Measure-to-measure Regression with Transformers

Matthew Vandergrift [1 2] **Martha White** [1 2 3] **Yury Polyanskiy** [4] **Philippe Rigollet** [4] **Lazar Atanackovic** [1 2 5]

Abstract

Many learning problems require predicting how populations evolve under an unknown transformation. A natural representation for such populations is a probability measure, with point clouds as a key example. In this work, we study the ***measure-to-measure* (M2M) regression** problem, in which one seeks to learn a map between probability measures from a finite collection of observed input–output pairs. In contrast to classical regression, where individual samples are transformed independently, M2M regression treats entire distributions as the data points. This perspective is vital in certain scientific applications, for example, cellular and molecular biology, where cells are known to evolve not as independent data points but as a collection. However, few existing approaches address the problem of M2M regression with sufficient expressivity and scalability. We present a formalization of nonlinear M2M regression and introduce two easy-to-use, expressive, and scalable approaches to learn such operators: transformers as ***static*** M2M maps and transformers as ***dynamic*** M2M velocity fields. Our approach leverages the natural measure-dependent and mean-field structure of transformers to learn nonlinear M2M maps on the space of probability distributions. We illustrate the effectiveness of our proposed method to generalize to unseen measures on synthetic experiments, interacting particle systems, and a large-scale patient-derived organoid dataset for predicting treatment response in colorectal cancer.

1. Introduction

Many modern datasets can be described as collections of distributions rather than individual vectors: bags of images, collections of time series, and sets of point clouds. As a result, a natural question arises: *can we learn to model unknown transformations on the space of distributions?* This is of particular interest in the natural sciences, where particles of physical systems are known to act as a collective and not independently. A motivating example arises from the advent of high-throughput single-cell perturbation screens (Dixit et al., 2016; Zapatero et al., 2023; Peidli et al., 2024; Zhang et al., 2025; Huang et al., 2025), which allows individuals' responses to stimuli to be measured by point clouds of cells in *control* and *treated* conditions. Here, an unknown function maps *pre-treatment* populations of cells to corresponding *post-treatment* populations, and the modeling objective is to recover the ground-truth transformation (Bunne et al., 2023; Atanackovic et al., 2025; Adduri et al., 2025; Wei et al., 2025). Since cells communicate and evolve as a collective rather than independently (Su et al., 2024; Armingol et al., 2021; Goodenough & Paul, 2009), faithfully capturing their complex dynamics is a core modeling challenge, and one that calls for computational frameworks that explicitly account for such distribution-dependent behavior.

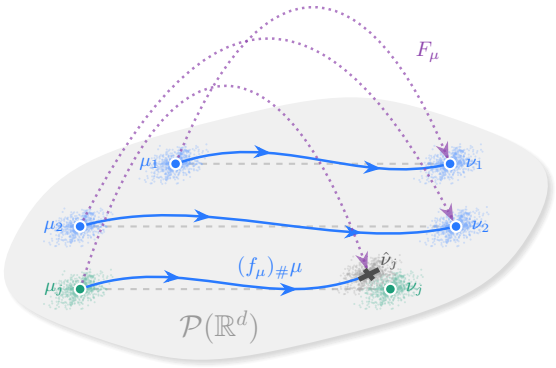


Figure 1. **Illustration of the *measure-to-measure* (M2M) regression problem.** In M2M regression, we treat entire distributions/measures as paired data points (e.g., $(\mu_i, \nu_i) \in \mathcal{P}(\mathbb{R}^d) \times \mathcal{P}(\mathbb{R}^d)$) to learn a continuous mapping between them. Leveraging the natural measure-dependent structure of transformers, we introduce two classes of methods for learning nonlinear M2M operators: *static* one-step pushforward maps $F : \mu \mapsto (F_\mu)_\#\mu$ and and *dynamic* flow maps $f : \mu \mapsto (f_\mu)_\#\mu$ obtained by integrating an ODE on $\mathbb{R}^d$. These learned operators can robustly generalize to *unseen* test measures (predict $\hat{\nu}_j$ given novel input μ_j).

Measure-to-measure (M2M) regression. Modeling such transformations naturally corresponds to learning functions between probability measures on $\mathcal{P}(\mathbb{R}^d)$ (see Figure 1).

[1]University of Alberta [2]Alberta Machine Intelligence Institute (Amii) [3]Canada CIFAR AI Chair [4]Massachusetts Institute of Technology [5]The Broad Institute of MIT and Harvard. Correspondence to: Lazar Atanackoivc <atanacko@ualberta.ca>.

Accepted at the 2026 Workshop on Generative and Agentic AI for Biology (ICML 2026)

Early work along these lines on *distribution-to-distribution regression* relied on kernel methods (Oliva et al., 2013), while subsequent work on distribution regression has largely focused on settings in which either the input or the output is a vector (Szabó et al., 2016; Muandet et al., 2017; Petersen & Müller, 2019), and thus does not apply directly to the problem we study here. A complementary line of work extends this setting to *measure-to-measure* (M2M) *regression* by modeling the regression operator as a transport map on the sample space (Ghodrati & Panaretos, 2022; 2023; Bunne et al., 2023; Girshfeld & Chen, 2025; Geuter et al., 2025b;a). Most of these approaches, however, rely on limited function classes and do not scale effectively to large problems.

Transformers as operators on measures. Transformers provide a substantially more expressive route to modeling nonlinear transformations on the space of probability measures, and have been used successfully to learn functions over sets of point clouds (Lee et al., 2019; Yang et al., 2019; Zhao et al., 2021; Haviv et al., 2025). A growing body of work analyzes their mean-field limits, emergent dynamics, and approximation properties as operators on spaces of probability measures (Sander et al., 2022; Amos et al., 2022; Geshkovski et al., 2024a; 2025; Chen et al., 2025a; Castin et al., 2025); in particular, under mild conditions, deep transformers are universal approximators of M2M maps (Yun et al., 2020; Alberti et al., 2023; Kratsios et al., 2022; Jiang & Li, 2024; Geshkovski et al., 2024b).

This literature has remained largely theoretical, and the practical use of transformers as operators between probability measures is still limited. Two recent works explore this direction in specific scenarios: approximating Gaussian mixture models (Zimin et al., 2025) and learning transformations on cell sets (Adduri et al., 2025). Both can be viewed as special cases of our *static* M2M framework, in which a transformer is used as a single-step transport map between measures.

Transformers as dynamic continuous-time transport maps. In contrast to such *static* M2M maps, a recent line of work models these transformations as *dynamic*, continuous-time transport maps whose velocity field is itself distribution-dependent (Atanackovic et al., 2025; Haviv et al., 2025). The natural geometric object underlying this viewpoint is the *flow map* on the ground space: given a (possibly measure-dependent) time-varying velocity field v_t on $\mathbb{R}^d$, the associated flow map $\Phi_t : \mathbb{R}^d \to \mathbb{R}^d$ sends an initial state x_0 to the solution at time t of the ODE $\dot{x}_t = v_t(x_t)$. Flow maps are appealing because they lift dynamics on the ground space to dynamics on the space of measures via push-forward, $\mu_t = (\Phi_t)_\sharp \mu_0$, thereby reducing the problem of learning an operator on $\mathcal{P}(\mathbb{R}^d)$ to that of learning a vector field on $\mathbb{R}^d$.

Crucially, this computational reduction comes at no expressiveness cost: under mild conditions, every M2M map can be realized as a continuous-time, measure-dependent flow map, making this class maximally expressive within $\mathcal{P}(\mathbb{R}^d) \to \mathcal{P}(\mathbb{R}^d)$ maps (Lavenant & Savaré, 2026). Frameworks such as flow matching (Liu et al., 2022; Lipman et al., 2022; Tong et al., 2023; Albergo et al., 2023) then provide a scalable route to training such models in practice. Self-attention has been routinely used to parametrize flow models for point-wise generation trained with flow matching (Jing et al., 2024; Ma et al., 2024; Hu et al., 2024), but such methods do not address the broader M2M regression problem. We close this gap by introducing a practical and scalable framework for M2M regression through both *static* and *dynamic* transformers.

Contributions. Building on these largely independent lines of research, we formalize the problem of learning nonlinear M2M operators and introduce two distinct transformer-based paradigms, addressing the *static* and *dynamic* settings respectively. Our core contributions are as follows:

- We formalize the nonlinear M2M regression problem through the lens of operators on $\mathcal{P}(\mathbb{R}^d)$, defining the framework via both direct mappings and continuous-time dynamics.
- We introduce two practical and scalable approaches to learn such operators: a *static* method that uses transformers to directly approximate the M2M map, and a *dynamic* method, *Transformer Flow Matching* (`M2M-TFM`), which elegantly connects the measure-dependence of transformers with continuous-time flow matching.
- We demonstrate strong empirical performance across several settings, including a large-scale real-world application: predicting colorectal cancer treatment responses in novel patients.

2. Measure-to-measure regression

In M2M regression, an unknown regression operator maps input measures to output measures. The fundamental statistical objects are probability measures rather than individual observations. The dataset consists of n pairs, where for each index $i = 1, \dots, n$, we observe two finite point clouds on $\mathbb{R}^d$, $\mathcal{X}_i = \{x_{i1}, \dots, x_{iN}\}$, $\mathcal{Y}_i = \{y_{i1}, \dots, y_{iN}\}$ which we represent by their empirical measure

$$\mu_i := \frac{1}{N}\sum_{j=1}^{N} \delta_{x_{ij}} \quad \text{and} \quad \nu_i := \frac{1}{N}\sum_{j=1}^{N} \delta_{y_{ij}}$$

Each observation is thus a pair $(\mu_i, \nu_i) \in \mathcal{P}(\mathbb{R}^d) \times \mathcal{P}(\mathbb{R}^d)$, and the dataset consists of n such pairs.[1]

The measures μ_i and ν_i are treated as the primary statistical objects—elements of $\mathcal{P}(\mathbb{R}^d)$ with finite support—rather

[1] We assume for simplicity that both point clouds, have the same number of elements. Extending our setup to point clouds of different cardinality is straightforward as the points x_{ij} and y_{ij} do not need to be paired.

than as finite-sample approximations of latent population-level distributions. We regard (μ_i, ν_i) as independent copies of a random pair $(\mu, \nu) \in \mathcal{P}(\mathbb{R}^d) \times \mathcal{P}(\mathbb{R}^d)$ satisfying the structural relation $\nu = \mathcal{T}^\star(\mu)$ for some measurable *operator* $\mathcal{T}^\star : \mathcal{P}(\mathbb{R}^d) \to \mathcal{P}(\mathbb{R}^d)$. In our motivating application, μ and ν correspond to pre- and post-intervention distributions, and $\mathcal{T}^\star$ represents the effect of a fixed intervention at the distributional level.

The goal of *measure-to-measure regression* is to estimate $\mathcal{T}^\star$ from the observed samples $\{(\mu_i, \nu_i)\}_{i=1}^n$. Although the observed measures are finitely supported, $\mathcal{T}^\star$ is defined on the full space $\mathcal{P}(\mathbb{R}^d)$. Consequently, M2M regression is an extrapolation problem on the space of measures: a good estimator should generalize beyond the empirical supports seen at training time, including to measures with different support sizes or to smooth limits thereof. The main challenge in this problem is to develop a function class for M2M maps that is simultaneously expressive, computationally scalable, and easy to implement.

3. Transformers for Measure-to-measure Regression

In this section, we outline the connections between transformers and M2M regression. We begin with a brief overview of transformer primitives needed to place these constructions in a common framework (Section 3.1), then we discuss M2M operators (Section 3.2), and lastly we introduce two novel methods which exploit the measure dependence of transformers to solve M2M regression tasks (Section 3.3 and Section 3.4).

3.1. Transformers

A transformer (Vaswani et al., 2017) operates on a finite collection of tokens $x_1, \dots, x_N \in \mathbb{R}^d$ by composing T blocks, each combining self-attention, a pointwise feedforward map (MLP), normalization, and residual connections. For a single attention head, given matrices Q, K, V, the attention output for token x_i is

$$\mathrm{Attn}(x_i; \{x_1, \dots, x_N\}) = V \sum_{j=1}^{N} x_j \frac{\exp(\langle Qx_i, Kx_j \rangle)}{\sum_{l=1}^{N} \exp(\langle Qx_i, Kx_l \rangle)}$$

Because attention does not depend on the order of the tokens $x_1, \dots, x_N$, we can equivalently write $\mathrm{Attn}(x_i; \mu_N)$ that conditions on the empirical measure,[2] $\mu_N := \frac{1}{N} \sum_{j=1}^{N} \delta_{x_j}$. More generally, for any measure μ on $\mathbb{R}^d$ and any $x \in \mathbb{R}^d$ we have,

$$\mathrm{Attn}(x; \mu) := \frac{V \int y \exp(\langle Qx, Ky \rangle)\, d\mu(y)}{\int \exp(\langle Qx, Ky \rangle)\, d\mu(y)}. \qquad (1)$$

[2]This *mean-field* structure is characteristic of *encoder* architectures. The causal masking used in *decoder* architectures for large language models breaks this symmetry and is not considered here.

