.270	0.007	4.38	0.30

□

F. Experimental Details

F.1. Synthetic letters data

The synthetic letters dataset contains 242 train populations a 10 test populations. Each population contains roughly between 750 and 2700 samples.

F.2. Organoid drug-screen data

The organoid drug-screen dataset contains a total of 927 replicates (or coupled populations) pr dates split, we use 713 populations for training and 103 left-out populations for testing. In patients split, we use 861 populations for training and 33 left-out populations for testing.

F.3. Extended Results

Here we include extended results from letter generation, in gure 4. As well, we included the results of all three different type of cell culture (Fibroblast alone, PDO alone, and PDO co-cultured with broblast) 3.

F.4. Model architectures and hyperparameters

ICNN. The ICNN baseline was constructed with two networks ICNN network $f(x)$ and $g(x)$, with non-negative leaky ReLU activation layers. $f(x)$ is used to minimize the transport distance and $g(x)$ is used to transport from source to target. It has four hidden units with width of 64, and a latent dimension of 50. Both networks uses Adam optimizer ($lr = 1e-4$, $\beta_1=0.5$, $\beta_2=0.9$). $g(x)$ is trained with an inner iteration of 10 for every iteration $f(x)$ is trained.

Vector Field Models. All vector eld models v_t are parameterized 4 linear layers with 512 hidden units and SELU activation functions. The FM vector eld model additionally takes a conditional input for the one-hot treatment encoding. CGFM takes the conditional input for the one-hot treatment conditions as well as a one-hot encoding for the population index condition. The MFM vector eld model takes population embedding conditions, that is output from the GCN, as input, as well as the treatment one-hot encoding. All vector eld models use temporal embeddings for time and positional embeddings for the input samples. We did not sweep the size of this embeddings space and found that a temporal embedding and positional embeddings sizes of 128 worked suf ciently well.

Graph Neural Network. We considered a GCN model that consists of a k-nearest neighbour graph edge pooling layer and 3 graph convolution layers with 512 hidden units. The nal GCN model layer outputs an embedding representation $z \in \mathbb{R}^d$. For the Synthetic experiment, we found that $d = 256$ performed well, and $d = 128$ performed well for the biological experiments. We normalize and project embeddings onto a hyper-sphere, and nd that this normalization helps improve training. Additionally, the GCN takes a one-hot cell-type encoding (encoding for Fibroblast cells or PDO cells) for the control populations p_0. This may be bene cial for PDOF populations where both Fibroblast cells and PDO cells are present. However, it is important to note that labeling which cells are Fibroblasts versus PDOs withing the PDOF cultures is dif cult

and noisy in itself, hence such a cell-type condition may yield no additive information/performance gain.

Optimization. We use the Adam optimizer with a learning rate of 0.0001 for all Flow-matching models (FM, CGFM, MFM). We also used the Adam optimizer with a learning rate of 0.0001 for the GCN model. To train the MFM (FM+GCN) models, we alternate between updating the vector field model parameters and the GCN model parameters. We alternate between updating the respective model parameters every epoch. FM and CGFM model were trained for 2000 epochs, while MFM models were trained for 4000 epochs. Due to the alternating optimization, the MFM vector field model receives half as many updates compared to its counterparts (FM and CGFM). Therefore, training for the double the epochs is necessary for fair comparison.

The hyperparameters stated in this section were selected from brief and small grid search sweeps. We did not conduct any thorough hyperparameter optimization.

G. Implementation Details

We implement all our experiments using PyTorch and PyTorch Geometric (Van Rossum and Drake Jr, 1995; Oliphant, 2007; Paszke et al., 2019; Fey and Lenssen, 2019), with support from Hydra (Yadan, 2019), Matplotlib (Hunter, 2007), numpy (Oliphant, 2006; Van Der Walt et al., 2011), and pandas (McKinney, 2012).

All experiments were conducted on a HPC cluster primarily on NVIDIA Tesla T4 16GB GPUs. Each individual seed experiment run required only 1 GPU. Each experiment ran between 3-11 hours and all experiments took approximately 500 GPU hours.

Table 3: Experimental results on the organoid drug-screen dataset for population prediction of treatment response at out-of-sample populations. Results are reported for models trained on data embedded into 10 principle components. We report the the 1-Wasserstein (W_1), 2-Wasserstein (W_2), and the maximum-mean-discrepancy (MMD) distributional distances. We consider 2 settings for MFM with varying nearest-neighbours parameter.