This view of attention emphasizes why transformers are particularly well-suited to M2M regression. Including the MLP, normalization, residual connections, and stacking blocks yields a discrete-time interacting particle system whose dynamics depend on the empirical distribution of the tokens. Abstractly, a transformer can thus be viewed as a learned, measure-dependent map $F_{\mu_N}(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ parameterized by the empirical measure μ_N of the input tokens. This perspective has proved fruitful for the mathematical analysis of transformers, starting with (Sander et al., 2022) and further developed in (Geshkovski et al., 2024a;b; 2025; Castin et al., 2025; 2024; Chen et al., 2026; 2025a;b). In practice, F_{μ_N} takes as input a set of samples (tokens) x which define the empirical measure μ_N and induces measure-dependence by acting on all tokens as a collective via the self-attention mechanism.

3.2. Measure-to-measure operators

M2M operators admit a variety of structural forms. A classical choice is a *Markov (linear) operator* $\mathcal{T} : \mathcal{P}(\mathbb{R}^d) \to \mathcal{P}(\mathbb{R}^d)$ of the form $\mathcal{T}(\mu)(A) = \int K(x, A)\, d\mu(x)$, induced by a Markov kernel K. A more recent line of work (Bunne et al., 2023) parametrizes $\mathcal{T}$ instead as a pushforward $\mathcal{T}(\mu) = f_{\#}\mu$ for some map $f : \mathbb{R}^d \to \mathbb{R}^d$, with the canonical choice being an optimal transport (OT) map; see e.g. Chewi et al. (2025). Neither family escapes a fundamental expressivity limitation, however, even when the two are combined: whenever two input measures have overlapping supports, so must their images under any Markov operator or pushforward map (Geshkovski et al., 2024b, Section 1.2).

In this work, we depart from the linear setting and consider nonlinear M2M operators. Specifically, we consider a parametric class of pushforward operators where the pushforward map can explicitly depend on the source measure itself. Such operators take the form

$$\mathcal{T}(\mu) = (f_\mu)_{\#}\mu. \qquad (2)$$

Notably, the existence and regularity of such transport maps were recently formalized by (Lavenant & Savaré, 2026), who showed that Lipschitz continuous operators $\mathcal{T}$ can always be represented as (2) with a continuous pushforward map f_μ—omitting trivial obstructions associated to mass splitting.

We consider two classes of approaches for learning the nonlinear M2M operator in Equation (2): modeling f_μ directly via a one-step transport map (*static* M2M regression; Section 3.3) and modeling f_μ via a vector field (*dynamic* M2M regression; Section 3.4). We implement f_μ using transformers, which inherently provide the corresponding measure-dependent maps required to approximate the operator in Equation (2). Sufficiently deep transformers are universal approximators of M2M operators (Yun et al., 2020;

Alberti et al., 2023; Kratsios et al., 2022; Chiang et al., 2023; Jiang & Li, 2024; Edelman et al., 2022; Jiang et al., 2023; Wang & E, 2024; Petrov et al., 2024; Sander & Peyré, 2024; Geshkovski et al., 2024b), and as a result provide a strong basis for our approaches.

3.3. *Static* transformers for M2M regression

We begin with a natural static formulation where we set the pushforward map in (2) to $f_\mu = F_\mu$ (Section 3.3), where F_μ is estimated using a transformer, $F_\mu^\theta(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$. We use a standard transformer architecture detailed in Section A.6. This induces an M2M operator $\mathcal{T}$ which maps the empirical measure of the inputs to that of the outputs in one-step, i.e.

$$\mathcal{T} : \mu_N = \frac{1}{N}\sum_{i=1}^N \delta_{x_i} \;\longmapsto\; (F^\theta_{\mu_N})_{\#}\mu_N = \frac{1}{N}\sum_{i=1}^N \delta_{F^\theta_{\mu_N}(x_i)}$$

This proposal to use a deep transformer is motivated by their measure dependent characteristics established in Section 3.1. In the following, we introduce a practical and efficient algorithm for training such models.

***Static* M2M training Objective.** Given a dataset of measure pairs $\{(\mu_i, \nu_i)\}_{i=1}^n$, let $\gamma_i \in \Pi(\mu_i, \nu_i)$ denote a chosen coupling between the i-th source and target distributions. By sampling an index i uniformly at random from $\{1, \dots, n\}$ and subsequently drawing samples $(x, y) \sim \gamma_i$, we use stochastic gradient descent to minimize the empirical loss

$$\mathcal{L}_{\text{M2M}}(\theta) = \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{(x,y)\sim\gamma_i}\big[\mathbf{D}\big(F^\theta_{\mu_i}(x), y\big)\big]. \tag{3}$$

In practice, any differential distributional loss $\mathbf{D}$ can be used. In this work, we consider a variety of distributional losses, including the maximum mean discrepancy (MMD) distance, energy distance (ED), and the Wasserstein distances $\mathcal{W}_1, \mathcal{W}_2$. When γ_i are OT couplings, we can additionally consider a mean squared error (MSE) loss. We use the Geometric Loss Functions package (Feydy et al., 2019) to compute differentiable distributional losses, and approximate $\mathcal{W}_1$ and $\mathcal{W}_2$ via the Sinkhorn algorithm (Cuturi, 2013). We label these methods `M2M-MMD`, `M2M-ED`, `M2M-`$\mathcal{W}_1$, `M2M-`$\mathcal{W}_2$, and `M2M-OTMSE`, respectively. We include detailed definitions of the respective distributional distances in Section A.5. We minimize Equation (3) using the Adam optimizer.

As we demonstrate empirically below, this static approach is already competitive across a wide range of M2M tasks. It nevertheless inherits a well-known obstacle: estimating and optimizing distances between probability measures is a notoriously hard statistical problem, especially in high dimensions (Si et al., 2020; Gao, 2023). Indeed, this is precisely the difficulty that score matching was developed to circumvent in the generative-modeling literature, by replacing the optimization of an intractable distributional objective with a tractable pointwise regression on the score function. Motivated by the same principle, we introduce a second, *dynamic* approach in which time-dependent transformers model a velocity field rather than an end-to-end pushforward map and are trained via a pointwise regression objective, thereby sidestepping distributional losses entirely.

3.4. *Dynamic* transformers for M2M regression

In this section, we introduce a novel approach that uses a time-dependent transformer and conditional flow matching (Lipman et al., 2022; Tong et al., 2023) to model the vector field for the iterative evolution of μ_i to ν_i. The resulting model is easier to optimize and more expressive, leveraging auto-regressive inference—iterating for multiple steps with the transformer to produce ν_i—in contrast to the static map that takes one step to output ν_i.

The *dynamic* formulation. We adopt a dynamical viewpoint inspired by Otto calculus and the geometry of the Wasserstein space; see, e.g., Chewi et al. (2025). Recall that the 2-Wasserstein space $\mathcal{P}_2(\mathbb{R}^d)$ admits a formal differential structure in which tangent vectors at a measure μ are identified with vector fields $\chi : \mathbb{R}^d \to \mathbb{R}^d$ satisfying suitable integrability conditions. With this structure, a curve $(\mu(t))_{t\in[0,T]}$ is absolutely continuous in the 2-Wasserstein metric if and only if for a time-dependent velocity field $\chi_t[\mu(t)] \in L^2(\mu(t))$, it satisfies a measure-dependent continuity equation of the form

$$\partial_t \mu(t) + \nabla\cdot\big(\mu(t)\,\chi_t[\mu(t)]\big) = 0. \tag{4}$$

In classical OT, admissible velocity fields are further restricted to gradients of scalar potentials, yielding a Riemannian structure on $\mathcal{P}_2(\mathbb{R}^d)$. We do not impose this restriction and allow for general, possibly non-gradient vector fields, which still generate absolutely continuous curves while significantly enlarging the class of admissible dynamics.

Given a family of velocity fields $(\chi_t[\mu(t)])_t$ and an initial condition $\mu(0)$, the continuity equation uniquely determines a curve $(\mu(t))_{t\in[0,T]}$. The corresponding M2M operator is defined as the *flow map* $\mathcal{T} : \quad \mu(0) \longmapsto \mu(T)$. As established in Section 3.1, transformers naturally define measure-dependent maps from $\mathbb{R}^d$ to $\mathbb{R}^d$ and therefore have precisely the structure of admissible velocity fields $\chi_t[\mu(t)]$. This motivates our key proposal: to *use a time-dependent transformer to parameterize the velocity fields* in the continuity equation. Integrating the resulting dynamics yields an absolutely continuous curve in $\mathcal{P}(\mathbb{R}^d)$ and, through its endpoint, a nonlinear M2M operator.

***Dynamic* M2M training objective.** To learn $\chi_t[\mu(t)]$, we introduce *Transformer Flow Matching* (`M2M-TFM`). As established in Section 3.1, transformers naturally process em-

pirical measures as sets of tokens, allowing them to define measure-dependent maps without requiring explicit conditioning networks. This makes them perfectly suited to parameterize the velocity fields in the measure-dependent continuity equation, which we denote as a time-dependent transformer $\chi_t^\theta[\mu(t)](\cdot) : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$. To train $\chi_t^\theta[\mu(t)]$, we extend Conditional Flow Matching (CFM) (Lipman et al., 2022; Tong et al., 2023) to marginalize over the training set of measure pairs. This is akin to the setup of (Atanackovic et al., 2025), but eschewing the need for a secondary model to capture distribution-dependence. Specifically, for a sampled pair $(x, y) \sim \gamma_i$, we define a continuous-time interpolation path $\phi_t(x, y)$, with $\phi_{t=0}(x, y) = x$ and $\phi_{t=1}(x, y) = y$, inducing the measure-level path $\mu_{i,t} = (\phi_t)_\sharp \gamma_i$. The parameters θ are then optimized by minimizing the following empirical loss:

$$\mathcal{L}_{\text{TFM}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{(x,y)\sim\gamma_i} \int_0^1 \left\| \chi_t^\theta[\mu_{i,t}](\phi_t(x,y)) - \frac{\partial}{\partial t}\phi_t(x,y) \right\|^2 dt. \quad (5)$$

In this work, we focus on linear paths of the form $\phi_t(x, y) = (1 - t)x + ty$, which yields the constant velocity $\frac{\partial}{\partial t}\phi_t(x, y) = y - x$. This induces the corresponding measure-level path $\mu_{i,t} = ((1 - t)\phi_{t=0} + t\phi_{t=1})_\sharp \gamma_i$. We restrict the couplings γ_i to OT plans, which can be efficiently approximated via mini-batch OT (Nguyen et al., 2022; Fatras et al., 2021). We optimize Equation (5) using the Adam optimizer and provide the pseudo-code for our training algorithm in Algorithm 1. Extensions to other conditional paths are left for future work.

At test-time, given an *unseen* source measure μ_j, we evaluate M2M-TFM by predicting the corresponding ν_i through solving the empirical ODE $\frac{d}{dt} z_t = \chi_t^\theta[\hat{\mu}_t](z_t)$ from $t = 0$ to $t = 1$ (e.g., via Euler integration). This realizes a prediction for the unseen measure $\hat{\nu}_j = (f_{\mu_j})_\sharp \mu_j$. We provide pseudo-code for inference/sampling in Algorithm 2. See Section A.6 for details regarding architecture and hyperparamters for *static* and *dynamic* models.

4. Related Work

Our work touches on three lines of research that have largely developed independently: (i) M2M regression and learning M2M operators (discussed in Section 1), (ii) transformers as non-linear operators on measures, and (iii) extensions of flow matching to the space of probability measures. Here we review the most closely related works and position our contribution relative to them.

Transformers as nonlinear operators on measures. Transformers have successfully been applied to learning complex functions on point cloud data (Lee et al., 2019; Yang et al., 2019; Zhao et al., 2021; Haviv et al., 2025). There is a growing body of work exploring transformers as nonlinear, permutation-equivariant operators acting on empirical measures (Geshkovski et al., 2024b; 2025; Lavenant & Savaré, 2026). However, there is less work on practically using transformers as learned operators between probability measures. Zimin et al. (2025) used a transformer for M2M regression for Gaussian mixture distributions, focused around the clustering capabilities of transformers optimized with a squared quadratic Wasserstein distance. Adduri et al. (2025) developed a more complex procedure specifically for predicting the transformations of cell sets by optimizing a transformer via an ED loss. The approach differs, in that in our own experiments with cell data, we directly input the cell description into the transformer, bypassing the cell (state) embedding model.

Algorithm 1 (*Training step*)

```
# Inputs:
#  Dataset: M = {(μ_i, ν_i)}_{i=1}^n
#       with μ_i ∈ R^{N_{μ_i} × d}, ν_i ∈ R^{N_{ν_i} × d}
#  b: (num measures) m: (particles per measure)
#  θ: Transformer parameters, χ_t^θ[μ(t)](·)

x, y = sample_measures(M, b, m)
# tuple ([b, m, d], [b, m, d])

x, y = ot_couplings(x, y)
# if using mini-batch OT

t = sample_uniform(0, 1) # sample b t's
z = t * y + (1-t) * x

v_target = y - x
v_pred = Transformer(z, t, θ)

loss = MSE(v_target, v_pred)
θ = update(loss, θ)
```

Algorithm 2 (*Inference / sampling*)

```
# Inputs:
#   Test Measure: μ_j ∈ R^{N_{μ_j} × d} ,
#   θ: Transformer parameters, χ_t^θ[μ(t)](·)
#   num_steps: number of simulation steps

dt = 1/num_steps, t = 0
z = μ_j # [1, N_{μ_j}, d]

for _ in range(num_steps):
    t += dt
    z += Transformer(z, t, θ) * dt

return z # approximates true ν_j
```

Learning flows on spaces of probability measures. A related line of work extends flow matching to the space of probability measures, and naturally splits according to the problem it addresses: *M2M regression* or *distribution alignment*. We discuss each in turn.

For M2M regression—the problem we consider—the most closely related approach is *Meta Flow Matching* (Meta-FM) of Atanackovic et al. (2025), which jointly learns a distribution-dependent vector field and a distribution embedding via flow matching. This setup requires alternating optimization between two separate networks (the

vector field and the embedding model), and the distributional dependence enters only through the initial condition at $t = 0$ rather than along the entire trajectory. Concurrent work by Fishman et al. (2026) pursues a closely related direction, using distributional encoders to model distribution-conditioned transport in the spirit of `Meta-FM`.

A second, distinct body of work addresses *distribution alignment* on $\mathcal{P}(\mathbb{R}^d)$, which is best understood by analogy with the Euclidean case. On $\mathbb{R}^d$, generative modeling learns to sample from an unknown distribution given i.i.d. samples, whereas regression learns a specific map $f : \mathbb{R}^d \to \mathbb{R}^d$ from input–output pairs (x_i, y_i). Distribution alignment is the natural lift of *generative modeling* to $\mathcal{P}(\mathbb{R}^d)$: the training data are themselves measures $\mu_1, \ldots, \mu_n$, viewed as i.i.d. samples from an unknown *meta-distribution* on $\mathcal{P}(\mathbb{R}^d)$, and the goal is to sample new measures from this same meta-distribution. M2M regression, by contrast, is the lift of *regression*: the data come as input–output pairs of measures (μ_i, ν_i), and the goal is to recover the operator that maps each μ_i to its specific ν_i. In this distribution-alignment category, Haviv et al. (2025) introduce *Wasserstein Flow Matching* (`WFM`), which lifts the flow matching framework of Lipman et al. (2022) from $\mathbb{R}^d$ to $\mathcal{P}(\mathbb{R}^d)$ using a permutation-equivariant velocity field, typically a transformer. Amos et al. (2022) pursue a similar goal via meta–optimal transport with input-convex neural networks, and Sakalyan et al. (2025) apply variational flow matching with transformers to the generation of complex biological states. Because these methods are designed to match meta-distributions rather than individual input–output pairs, they yield sub-optimal solutions when applied directly to M2M regression; a gap we document empirically in Section 5.1. A complementary direction uses related tools to learn distributional *representations* (Fishman et al., 2025).

5. Experiments

In this section, we evaluate our method's ability to generalize to previously *unseen* test measures. For this we consider three experimental settings: **(i: *synthetic multi-measure objects***, Section 5.1) a dataset of M measure pairs, (**ii: *dynamics***, Section 5.2) simulated population dynamics on multi-measure-multi-timepoint Mckean-Vlasov systems, and (**iii: *a real world application***, Section 5.3) prediction of response to cancer treatment for *unseen* patients. In all settings, the objective is to evaluate generalization performance in unseen measures. We describe our individual experimental setups, implementation details, and findings in the following sections.

Baselines. We compare our approaches (`M2M-TFM`, `M2M-ED`, `M2M-`$\mathcal{W}_1$) to both non-measure-dependent and measure-dependent methods. We consider Conditional Flow Matching (`CFM`) (Lipman et al., 2022; Tong et al., 2023) as the non-measure-dependent baseline.[3] We consider `Meta-FM` (Atanackovic et al., 2025) as a flow matching-based measure-dependent approach tailored for the problem of M2M regression. We additionally consider `WFM` as a baseline (which assumes unpaired measures) for the synthetic multi-measure objects experiments (***ii***, Section 5.1). For the multi-time-point McKean-Vlasov experiments (***ii***, Section 5.2), we additionally consider two state-of-the-art baselines: the Neural McKean-Vlasov Process (`NMKV`) Yang et al. (2024) and the mean-field transformer (`MF-Transformer`) (Biswal et al., 2025). We use the best performing variant of `NMKV`, the *empirical measure architecture*. We provide further details in Section A.4.

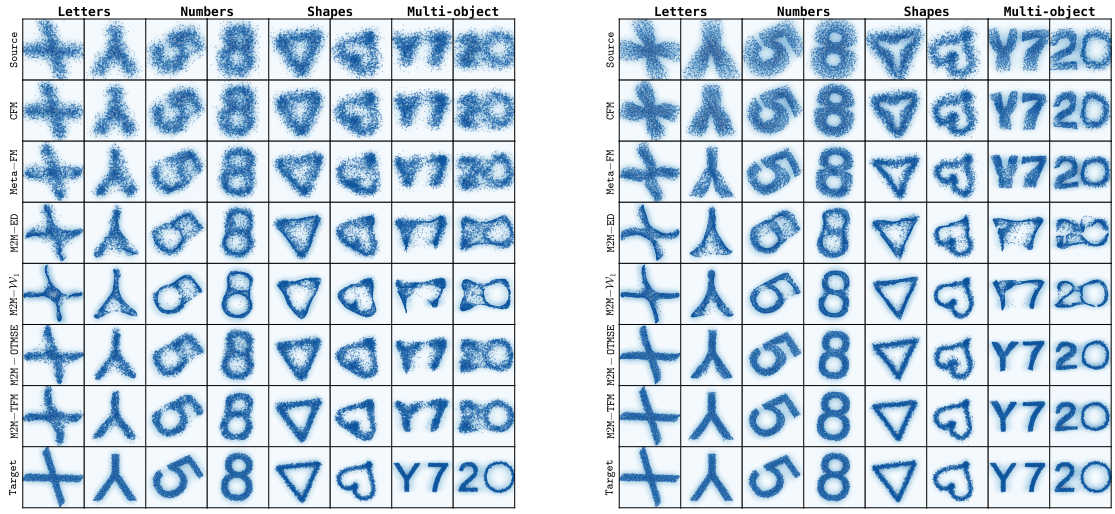
5.1. Paired multi-measure objects

Setup. We consider the synthetic letters *silhouettes* setup from Atanackovic et al. (2025), with several adjustments to increase the difficulty of the learning problem. To construct a dataset of measure pairs $\mathcal{M} = \{(\mu_i, \nu_i)\}_{i=1}^{M=240}$, we first define a set of *target* measures $\{\nu_i\}_{i=1}^{M}$, using 2D letter shapes. We use all letters, except "X" and "Y", with each letter in 10 random orientations. We place N_i particles (approximately 2000 ± 500 per-measure) on the silhouettes of each target measure ν_i, and then apply a corruption process to acquire μ_i. We use both the forward diffusion from Atanackovic et al. (2025) and a *kernel interaction* corruption process defined in Section A.1. Unlike the setup in Atanackovic et al. (2025), we shuffle all the samples/particles between the source and target measures of each object silhouette.

Models are tasked with learning the reverse corruption process ($\mu \to \nu$) and are evaluated on unseen measures. Since particles within measures are unpaired, we use mini-batch OT to compute couplings between particles every training step. We evaluate performance on four categories of left-out measures, each with 1500 to 2000 particles; *letters* "X" and "Y", *numbers*, *shapes*, and *multi-object* compositions. We use the $\mathcal{W}_1$ and ED distributional metrics to quantify performance.

Results. We report quantitative results in Table 1 and visualization of model predictions in Figure 2. We observe that `M2M-TFM` outperforms all other approaches across all left-out measure categories, including having lower variance, while our corresponding static approaches out-perform baselines. We observe that indeed `WFM` struggles to compete with approaches tailored for the M2M regression problem as it is required to learn the true corresponding *distribution alignment* between measures in addition to the sample-level transport maps. We include extended quantitative results

[3] We remark that we use "`CFM`" to denote two different variants used separately in each experimental setting. In Section 5.1 and Section 5.2 we use mini-batch `OT-CFM` directly and in Section 5.3 we use `OT-CFM` with precomputed couplings.



(a) **Diffusion process** *(b)* **Kernel interaction process**

Figure 2. **Visualization of model predictions on multi-measure objects for unseen measures.** Methods are tasked with transporting particles from *source* measures (top row) to *target* measures (bottom row). We show visuals for models trained to reverse (a) a diffusion process, and (b) a kernel interaction process.

Table 1. **Prediction performance comparison on multi-measure objects.** Mean ± Std computed over 5 random model seeds and 8 "*object*" silhouettes. **Bold** indicates the best performing method.

	Diffusion		Kernel interactions	
Method	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)
Baselines				
CFM	0.2491 ± 0.0032	0.0071 ± 0.0004	0.2744 ± 0.0030	0.0128 ± 0.0005
Meta-FM	0.2306 ± 0.0114	0.0082 ± 0.0017	0.2292 ± 0.0150	0.0130 ± 0.0022
WFM	0.5834 ± 0.1399	0.1201 ± 0.0813	0.5226 ± 0.0545	0.0931 ± 0.0303
Static (ours)				
M2M-MMD	0.2036 ± 0.0094	0.0073 ± 0.0012	0.1725 ± 0.0312	0.0068 ± 0.0018
M2M-ED	0.2204 ± 0.0242	0.0087 ± 0.0026	0.2155 ± 0.0478	0.0106 ± 0.0035
M2M-$\mathcal{W}_2$	0.2288 ± 0.0151	0.0102 ± 0.0023	0.1888 ± 0.0380	0.0080 ± 0.0025
M2M-$\mathcal{W}_1$	0.2092 ± 0.0055	0.0085 ± 0.0005	0.1663 ± 0.0235	0.0069 ± 0.0009
M2M-OTMSE	0.1680 ± 0.0032	0.0044 ± 0.0003	0.1334 ± 0.0088	0.0053 ± 0.0009
Dynamic (ours)				
M2M-TFM	$\mathbf{0.1300 \pm 0.0009}$	$\mathbf{0.0023 \pm 0.0001}$	$\mathbf{0.0780 \pm 0.0020}$	$\mathbf{0.0020 \pm 0.0002}$

Table 19 and Table 20, as well as extended visualizations in Figure 5.

5.2. Multi-timepoint McKean-Vlasov SDEs

We evaluate our approaches for predicting the evolution of synthetic McKean-Vlasov processes (McKean Jr, 1966). We use McKean-Vlasov processes because their dynamics are distribution dependent and therefore a measure-to-measure approach is required for accurate prediction.

Setup. For each McKean-Vlasov experiment, the models are trained on 70 systems, and tested on 10 systems. Each training system consists of a 100 element measure tuple, $(\mu_0, \mu_1, \cdots, \mu_{99})$ representing 100 time-points in the evolution of the McKean-Vlasov system. We train the model by regressing between all time-points independently. At each step we train on a sub-sample $b = 16$ of the possible $\{(\mu_t, \mu_{t+1})\}_{t=0}^{t=98}$ pairs. For evaluation the model receives a new $\{\mu_1$ and must predict $\hat{\mu}_2, \hat{\mu}_3, \cdots, \hat{\mu}_{100}$, with evaluation metrics averaged over all 100 time-points. We train

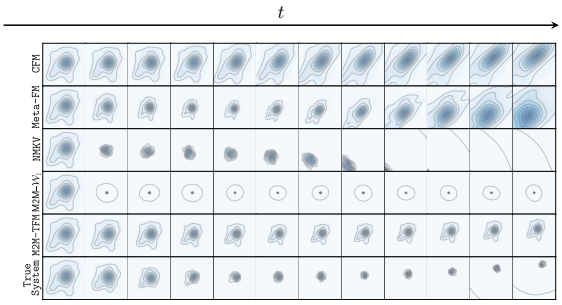


Figure 3. **Visualization of various model predictions on 2 dimensional Kuramoto system.** We set new initial conditions, apply the kuramoto dynamics to it and have the models predict. For static methods we only show M2M-$\mathcal{W}_1$.

and evaluate on 9 McKean-Vlasov processes, the Kuramoto model, Mean-Field Atlas, and the FitzHugh-Nagumo model, each with a 2, 50, and 100 dimensional versions. We adapt the McKean-Vlasov processes from Yang et al. (2024), with modified dynamics and evaluation in new measures. Details about each of the systems, the initial conditions used, and modifications to the dynamics are provided in Section A.2. We provide the hyperparameters for all methods in Section A.6.