	Fibroblasts											
	Train						Test					
	W_1		W_2		MMD ($\times 10^{-3}$)		W_1		W_2		MMD ($\times 10^{-3}$)	
FM	1.599	0.071	1.761	0.137	282	0.34	1.667	0.003	1.846	0.064	7.85	0.15
ICNN	1.695	0.08	1.796	0.09	482	3.412	1.6	0.009	1.68	0.013	622	1.32
CGFM	1.496	0.019	1.572	0.016	1.45	0.14	1.566	0.028	1.652	0.026	6.46	0.82
MFM (k = 0)	1.551	0.037	1.632	0.042	2.31	0.71	1.453	0.200	1.527	0.022	3.66	0.67
MFM (k = 10)	1.555	0.034	1.635	0.039	2.54	0.42	1.441	0.003	1.514	0.001	3.37	0.72
	PDO											
	Train						Test					
	W_1		W_2		MMD ($\times 10^{-3}$)		W_1		W_2		MMD ($\times 10^{-3}$)	
FM	1.996	0.196	2.171	0.243	6.79	3.40	2.128	0.064	2.312	0.075	7.88	1.26
ICNN	2.315	0.060	2.478	0.057	2368	0.006	2.538	0.018	2.731	0.027	2328	20.6
CGFM	1.662	0.026	1.760	0.023	1.74	0.16	2.460	0.018	2.533	0.023	13.6	0.25
MFM (k = 0)	1.837	0.058	1.964	0.059	3.74	0.29	2.010	0.142	2.168	0.182	6.01	1.77
MFM (k = 10)	1.838	0.035	1.957	0.038	3.75	0.41	1.971	0.082	2.114	0.101	5.42	1.11
	PDOF											
	Train						Test					
	W_1		W_2		MMD ($\times 10^{-3}$)		W_1		W_2		MMD ($\times 10^{-3}$)	
FM	2.390	0.148	2.806	0.198	11.0	2.21	4.026	0.018	4.683	0.011	49.0	1.66
ICNN	2.479	0.06	2.826	0.063	291	9.24	3.968	0.0554	4.579	0.060	1263	37.5
CGFM	2.160	0.170	2.530	0.237	7.90	1.79	4.000	0.010	4.629	0.012	49.2	0.76
MFM (k = 0)	2.202	0.072	2.548	0.089	8.98	0.59	3.717	0.138	4.360	0.162	40.3	3.52
MFM (k = 10)	2.251	0.143	2.631	0.197	9.45	1.52	3.565	0.132	4.201	0.119	36.1	4.97

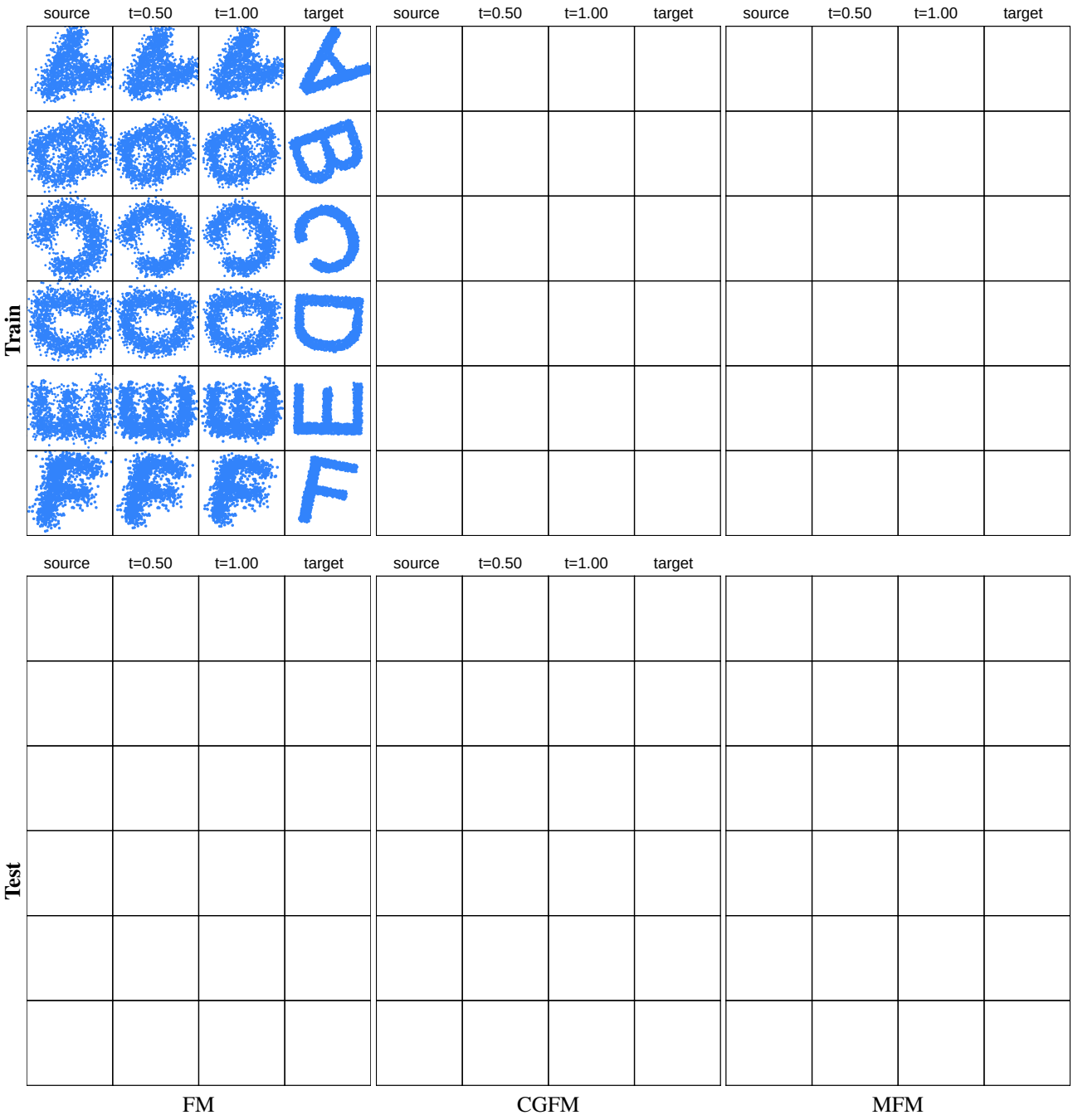

Figure 4: Model-generated samples for synthetic letters from the source ($t = 0$) to target ($t = 1$) distributions.