Results. We report results in Table 2, using the fifty dimensional processes. We provide results for all three processes in 2, 50 and 100 dimensions in Section C. We observe that M2M-TFM outperforms in almost all the systems. The consistency of our method's strong performance across dimensions demonstrates the scalability and the performance across systems indicates the expressivity. NMKV performs worse than one might expect, potentially because it is built for generalizing within a measure, rather than between mea-

Table 2. **Prediction performance comparison on 100 dimensional McKean-Vlasov systems.** Mean±Std computed over 6 random seeds. **Bold** indicates the best performing method. We include extended results in Section C.2

	Kuramoto		FitzHugh-Nagumo		Atlas	
Method	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)
Baselines						
CFM	11.312 ± 0.119	6.978 ± 0.135	19.418 ± 0.108	2.435 ± 0.044	19.434 ± 0.115	7.132 ± 0.060
MF-Transformer	13.657 ± 0.422	21.019 ± 0.975	25.044 ± 0.711	8.012 ± 1.000	24.509 ± 0.355	23.359 ± 0.707
NMKV	9.978 ± 0.702	14.130 ± 1.306	20.049 ± 0.320	11.618 ± 0.670	28.830 ± 1.785	30.867 ± 3.830
Meta-FM	7.303 ± 0.204	6.231 ± 0.322	18.683 ± 0.240	5.173 ± 0.177	24.891 ± 0.464	14.720 ± 0.496
Static (ours)						
M2M-$\mathcal{W}_1$	3.684 ± 0.122	4.786 ± 0.246	15.688 ± 0.136	8.271 ± 0.266	21.838 ± 0.996	26.020 ± 1.946
M2M-ED	13.524 ± 0.403	14.461 ± 2.041	22.113 ± 0.691	4.405 ± 0.791	27.282 ± 0.183	10.783 ± 0.059
M2M-OTMSE	8.636 ± 0.916	14.709 ± 1.833	18.266 ± 0.481	10.137 ± 0.835	21.329 ± 0.429	25.211 ± 0.855
Dynamic (ours)						
M2M-TFM	$\mathbf{3.229 \pm 0.055}$	$\mathbf{2.584 \pm 0.095}$	$\mathbf{14.710 \pm 0.113}$	$\mathbf{1.121 \pm 0.158}$	$\mathbf{17.345 \pm 0.059}$	$\mathbf{6.649 \pm 0.096}$

sures. Further, the Meta-FM baseline is often worse than M2M-TFM we suspect this is because it can only model dependence from the initial time-point, along with needing to split its optimization steps between two models. We provide an ablation on the number of simulation steps used for M2M-TFM in Section C.2. We visually demonstrate the superiority of our approach on one of the 2-dimensional Kuramoto-model systems in Figure 3.

5.3. Patient-specific treatment response for colorectal cancer

Setup. Here, we consider the problem of predicting cellular response in novel patients. We use a large-scale multi-donor patient-derived organoid (PDOs)[4] dataset introduced by Zapatero et al. (2023). This dataset comprises single-cell (the *samples*) mass-cytometry readouts over 44 bio-markers (features) from PDOs derived from 10 patients. Response screens are conducted over varying treatments, tumor micro-environment conditions, and replicated experiments. As a result, the dataset consists of a large quantity of treatment response populations coupled with a corresponding control population (the *measure* pairs (μ_i, ν_i)). We follow the pre-processing steps from Atanackovic et al. (2025) and define three evaluation splits based on held-out patients (labeled PDO-21, PDO-27, and PDO-75). We use all 44 bio-markers available. After applying additional quality control steps (detailed in Section A.3), the resulting quantity of train/test measure pairs are (PDO-21:$393/62$, PDO-27:$404/51$, PDO-75:$390/65$). We quantify prediction performance using distributional distances $\mathcal{W}_1$, ED, and r^2 (overall average correlation coefficient) for each left-out test measure. We provide further experimental details in Section A.3.

[4]A patient-derived organoid (PDO) is a biological model system which is constructed using donor cells from a patient. PDOs capture both the physical/functional characteristics and the genetic profile of the original tissue.

Table 3. **Comparison of method prediction performance on *unseen* patients on the patient-derived organoid dataset.** We report mean and standard deviation across 3 left-out-patients and 4 model seeds per-patient. **Bold** indicates the top performer and underline indicates the second best performer.

	$\mathcal{W}_1 \downarrow$	ED $\downarrow$	$r^2 \uparrow$
Baselines			
CFM	4.4099 ± 0.2580	$\underline{0.4828} \pm \underline{0.1695}$	0.8961 ± 0.0153
Meta-FM	4.4358 ± 0.2240	0.5160 ± 0.1358	$\underline{0.8986} \pm \underline{0.0104}$
Static (ours)			
M2M-ED	4.5184 ± 0.2323	0.5689 ± 0.1551	0.8882 ± 0.0101
M2M-$\mathcal{W}_1$	$\mathbf{4.1005 \pm 0.2270}$	0.8270 ± 0.1242	0.7660 ± 0.0228
M2M-OTMSE	$\underline{4.2772} \pm \underline{0.1792}$	1.0085 ± 0.0962	0.7211 ± 0.0210
Dynamic (ours)			
M2M-TFM	4.3526 ± 0.2175	$\mathbf{0.4715 \pm 0.1319}$	$\mathbf{0.9029 \pm 0.0050}$

Results. We report results for the task of predicting patient-specific treatment response for unseen patients in Table 3. Although by construction of the dataset each patient comprises of numerous replicate experiments of pre- and post-treatment single cell population (measure pairs), this task is especially difficult as measurements are only available for 10 patients in this dataset. As a result, generalizing to an unseen patient (model trained on data from 9 patients, and evaluated on the left-out patient) requires a model to learn the M2M operator from limited patient-level "observations". We observe that our static approaches perform best on the metrics corresponding to their respective losses (i.e. ED-loss and $\mathcal{W}_1$-loss), while exhibiting worse performance on the counterpart metrics. In contrast, M2M-TFM yields a balanced performance across all metrics. The strong $\mathcal{W}_1$ performance of M2M-$\mathcal{W}_1$ suggests that measure-dependence is beneficial here, while M2M-TFM's balanced performance makes it the most robust choice in practice.

6. Conclusion & Discussion

In this work, we established transformers as a natural and powerful architecture for measure-to-measure (M2M) regression. By exploiting the inherent measure-dependence

of transformers, we introduced both *static* and *dynamic* frameworks for learning nonlinear M2M regression operators. Notably, `M2M-TFM` (our *dynamic* approach) integrates flow matching with the measure-dependent nature of transformers, yielding competitive performance across a range of M2M regression tasks. Together, our novel methods provide practical, scalable, and expressive solutions for M2M regression, which we demonstrated through a variety of empirical experiments, including the challenging real-world task of predicting patient-specific treatment response.

Limitations & Future work. While our proposed methods offer practical and expressive solutions for M2M regression, they are currently limited to modeling simplified transport dynamics of physical processes. For instance, in this work we only considered linear interpolation paths for our *dynamic* M2M regression approach (i.e. `M2M-TFM`), yet many physical processes, such as cell dynamics, inherently exhibit nonlinear, stochastic, and birth-death dynamics. Extending `M2M-TFM` to settings with non-gradient dynamics (Petrović et al., 2026; Rathod et al., 2026; Guan et al., 2026), stochasticity (Tong et al., 2024), and the unbalanced setting (Peng et al., 2026) presents several directions for future advancement.

Another avenue for future research involves exploring the use of our static approaches as one-step generative models, aligned with the recent advancements in this direction (Deng et al., 2026; Zhu et al., 2026; Geng et al., 2026; Song & Dhariwal, 2024; Boffi et al., 2024; Potaptchik et al., 2026). For example, our *static* approaches yield one-step transport maps from arbitrary source distributions and it is straightforward to adapt our framework for the task of generative modeling. Lastly, our work is also amenable to extensions and applications on other large-scale perturbation screen datasets, such as Zhang et al. (2025) and Huang et al. (2025), which we leave for future work.

Beyond empirical extensions, addressing certain theoretical questions for *dynamic* M2M regression with transformers remains an important and open direction. For instance, we do not formalize any guarantees for learning nonlinear M2M operators via vector field models under our flow matching framework. We leave proving such guarantees and other work in this direction as future work. Ultimately, our work establishes two transformer-based approaches as a natural and scalable paradigm for M2M regression, providing a versatile foundation for these subsequent questions and future work.

Acknowledgments

This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada CIFAR AI Chairs program. The research was also enabled by computational resources provided by the Digital Research Alliance of Canada (https://alliancecan.ca) and the Alberta Machine Intelligence Institute (https://www.amii.ca/). In addition, LA was in-part supported by the Eric and Wendy Schmidt Center at the Broad Institute of MIT and Harvard, and by the NSERC Postdoctoral Fellowship.

Impact Statement

This paper presents mainly theoretical and methodological additions to the field of generative modeling. There are numerous potential societal consequences of generative modeling but none are unique to our work nor which need to be highlighted here. Notably our work does not focus on generating images or audio which are known to be of higher risk in the field of generative AI. We focus mainly on regressing between distributions which is not more or less societally impactful than other generative AI works.

References

Adduri, A. K., Gautam, D., Bevilacqua, B., Imran, A., Shah, R., Naghipourfar, M., Teyssier, N., Ilango, R., Nagaraj, S., Dong, M., et al. Predicting cellular responses to perturbation across diverse contexts with state. *BioRxiv*, 2025.

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Alberti, S., Dern, N., Thesing, L., and Kutyniok, G. Sumformer: Universal approximation for efficient transformers. In *Topological, Algebraic and Geometric Learning Workshops (TAGL)*, 2023.

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

Armingol, E., Officer, A., Harismendy, O., and Lewis, N. E. Deciphering cell–cell interactions and communication from gene expression. *Nature Reviews Genetics*, 2021.

Atanackovic, L., Zhang, X., Amos, B., Blanchette, M., Lee, L. J., Bengio, Y., Tong, A., and Neklyudov, K. Meta flow matching: Integrating vector fields on the wasserstein manifold. In *International Conference on Learning Representations*, 2025.

Biswal, S., Elamvazhuthi, K., and Sonthalia, R. Universal approximation of mean-field models via transformers. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Boffi, N. M., Albergo, M. S., and Vanden-Eijnden, E. Flow map matching with stochastic interpolants: A mathemat-

ical framework for consistency models. *arXiv preprint arXiv:2406.07507*, 2024.

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

Castin, V., Ablin, P., and Peyré, G. How smooth is attention? In *International Conference on Machine Learning*, 2024.

Castin, V., Ablin, P., Carrillo, J. A., and Peyré, G. A unified perspective on the dynamics of deep transformers. *arXiv preprint arXiv:2501.18322*, 2025.

Chen, S., Lin, Z., Polyanskiy, Y., and Rigollet, P. Quantitative clustering in mean-field transformer models, 2025a. arXiv:2504.14697.

Chen, S., Lin, Z., Polyanskiy, Y., and Rigollet, P. Critical attention scaling in long-context transformers. In *International Conference on Learning Representations*, 2026.

Chen, Z., Polyanskiy, Y., and Rigollet, P. Clustering in self-attention dynamics with Wasserstein-Fisher-Rao gradient flows. In *NeurIPS workshop on Dynamics at the Frontiers of Optimization, Sampling, and Games*, 2025b.

Chewi, S., Niles-Weed, J., and Rigollet, P. *Statistical optimal transport*, volume 2364 of *Lecture Notes in Mathematics*. Springer, Cham, 2025.

Chiang, D., Cholak, P., and Pillay, A. Tighter bounds on the expressivity of transformer encoders. In *International Conference on Machine Learning*, 2023.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. *Neural Information Processing Systems*, 26, 2013.

Deng, M., Li, H., Li, T., Du, Y., and He, K. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Dixit, A., Parnas, O., Li, B., Chen, J., Fulco, C. P., Jerby-Arnon, L., Marjanovic, N. D., Dionne, D., Burks, T., Raychowdhury, R., et al. Perturb-seq: dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. *cell*, 167(7), 2016.

Edelman, B. L., Goel, S., Kakade, S., and Zhang, C. Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning*, 2022.

Fatras, K., Zine, Y., Majewski, S., Flamary, R., Gribonval, R., and Courty, N. Minibatch optimal transport distances; analysis and applications. *arXiv preprint arXiv:2101.01792*, 2021.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouve, A., and Peyré, G. Interpolating between optimal transport and mmd using sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Fishman, N., Gowri, G., Yin, P., Gootenberg, J., and Abudayyeh, O. Generative distribution embeddings. *arXiv preprint arXiv:2505.18150*, 2025.

Fishman, N., Gowri, G., Fischer, P. L., Zitnik, M., Abudayyeh, O., and Gootenberg, J. Distribution-conditioned transport. *arXiv preprint arXiv:2603.04736*, 2026.

Gao, R. Finite-sample guarantees for wasserstein distributionally robust optimization: Breaking the curse of dimensionality. *Operations Research*, 2023.

Geng, Z., Deng, M., Bai, X., Kolter, Z., and He, K. Mean flows for one-step generative modeling. *Neural Information Processing Systems*, 2026.

Geshkovski, B., Letrouit, C., Polyanskiy, Y., and Rigollet, P. The emergence of clusters in self-attention dynamics. In *Neural Information Processing Systems*, 2024a.

Geshkovski, B., Rigollet, P., and Ruiz-Balet, D. Measure-to-measure interpolation using transformers. *arXiv preprint arXiv:2411.04551*, 2024b.

Geshkovski, B., Letrouit, C., Polyanskiy, Y., and Rigollet, P. A mathematical perspective on transformers. *Bulletin of the American Mathematical Society*, 62(3), 2025.

Geuter, J., Bonet, C., Korba, A., and Alvarez-Melis, D. DDEQs: Distributional deep equilibrium models through wasserstein gradient flows. In *International Conference on Artificial Intelligence and Statistics*, 2025a.

Geuter, J., Kornhardt, G., Tomasson, I., and Laschos, V. Universal neural optimal transport. In *International Conference on Machine Learning*, 2025b.

Ghodrati, L. and Panaretos, V. M. Distribution-on-distribution regression via optimal transport maps. *Biometrika*, 2022.

Ghodrati, L. and Panaretos, V. M. Transportation of measure regression in higher dimensions. *arXiv preprint arXiv:2305.17503*, 2023.

Girshfeld, I. and Chen, X. Neural local wasserstein regression. In *Proceedings of the Conference on Topology, Algebra, and Geometry in Data Science (TAG-DS)*, 2025.

Goodenough, D. A. and Paul, D. L. Gap junctions. *Cold Spring Harbor perspectives in biology*, 2009.

Guan, V., Atanackovic, L., and Neklyudov, K. A call to lagrangian action: Learning population mechanics from temporal snapshots. *arXiv preprint arXiv:2605.08550*, 2026.

Haviv, D., Pooladian, A.-A., Pe'er, D., and Amos, B. Wasserstein flow matching: Generative modeling over families of distributions. In *International Conference on Machine Learning*, 2025.

Hu, V. T., Zhang, W., Tang, M., Mettes, P., Zhao, D., and Snoek, C. Latent space editing in transformer-based flow matching. In *AAAI conference on artificial intelligence*, 2024.

Huang, A. C., Hsieh, T.-H. S., Zhu, J., Michuda, J., Teng, A., Kim, S., Rumsey, E. M., Lam, S. K., Anigbogu, I., Wright, P., et al. X-atlas/orion: Genome-wide perturb-seq datasets via a scalable fix-cryopreserve platform for training dose-dependent biological foundation models. *bioRxiv*, 2025.

Jiang, H. and Li, Q. Approximation rate of the transformer architecture for sequence modeling. In *Neural Information Processing Systems*, 2024.

Jiang, H., Li, Q., Li, Z., and Wang, S. A brief survey on the approximation theory for sequence modelling, 2023. arXiv preprint arXiv:2302.13752.

Jing, B., Stärk, H., Jaakkola, T., and Berger, B. Generative modeling of molecular dynamics trajectories. *Neural Information Processing Systems*, 2024.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017.

Kratsios, A., Zamanlooy, B., Liu, T., and Dokmanić, I. Universal approximation under constraints is possible with transformers. In *International Conference on Learning Representations*, 2022.

Lavenant, H. and Savaré, G. Continuous transformations of probability measures and their transport representations. *arXiv preprint arXiv:2604.16653*, 2026.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pp. 3744–3753. PMLR, 2019.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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

Ma, N., Goldstein, M., Albergo, M. S., Boffi, N. M., Vanden-Eijnden, E., and Xie, S. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*. Springer, 2024.

McKean Jr, H. P. A class of markov processes associated with nonlinear parabolic equations. *Proceedings of the National Academy of Sciences*, 56(6):1907–1911, 1966.

Muandet, K., Fukumizu, K., Sriperumbudur, B., and Schölkopf, B. Kernel mean embedding of distributions: A review and beyond. *Found. Trends Mach. Learn.*, 10, June 2017.

Nguyen, K., Nguyen, D., Nguyen, Q., Pham, T., Bui, H., Phung, D., Le, T., and Ho, N. On transportation of mini-batches: A hierarchical approach. In *International Conference on Machine Learning*, 2022.

Oliva, J. B., Póczos, B., and Schneider, J. Distribution to distribution regression. In *International Conference on International Conference on Machine Learning*, 2013.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

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. scperturb: harmonized single-cell perturbation data. *Nature Methods*, 2024.

Peng, Q., Wang, Z., Ying, J., Sun, Y., Nie, Q., Zhang, L., Li, T., and Zhou, P. Wfr-fm: Simulation-free dynamic unbalanced optimal transport. *arXiv preprint arXiv:2601.06810*, 2026.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. Film: visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2018.

Petersen, A. and Müller, H.-G. Fréchet regression for random objects with euclidean predictors. *The Annals of Statistics*, 2019.

Petrov, A., Torr, P. H., and Bibi, A. Prompting a pretrained transformer can be a universal approximator, 2024. arXiv preprint arXiv:2402.14753.

Petrović, K., Atanackovic, L., Moro, V., Kapuśniak, K., Ceylan, I. I., Bronstein, M. M., Bose, J., and Tong, A. Curly flow matching for learning non-gradient field dynamics. In *Neural Information Processing Systems*, 2026.

Potaptchik, P., Saravanan, A., Mammadov, A., Prat, A., Albergo, M. S., and Teh, Y. W. Meta flow maps enable scalable reward alignment. *arXiv preprint arXiv:2601.14430*, 2026.

Rathod, S. S., Liò, P., and Zhang, X. Splineflow: Flow matching for dynamical systems with b-spline interpolants. *arXiv preprint arXiv:2601.23072*, 2026.

Rohbeck, M., Brouwer, E. D., Bunne, C., Huetter, J.-C., Biton, A., Chen, K. Y., Regev, A., and Lopez, R. Modeling complex system dynamics with flow matching across time and conditions. In *International Conference on Learning Representations*, 2025.

Sakalyan, K., Palma, A., Guerranti, F., Theis, F. J., and Günnemann, S. Modeling microenvironment trajectories on spatial transcriptomics with nicheflow. *arXiv preprint arXiv:2511.00977*, 2025.

Sander, M. E. and Peyré, G. Towards understanding the universality of transformers for next-token prediction, 2024. arXiv preprint arXiv:2410.03011.

Sander, M. E., Ablin, P., Blondel, M., and Peyré, G. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022.

Si, N., Blanchet, J., Ghosh, S., and Squillante, M. Quantifying the empirical wasserstein distance to a set of measures: Beating the curse of dimensionality. *Neural Information Processing Systems*, 2020.

Song, Y. and Dhariwal, P. Improved techniques for training consistency models. In *International Conference on Learning Representations*, 2024.

Su, J., Song, Y., Zhu, Z., Huang, X., Fan, J., Qiao, J., and Mao, F. Cell–cell communication: new insights and clinical implications. *Signal transduction and targeted therapy*, 2024.

Szabó, Z., Sriperumbudur, B. K., Póczos, B., and Gretton, A. Learning theory for distribution regression. *Journal of Machine Learning Research*, 2016.

Tong, A., Fatras, K., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Wolf, G., and Bengio, Y. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Tong, A. Y., Malkin, N., Fatras, K., Atanackovic, L., Zhang, Y., Huguet, G., Wolf, G., and Bengio, Y. Simulation-free schrödinger bridges via score and flow matching. In *International Conference on Artificial Intelligence and Statistics*, 2024.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Neural information processing systems*, 2017.

Wang, M. and E, W. Understanding the expressive power and mechanisms of transformer for sequence modeling, 2024. arXiv preprint arXiv:2402.00522.

Wei, Z., Wang, Y., Gao, Y., Wang, S., Li, P., Si, D., Gao, Y., Wu, S., Li, D., Dong, K., et al. Benchmarking algorithms for generalizable single-cell perturbation response prediction. *Nature Methods*, 2025.

Yang, H., Hasan, A., Ng, Y., and Tarokh, V. Neural McKean-Vlasov processes: Distributional dependence in diffusion processes. In Dasgupta, S., Mandt, S., and Li, Y. (eds.), *International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, 2024.

Yang, J., Zhang, Q., Ni, B., Li, L., Liu, J., Zhou, M., and Tian, Q. Modeling point clouds with self-attention and gumbel subset sampling. In *IEEE/CVF conference on computer vision and pattern recognition*, 2019.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020.

Zapatero, M. R., Tong, A., Opzoomer, J. W., O'Sullivan, R., Rodriguez, F. C., Sufi, J., Vlckova, P., Nattress, C., Qin, X., Claus, J., et al. Trellis tree-based analysis reveals stromal regulation of patient-derived organoid drug responses. *Cell*, 186(25):5606–5619, 2023.

Zhang, J., Ubas, A. A., de Borja, R., Svensson, V., Thomas, N., Thakar, N., Lai, I., Winters, A., Khan, U., Jones, M. G., et al. Tahoe-100m: A giga-scale single-cell perturbation atlas for context-dependent gene function and cellular modeling. *BioRxiv*, pp. 2025–02, 2025.

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. Point transformer. In *IEEE/CVF international conference on computer vision*, 2021.

Zhu, H., Xiao, T., Zhou, S., Guo, Z., Zhang, H., Xu, S., and Honavar, V. G. Simple distillation for one-step diffusion models. In *Neural Information Processing Systems*, 2026.

Zimin, A., Kutakh, A., Polyanskiy, Y., and Rigollet, P. Learning gaussian mixture models via transformer measure flows. In *ICML 2025 Workshop on Methods and Opportunities at Small Scale*, 2025.

A. Additional Experimental Details

The relevant code for our experimental results can be found at https://github.com/Matthew-Vandergrift/m2m-transformers.

A.1. Multi-measure Objects

In this section we outline additional experimental setup details for the Multi-measure Objects experiments. The kernel interaction process in this experiment is defined by the following dynamics equation,

$$x_{t+\Delta t} = x_t + \eta(x_t - \sum_j A_{ij}x_j)\Delta t + \sigma\sqrt{\Delta t}\epsilon$$

where $A_{ij} = \exp(-\|x_i - x_j\|^2/2h^2)/\sum_k \exp(-\|x_i - x_k\|^2/2h^2)$ is the normalized Gaussian kernel with bandwidth h, η is the repulsion strength, and σ is the noise scale. For this experiment, we use $\eta = 0.3$, $\sigma = 0.001$, $h = 0.75$, $\Delta t = 0.05$, and simulate the process for 50 steps.

A.2. Mckean-Vlasov Systems

A.2.1. Dynamics for McKean-Vlasov Systems

We use modified version of the McKean-Vlasov Systems from Yang et al. (2024), the Kuramoto model, and the Mean-Field Atlas system, with a modification to the FitzHugh-Nagumo model. We modify the systems to ensure that given different initial measures the resulting end states of the system are different. This was done because when we naively extended these systems to higher dimensions it was observed that the initial distributions had little effect and so the systems were modified. In particular we found that the dynamics listed in Yang et al. (2024) lead to a lower diversity of final states in these cases. We enumerate the equations we used for governing a particular particle i of the N particles at time t, which we denote by $X_{i,t}$ as follows. $X_{i,t}$ is a d dimensional vector representing one particle. Note that $X_{t,j}$ for a specific j denotes another particle in the system at time t, and that W_i is a Wiener random process, sampled individually for each i. Additionally $X_{i,t}^{\text{dim } d}$ to denote the dth dimension of a particle, i.e one of the entries in the d dimensional vector.

For the Kuramoto Model we use,

$$dX_{i,t} = \sin(X_{i,t}) + (2/N)\sum_{j=1}^{N}(\sin(X_{j,t} - X_{i,t}))dt + \sigma * dW_i \text{ for } t \in [0, 5]$$

With initial conditions consisting of 333 particles drawn from $U_{[-1,1]}$ and 167 particles drawn from $\mathcal{N}(A, 0.1)$ with A sampled from $U_{[-1,1]}$. We use $\sigma = 0.2$ and we simulate 100 time points from $t = 0$ to $t = 5.0$.

For the FitzHugh-Nagumo model we use a different equation for the first dimension, $X_{i,t}^1$ and the second dimension $X_{i,t}^2$ when $d = 2$. If $d > 2$ then dimension 1 follows the equation for dimension 1, and all other dimensions follow the equation for dimension 2.

$$\begin{aligned} dX_{i,t}^{\text{dim1}} &= \left(0.2X_{i,t}^{\text{dim1}}(X_{i,t}^{\text{dim1}} - 0.5)(1 - X_{i,t}^{\text{dim1}}) - \frac{1}{n}\sum_{j=1}^{n} X_{j,t}^{\text{dim2}} + I\right)dt \\ &+ \left(\frac{1}{N}\sum_{j=1}^{N}(X_{i,t}^{\text{dim1}} - X_{j,0}^{\text{dim1}})\right)dt \\ dX_{i,t}^{\text{dim2}} &= \left(\frac{-bX_{i,t}^{\text{dim2}} + c * X_{i,t}^{\text{dim2}} + d}{\tau}\right)dt \text{ for } t \in [0, 10] \text{ if d > 2, } t \in [0, 4] \text{ if d = 2} \end{aligned}$$

With initial conditions consisting of 333 particles drawn from $U_{[-6,6]}$ and 167 drawn from $\mathcal{N}(A, 0.1)$ with A drawn from $U_{[-6,6]}$. We use a different value of b, c, d for each dimension. We set the values of each of these coefficients to be evenly spaced over the following ranges $b \in [0.5, 0.8], c \in [0.5, 1.0], d \in [0.3, 0.7]$ and $\tau \in [1.0, 10.0]$. We always set $I = 0.1\sin(10t), \sigma = 0.1$. Notably for the Kuramoto system in 2 dimension we only simulate up to a max time of 4.0.

For the MeanField Atlas we use,

$$dX_{i,t}^{\dim d} = \gamma\left(\frac{1}{N}\sum_{j=1}^{N} 1_{[x_{i,t}-Xj,t\geq 0]}\right) * dt + \sigma * dW_{i,t}$$
$$\text{with } \gamma(x) = 5 * (0.5 - x) + (X_{i,t} - 0.01 * X_{i,t}^3) + 1.5\sin(X_{i,t}^{\dim d-1}) \text{ for } t \in [0, 2]$$

With initial conditions consisting of 333 particles drawn from $U_{[-2,2]}$ and 167 drawn from $\mathcal{N}(A, 0.3)$ where A is drawn from $U_{[-1,1]}$. We use $\sigma = 0.5$, and we simulate for 100 time-points evenly spaced between 0 and 2.0.

A.2.2. Extra Training Details

For training models on these systems with 100 timepoints, we treat each pair of timepoints as if it were its own independent measure pair. To be specific, each system is a 100 element tuple $(\mu_0, \mu_1, \cdots, \mu_{99})$, and we train our model by having it regress between μ_t, μ_{t+1} independently. We generate these pairs to regress between by randomly sampling a batch of 16 time-points from the tuple of length 100. We update on each of these 16 pairs in parallel. We choose to treat these pairs as independent rather than employ complex solutions used for multi-marginal flow matching (Rohbeck et al., 2025). This means that the models trained in this setup, sample two adjacent time-points for updating. Since the model is conditioned on time, it learns a single map then from μ_0 to μ_{99} formed by connecting all the learned maps between adjacent time points. Additionally since have 500 particles in each measure we sample a batch of 128 particles and compute OT couplings at each time before for training. All methods use the same batch size and all methods use the same mini-batch OT procedure. At inference time we only provide the models with the first time point, and simulate until the next time point, then we repeat this process using the output of the model as the next starting point. We do this until we reach the final time point to produce a prediction.

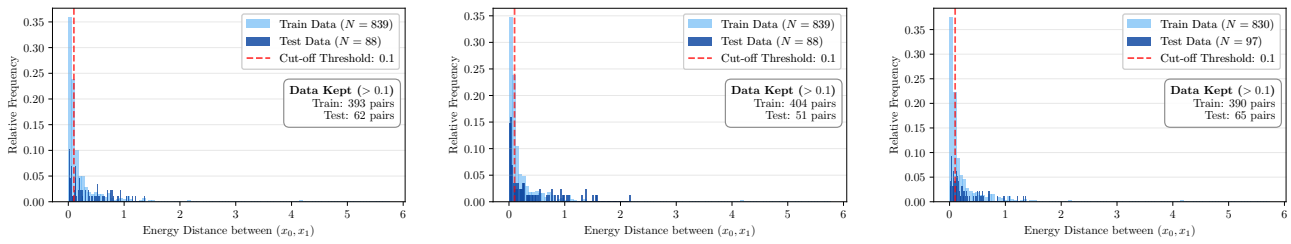
A.3. Patient-derived Organoid (PDO) Dataset

We adopt the data processing framework established in Atanackovic et al. (2025). We refer the reader to Atanackovic et al. (2025) (Appendix D.2) for a more detailed outline of the PDO dataset.

To ensure high-quality training data, we applied additional quality control (QC) filtering. Specifically, we select measure pairs with sufficiently large distributional distance (determined by some threshold). We achieve this by computing ED across all measure pairs in the dataset, and dropping all pairs that don't meet the threshold. We select a threshold of ED = 0.1. We show the empirical distribution of ED across all measures for the respective splits in Figure 4 and the corresponding split statistics.

Lastly, we pair all training cells in each training measure pair per-split pre-training. We do this by first projecting the data features (bio-markers) onto the top 5 principal components of the training data in each split. We then iterate through all measures and acquire the OT pairing between the respective measures.

Models are evaluated throughout training on 9 randomly selected left-out measure pairs from the training set. The 1-Wasserstein metric is tracked every 50 epochs, and the best model checkpoint is selected based on this metric, and likewise used for evaluation on the test set (i.e. left out patient measure pairs).



(a) PDO-21 *(b)* PDO-27 *(c)* PDO-75

Figure 4. Empirical distribution of ED across all measures. Statistics reported for each split.

A.4. Additional Information on Baselines

In this section we offer information about our baseline methods. All models incorporate positional encoding for samples/-particles. For time-dependent models, we use sinusoidal embeddings for time.

Conditional Flow Matching (`CFM`). For our `CFM` baseline, we follow the setup from (Atanackovic et al., 2025). For multo-measure objects experiemnts, we use a densely-connected MLP based architecture. For the McKean-Vlasov and biological experiments, we use ResNet architecture. For the biological experiments (***iii***, Section 5.3), Atanackovic et al. (2025) demonstrated that `OT-CFM` yields competitive performance relative to CellOT (Bunne et al., 2023), an input convex neural network (ICNN)-based approach, on the colorectal cancer treatment-response dataset, which we consider in Section 5.3. Similar to (Bunne et al., 2023), `OT-CFM` provides an optimal transport-based approximation for the M2M operator, and as a result, we use `OT-CFM` as a competitive baseline in the biological setting and do not compare directly to CellOT.

Meta Flow Matching (`Meta-FM`). We follow the setup from (Atanackovic et al., 2025), but forgo the graph convolutional network (GCN) distribution encoder and use a transformer as the encoding architecture instead. The transformer architecture similarly maps a empirical distribution, i.e matrix input, into a single vector output. We use the same vector field model as in `CFM`), with the additional condition on the distribution embedding. We train `Meta-FM` for twice the gradient steps relative to `CFM` as it required alternating updates of two set of parameters. This provides a fair comparison with `CFM`.

Wasserstein Flow Matching (`WFM`). We consider `WFM` as an additional baselines for the multi-measure objects experiments. We use the exact architecture made public in the code released with the original paper in (Haviv et al., 2025). `WFM` operates by first pairing measures (i.e. does not assume *paired* measures) with a *meta*-OT step (OT over measures). We additionally added particle batching within each pair of measures to be consistent with our multi-measure objects experiment setup. We do this to ensure that `WFM` and the other methods train on roughly the same amount of data for a fair comparison. In our experiments we use 16 measures as the batch of measures.

Neural McKean Vlasov (`NMKV`). For the method from Yang et al. (2024), `NKMV`, we use the empirical measure architecture with a modifed training loop to introduce batching of particles (i.e. to match our setting). We set the gradient based learning of sigma to be false, since this is the default in Yang et al. (2024). We do not enable Brownian bridge interpolation since our data does not contain missing/irregular time-points. We set the hidden dim for 'MF model' to 0 because we wish to use the empirical measure architecture which performed well in the original work. We use the implementation of their Mean Field MLP network architecture from the provided codebase (Yang et al., 2024).

Mean Field Transformer (`MF-Transformer`). For the method from Biswal et al. (2025), `MF-Transformer`, we use the publicly released model architecture, and train it to regress towards their proposed finite derivative approximation target. The approximation target in the McKean Vaslov setting corresponds to the following,

$$\text{Target} := \frac{(\mu_{t+1} - \mu_{t-1})}{(2 * \Delta t)}$$

Notably in that work there were many more time-points as such the finite derivative approximation was likely more suited to the setting in that work which would explain why the method struggled in this case.

A.5. Metrics and Distributional Losses

We consider a variety of distributional distances to quantify model performance and to serve as loss functions for our static approaches. Specifically, we consider the 1-Wasserstein and 2-Wasserstein distances, the maximum mean discrepancy (MMD) distance, and energy distance (ED). We additionally consider mean squared error (MSE) as a loss function when used in conjunction with optimal transport couplings.

Maximum Mean Discrepancy (MMD) Distance and Energy Distance (ED). Let $\hat{\mu} = \{x_1, \cdots, x_k\}$ and $\hat{\nu} = \{y_1, \cdots, y_m\}$ denote two empirical distributions with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}^d$. For loss function, we always have $k = m$, while for metrics do not require this constraint (predicted and target distributions may differ in the number of samples). We denote the maximum mean discrepancy distance as

$$\texttt{MMD}(\hat{\mu}, \hat{\nu}) := \frac{1}{k^2}\sum_{i=1}^{k}\sum_{i'=1}^{k} k(x_i, x_{i'}) + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{m} k(y_j, y_{j'}) - \frac{2}{km}\sum_{i=1}^{m}\sum_{j=1}^{m} k(x_i, y_j),$$

where $k(\cdot, \cdot)$ is a kernel function. For what we label as "maximum mean discrepancy (MMD)", we use a Radial Basis Kernel

for k with parameter γ. When using MMD as a loss function (experiments in Section 5.1) we use $\gamma = 0.05$. When using MMD as a metric (in extended results in Section C.1), we compute an average over $\gamma = [2, 1, 0.5, 0.1, 0.01, 0.005]$. Energy distance (ED) then simply follows by consider the kernel $k(x, y) = -\|x - y\|$.

Wasserstein Distance. For $p \geq 1$, we can define the p-Wasserstein distance ($\mathcal{W}_p$) as

$$\mathcal{W}_p(\hat{\mu}, \hat{\nu}) := \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int \|(x - y)\|^p d\gamma(x, y) \right)^{\frac{1}{p}},$$

with Γ denoting the set of couplings between μ and ν, i.e the set of pairings between elements of μ to elements of ν. For model performance comparisons, we evaluate the 1-Wasserstein distance ($\mathcal{W}_1$ for $p = 1$) and the 2-Wasserstein distance ($\mathcal{W}_2$ for $p = 2$).

Mean squared error (MSE). We consider MSE, used in conjunction with mini-batch optimal transport, as an additional loss function. Given given coupled samples $(x, y) \sim \gamma(x, y)$ with $k = m = N$, we denote the `OTMSE` loss as

$$\texttt{MSE}_\gamma(\hat{\mu}, \hat{\nu}) := \frac{1}{N} \sum_{i=1}^{N} \|x_i - y_i\|^2.$$

We do not compute MSE as a metric for model evaluation.

Coefficient of determination (r^2). This metric was considered by Atanackovic et al. (2025) and Bunne et al. (2023) to evaluate method performance between model predicted treatment distributions and target treatment distributions. We use r^2 for our biological single-cell experiments in Section 5.3. In essence, this metric computes a correlation of correlations across between predicted and target empirical distributions. We refer the reader to Atanackovic et al. (2025) for a definition of this metric.

Computation of Losses. To compute distributional losses efficiently, we employ the Geometric Loss Functions package (Feydy et al., 2019). This package provides auto-differentiable versions of these distances enabling efficient and scalable training of our static approaches. Specifically for the p-Wasserstein loss, we approximate $\mathcal{W}_p$ (for $p = 1$ and $p = 2$) via the Sinkhorn algorithm (Cuturi, 2013).

Metrics. We use these corresponding distributional distances as metrics to evaluate the performance of all methods. Since our objective is to the ability of all methods to recover the correct *target* distribution, we do not consider MSE as a metric for evaluation.

A.6. Architecture and Hyperparameters

Architecture. Our architecture is a transformer model which accepts as input an empirical measure represented by a matrix of points. At a high level our transformer consists of an input projection, a sequence of residual blocks, and a final output projection. Our input projection being by augmenting each point with sinusoidal spatial Fourier features (using a learned random matrix with 128 frequencies). This is then feed into an linear input projection layer to the hidden dimension size of 512. Time is encoded and passed into the model with a sinusoidal timestep embedding. Together, this is then fed into a stack of AdaLNBlocks (Perez et al., 2018) blocks, each block uses adaptive layer normalization modulated by time embedding (and any additional covariates/conditions that are used, e.g. treatments in the biological expeirmets), and provides gating. Between each block there is multi-head attention, which acts globally across all samples in the current measure. The final layer consists of a linear projection to the ambient dimension. Our architecture design is akin to transformer architectures used for image generation via diffusion and stochastic interpolants (Peebles & Xie, 2023; Ma et al., 2024).

We use the same architecture for the static M2M methods but omit time conditioning, as it operates strictly as a one-step flow map. We optimize our model with Adam (Kingma & Ba, 2017) using it's default hyperparameters. In Section A.7 and Section A.8 we list the hyperparameters used for our experiments.

Hyperparameter Tuning for multi-measure objects. For the multi-measure objects experiment, we tuned hyperparameters of all methods with a basic grid search over small ranges. For our static and dynamic M2M approaches, we predominately tuned model width, depth, learning rate. We ablated width (number of hidden units = [64, 128, 256, 512]) and depth (number of transformer blocks = [1,2,3,4,5]), and learning rate = [5e-4, 1e-4, 5e-5]. We found performance of our static method predominantly depended on depth (i.e. number of transformer *blocks*), while the dynamic approach was generally invariant.

For CFM, we used the base hyperparamters from (Atanackovic et al., 2025). We also used the base hyperparameters for the vector field in our Meta-FM implementation, but tuned our new self-attention-based distribution encoder using similar depth and width. We used evaluation the held-out "X" silhouettes to tune models, but remark, extensive tuning was not required to find a solid performing set of hyperparameters.

For WFM we initially used the default hyperparameters as given in the provided code from (Haviv et al., 2025), however we observed these hyperparameters led to over-fitting and so we lowered the number of training iterations, reduced the number of layers, and reduced the number of heads. We list these values in Table 18. We found that with this a smaller network WFM performed better in this setting.

Hyperparameter tuning for McKean-Vlasov experiments. We now discuss how we set the hyperparameters for our models, listed in, Section A.7 and for the baseline models, listed in Section A.8. For the McKean-Vlasov experiments we performed a grid search learning rate sweep for all methods on a 100 dimensional kuramoto system using a held out seed, i.e a system not used for evaluating any of the models, and choose the best learning rate to apply to all other systems. We test the following learning rates $[0.00001, 0.0001, 0.00005, 0.0005, 0.009]$, and for Meta-FM we test the cross product of this list for both the encoder and decoder learning rates. For learning rates which lead to divergence when applied to the other systems, we lowered the learning rates until no divergence occurred. For all other hyperparameters we attempt to ensure the models have similar number of parameters and hence we set the number of layers to 5, and the hidden widths to 512. All models which use dropout have it set to 0.1.

Hyperparameter tuning for biological experiments. For biological experiments, similar to letters, we conducted a minimal grid search over model depth, width, learning rate, and number of training epochs. Specifically, for our M2M methods, we ablated width (number of hidden units = [64, 128, 256, 512]) and depth (number of transformer blocks = [3, 5, 7, 9]), and learning rate = [5e-4, 1e-4, 5e-5]. We used a validation set of left out measure pairs within each of the 9 training patients for each respective split to evaluate performance throughout training, and evaluate on this validation set every 50 epochs. We used the best performing checkpoint found throughout training on this validation set as the final model for inference on the test set (completely unseen measures from the left-out patient).

A.7. Hyperparameters used by our approaches

For the multi-measure object experiments in section Section 5.1 M2M-TFM uses the hyperparameters listed in Table 4. For the multi-time-point McKean Vlasov experiments in Section 5.2 M2M-TFM uses the hyperparameters listed in Table 6. For the patient-derived organoid dataset experiments in Section 5.3 M2M-TFM uses the hyperparameters listed in Table 5. For the static maps the same hyperparameters are used in the multi-measure objects and biological (patients) experiments except of course only 1 function evaluation is done at inference. In the Mckean Vlasov experiments the same hyperparameters are used except for learning rates, the complete hyperparameters for the static methods are listed in Table 7, Table 8, and Table 9.

Table 4. Hyperparameters for the M2M-TFM model on multi-measure Object experiments.

Hyperparameter	Value
Input dimension	2
Hidden dimension	64
Number of layers	5
Number of attention heads	8
Learning rate	5×10^{-5}
Dropout	0.05
Time embedding dimension	128
Time-varying dynamics	True
Time steps at inference	100
Measure Batch Size	16
Particle Batch Size	512
OT during Training	True
Epochs (1 training pass over dataset)	1000

Table 5. Hyperparameters for the M2M-TFM model for the Patients experiments.

Hyperparameter	Value
Input dimension	44
Hidden dimension	512
Number of layers	5
Number of attention heads	8
Learning rate	1×10^{-4}
Dropout	0.05
Time steps at inference	100
Measure Batch Size	1
Particle Batch Size	2048
OT during Training	False
Epochs	750

Table 6. Hyperparameters for the `M2M-TFM` transformer model in McKean Vlasov Experiments

Hyperparameter	Value
Input dimension	{2,50,100}
Hidden dimension	512
Number of layers	5
Number of attention heads	4
Learning Rate	1×10^{-5}
Linear LR schedule	start factor 1.0, end factor 0.01
Dropout	0.1
Time embedding dimension	128
Steps between marginals at inference	100
Measure Batch Size	16
Particle Batch Size	128
OT during Training	True
Iterations	100,000

Table 7. Hyperparameters for `M2M-MMD`, `M2M-ED`, `M2M-`$\mathcal{W}_2$, `M2M-`$\mathcal{W}_1$ and `M2M-OTMSE` models on multi-measure object experiments.

Hyperparameter	Value
Input dimension	2
Number of layers	5
Number of attention heads	8
Hidden dimension	64
Learning rate	1×10^{-4}
Dropout	0.05
Time steps at inference	1
Measure Batch Size	16
Particle Batch Size	512
OT during Training	True
Epochs (1 training pass over dataset)	2000

Table 8. Hyperparameters for `M2M-ED`, `M2M-OTMSE` and `M2M-`$\mathcal{W}_1$ in McKean-Vlasov Experiments.

Hyperparameter	Value
Input dimension	{2,50,100}
Learning rate	1×10^{-4}
Number of layers	5
Number of attention heads	4
Hidden dimension	512
Dropout	0.1
Time embedding dimension	128
Training iterations	100,000
Steps per marginal at inference	1
Measure Batch Size	16
Particle Batch Size	128
OT during Training	True

Table 9. Hyperparameters for `M2M-MMD`, `M2M-ED`, `M2M-`$\mathcal{W}_2$, `M2M-`$\mathcal{W}_1$ and `M2M-OTMSE` in the biological experiments.

Hyperparameter	Value
Input dimension	44
Hidden dimension	512
Number of layers	7
Number of attention heads	8
Learning rate	1×10^{-4}
Dropout	0.05
Time steps at inference	1
Measure Batch size	1
Particle Batch Size	2048
OT during Training	False
Epochs	1000

A.8. Baseline Models Hyperparameters

For the multi-measure objects experiments we list the `CFM` model hyperparameters in Table 10. We use the same `Meta-FM` hyperparameters as (Atanackovic et al., 2025), but instead of a graph convolution network we use a transformer population embedding which uses 3 layers, with a width of 64, and 8 heads. The hyperparameters for `Meta-FM` are shown in table Table 11. For `WFM` the hyperparameters used are in Table 18. Note that we used the default hyperparameters plus our batching and we found that the model overfit, as a result we used less gradient steps $(20,000)$ and with less layers (4) which

led to improved performance. For the McKean-Vlasov experiments in Section 5.2 we list hyperparameters in Table 12 for CFM, Table 15 for NMKV, Table 14 for MF-Transformer, and Table 13 for Meta-FM. We note that for CFM, M2M-TFM, and all static transformer methods a linear learning rate schedule was used with a start factor 1.0 and an end factor of 0.01 where each training step the learning rate was decayed until reaching the final factor at the end of training. For the patients experiments in Section 5.3, we list the CFM hyperparameters in Table 16, and the hyperparameter for Meta-FM in Table 17.

Table 10. Hyperparameters for the CFM model in the multi-measure object experiments.

Hyperparameter	Value
Input dimension	2
Hidden dimension	512
Number of layers	4
Skip connections	False
OT during Training	True
Time steps at inference	100
Learning rate	1×10^{-4}
Measure Batch Size	16
Particle Batch Size	512
Epochs (passes over training data)	1000

Table 11. Hyperparameters for Meta-FM model in the multi-measure object experiments.

Hyperparameter	Value
Decoder Width	512
Number of decoder layers	4
Transformer hidden dimension	64
Number of transformer layers	3
Number of attention heads	8
Flow learning rate	1×10^{-4}
Encoder learning rate	1×10^{-4}
OT during Training	True
Steps at Inference	100
Measure Batch Size	1
Particle Batch Size	2048
Epochs (passes over dataset)	2000

Table 12. Hyperparameters for CFM model in McKean-Vlasov experiments.

Hyperparameter	Value
Input dimension	{ 2, 50 ,100 }
Hidden dimension	512
Number of layers	5
Number of attention heads	4
Learning Rate	5×10^{-4}
Dropout	0.1
Time embedding dimension	128
Steps per marginal at inference	100
Measure Batch Size	16
Particle Batch Size	128
OT during Training	True
Iterations	100,000

Table 13. Hyperparameters for Meta-FM (Atanackovic et al., 2025) model in McKean-Vlasov experiments.

Hyperparameter	Value
Input dimension	{ 2, 50 ,100 }
Hidden dimension Decoder	512
Hidden dimension Encoder	512
Number of layers	5
Number of heads	8
Learning Rate Decoder	1×10^{-5}
Learning Rate Encoder	1×10^{-5}
Dropout	0.1
Time embedding dimension	128
Steps per marginal at inference	100
Measure Batch Size	16
Particle Batch Size	128
OT during Training	True
Iterations	200,000

Table 14. Hyperparameters for `MF-Transformer` (Biswal et al., 2025) model in McKean-Vlasov experiment.

Hyperparameter	Value
Input dimension	{ 2, 50 ,100 }
Training iterations (1 gradient step)	200,000
Hidden dimension	512
Number of layers	5
Number of attention heads	4
Learning Rate	1×10^{-5}
Dropout	0.1
Time embedding dimension	128
Steps per marginal at inference	100
Measure Batch Size	16
Particle Batch Size	128
OT during Training	True
Iterations (1 gradient step)	100,000

Table 15. Hyperparameters for `NMKV` (Yang et al., 2024) model in McKean-Vlasov experiments.

Hyperparameter	Value
Input dimension	{ 2, 50 ,100 }
Hidden dimension	512
Number of layers	5
Learning Rate	5×10^{-4}
Particle Batch Size	16
Measure Batch Size	128
OT during Training	True
Model Sigma	1.0
Number of Epochs	100,000

Table 16. Hyperparameters for `CFM` in the biological experiments

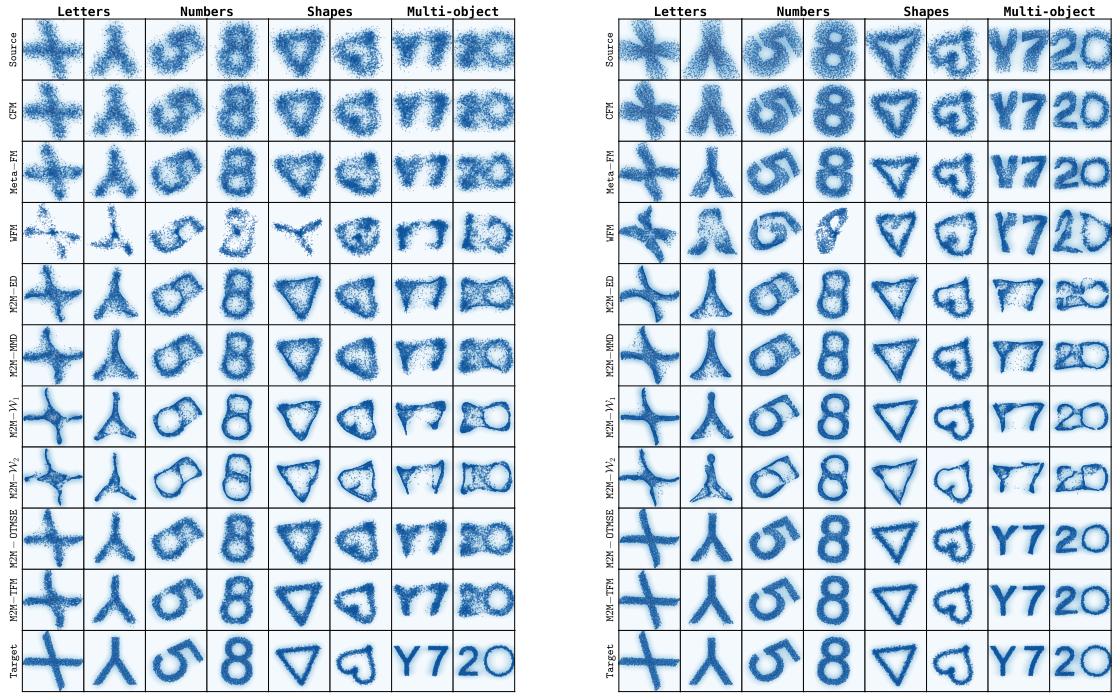
Hyperparameter	Value
Input dimension	44
Width	512
Number of flow layers	7
Learning rate	1×10^{-4}
OT during Training	False
Steps at Inference	100
Measure Batch Size	1
Particle Batch Size	2048
Epochs (passes over the data)	750

Table 17. Hyperparameters for `Meta-FM` model in the biological experiment.

Hyperparameter	Value
Input dimension	44
Decoder hidden dimension	512
Number of decoder layers	7
Transformer hidden dimension	64
Number of transformer layers	3
Number of attention heads	8
Flow learning rate	1×10^{-4}
Encoder learning rate	1×10^{-4}
OT during Training	False
Steps at Inference	100
Batch size	1
Particle Batch Size	2048
Epochs (passes over data)	1500
Validation frequency (epochs)	250

Table 18. Hyperparameters for `WFM` (Haviv et al., 2025) in the multi-measure object experiments.

Hyperparameter	Value
Input dimension	2
Number of transformer layers	4
Number of attention heads	4
Point embedding dimension	256
MLP hidden dimension	256
Dropout rate	0.1
Monge map / OT coupling	Rounded matching
Sinkhorn iterations	200
Entropic OT regularization ε	2×10^{-3}
Data scaling	None
Learning rate (initial)	2×10^{-4}
LR schedule	Exponential decay (every 1000 steps, factor 0.998)
Training iterations (gradient steps)	20,000 (100, 000 found to do worse)
Measure Batch Size	16
Particle Batch Size	512
Steps at inference	100



(a) **Diffusion process** *(b)* **Kernel interaction process**

Figure 5. Extended visualization of model predictions on multi-measure objects for unseen measures.

B. Computational Requirements

We run our experiments using a research compute cluster with L40 Nvidia GPUs (48GB) and H100 GPUs (80 GB). All jobs utilized GPUs, with CPU demand being negligible. All jobs took between $2-24$ hours to complete. The multi-measure objects experiment took 2 to 4 hours on a single L40, using the entire GPU memory. The McKean-Vlasov experiments were ran on $1/4$ of an L40 using NVIDIA Multi-Instance GPU, this means they had access to $1/4$ the compute and $1/4$ the memory of an L40. A single seed used approximately 6 hours of time to complete, with negligible variance between different models. The one exception to this setup was the NMKV model which took 12 hours to train using $1/8$ the compute and $1/8$th the memory of an H100 GPU. Lastly the patients experiments used a single $L40$ and took approximately 8 hours again with small variance between models.

C. Additional Experiments and Results

C.1. Multi-measure Objects

We present extended results for the multi-measure objects experiments in Table 19 and Table 20, as well as extended visuals in Figure 5.

C.2. Mckean-Vlasov Systems

We include a table with metrics for all 9 systems, this includes the $2, 50$, and 100 dimension versions of our Kuramoto, Atlas, and FitzHugh-Nagumo processes. The results for the 2 dimensional systems are listed in Table 21, for the 50 dimensional systems in Table 22, and for 100 dimensional systems in Table 23. In Table 24 we provide results for our ablation over the number of inference steps, i.e number of function evaluations used in inference for `M2M-TFM` on the 50 dimensional systems.

Table 19. Generative performance for **Diffusion** system. Mean ± Std computed over 5 random seeds. **Bold** indicates the best performing method.

	MODEL	$\mathcal{W}_1$ (↓)	$\mathcal{W}_2$ (↓)	MMD (↓)	ED (↓)
LETTERS	CFM	0.2407 ± 0.0022	0.3137 ± 0.0028	0.0032 ± 0.0001	0.0066 ± 0.0003
	Meta−FM	0.2146 ± 0.0170	0.2715 ± 0.0193	0.0027 ± 0.0005	0.0070 ± 0.0019
	WFM	0.6299 ± 0.1538	0.7228 ± 0.1615	0.0424 ± 0.0207	0.1200 ± 0.0563
	M2M−ED	0.1819 ± 0.0116	0.2385 ± 0.0147	0.0020 ± 0.0003	0.0048 ± 0.0006
	M2M−MMD	0.1876 ± 0.0246	0.2386 ± 0.0284	0.0021 ± 0.0005	0.0073 ± 0.0031
	M2M−$\mathcal{W}_1$	0.2027 ± 0.0068	0.2418 ± 0.0123	0.0026 ± 0.0006	0.0081 ± 0.0010
	M2M−$\mathcal{W}_2$	0.2243 ± 0.0201	0.2684 ± 0.0229	0.0033 ± 0.0008	0.0089 ± 0.0013
	M2M−OTMSE	0.1339 ± 0.0054	0.1676 ± 0.0054	0.0010 ± 0.0002	0.0027 ± 0.0005
	M2M−TFM	$\mathbf{0.1118 \pm 0.0035}$	$\mathbf{0.1361 \pm 0.0042}$	$\mathbf{0.0006 \pm 0.0001}$	$\mathbf{0.0017 \pm 0.0001}$
NUMBERS	CFM	0.2060 ± 0.0047	0.2710 ± 0.0056	0.0021 ± 0.0001	0.0045 ± 0.0004
	Meta−FM	0.1895 ± 0.0085	0.2403 ± 0.0106	0.0019 ± 0.0002	0.0051 ± 0.0012
	WFM	0.6478 ± 0.3541	0.7665 ± 0.3824	0.0557 ± 0.0666	0.1561 ± 0.1847
	M2M−ED	0.1714 ± 0.0124	0.2230 ± 0.0153	0.0014 ± 0.0003	0.0038 ± 0.0008
	M2M−MMD	0.1527 ± 0.0084	0.2047 ± 0.0090	0.0010 ± 0.0001	0.0037 ± 0.0012
	M2M−$\mathcal{W}_1$	0.1721 ± 0.0054	0.2152 ± 0.0091	0.0017 ± 0.0003	0.0050 ± 0.0007
	M2M−$\mathcal{W}_2$	0.1971 ± 0.0194	0.2408 ± 0.0225	0.0022 ± 0.0005	0.0067 ± 0.0018
	M2M−OTMSE	0.1376 ± 0.0068	0.1837 ± 0.0071	0.0008 ± 0.0001	0.0022 ± 0.0005
	M2M−TFM	$\mathbf{0.1072 \pm 0.0037}$	$\mathbf{0.1306 \pm 0.0045}$	$\mathbf{0.0004 \pm 0.0000}$	$\mathbf{0.0012 \pm 0.0003}$
SHAPES	CFM	0.2689 ± 0.0061	0.3324 ± 0.0076	0.0041 ± 0.0003	0.0090 ± 0.0009
	Meta−FM	0.2516 ± 0.0286	0.3083 ± 0.0289	0.0041 ± 0.0010	0.0109 ± 0.0039
	WFM	0.5900 ± 0.2258	0.6744 ± 0.2383	0.0570 ± 0.0533	0.1541 ± 0.1361
	M2M−ED	0.2026 ± 0.0072	0.2507 ± 0.0067	0.0026 ± 0.0002	0.0069 ± 0.0007
	M2M−MMD	0.2082 ± 0.0145	0.2578 ± 0.0165	0.0028 ± 0.0004	0.0075 ± 0.0015
	M2M−$\mathcal{W}_1$	0.1951 ± 0.0122	0.2377 ± 0.0126	0.0028 ± 0.0004	0.0082 ± 0.0014
	M2M−$\mathcal{W}_2$	0.2138 ± 0.0261	0.2505 ± 0.0309	0.0035 ± 0.0013	0.0107 ± 0.0046
	M2M−OTMSE	0.1780 ± 0.0020	0.2231 ± 0.0029	0.0019 ± 0.0000	0.0047 ± 0.0002
	M2M−TFM	$\mathbf{0.1286 \pm 0.0041}$	$\mathbf{0.1558 \pm 0.0057}$	$\mathbf{0.0009 \pm 0.0001}$	$\mathbf{0.0024 \pm 0.0001}$
MULTI−OBJECT	CFM	0.2806 ± 0.0018	0.3519 ± 0.0025	0.0035 ± 0.0001	0.0080 ± 0.0002
	Meta−FM	0.2666 ± 0.0046	0.3299 ± 0.0043	0.0040 ± 0.0002	0.0098 ± 0.0005
	WFM	0.4660 ± 0.0571	0.5478 ± 0.0654	0.0157 ± 0.0057	0.0505 ± 0.0199
	M2M−ED	0.2577 ± 0.0078	0.3334 ± 0.0088	0.0036 ± 0.0004	0.0094 ± 0.0010
	M2M−MMD	0.2659 ± 0.0067	0.3489 ± 0.0211	0.0038 ± 0.0001	0.0108 ± 0.0006
	M2M−$\mathcal{W}_1$	0.2670 ± 0.0123	0.3361 ± 0.0148	0.0047 ± 0.0006	0.0127 ± 0.0013
	M2M−$\mathcal{W}_2$	0.2800 ± 0.0199	0.3510 ± 0.0311	0.0049 ± 0.0007	0.0143 ± 0.0027
	M2M−OTMSE	0.2226 ± 0.0062	0.2755 ± 0.0069	0.0026 ± 0.0002	0.0079 ± 0.0009
	M2M−TFM	$\mathbf{0.1724 \pm 0.0022}$	$\mathbf{0.2172 \pm 0.0021}$	$\mathbf{0.0015 \pm 0.0001}$	$\mathbf{0.0038 \pm 0.0003}$
AVERAGE	CFM	0.2491 ± 0.0032	0.3173 ± 0.0044	0.0032 ± 0.0001	0.0071 ± 0.0004
	Meta−FM	0.2306 ± 0.0114	0.2875 ± 0.0131	0.0032 ± 0.0003	0.0082 ± 0.0017
	WFM	0.5834 ± 0.1399	0.6779 ± 0.1493	0.0427 ± 0.0305	0.1201 ± 0.0813
	M2M−ED	0.2034 ± 0.0073	0.2614 ± 0.0081	0.0024 ± 0.0002	0.0062 ± 0.0005
	M2M−MMD	0.2036 ± 0.0094	0.2625 ± 0.0088	0.0024 ± 0.0002	0.0073 ± 0.0012
	M2M−$\mathcal{W}_1$	0.2092 ± 0.0055	0.2577 ± 0.0037	0.0029 ± 0.0001	0.0085 ± 0.0005
	M2M−$\mathcal{W}_2$	0.2288 ± 0.0151	0.2777 ± 0.0189	0.0035 ± 0.0006	0.0102 ± 0.0023
	M2M−OTMSE	0.1680 ± 0.0032	0.2125 ± 0.0034	0.0016 ± 0.0001	0.0044 ± 0.0003
	M2M−TFM	$\mathbf{0.1300 \pm 0.0009}$	$\mathbf{0.1599 \pm 0.0012}$	$\mathbf{0.0009 \pm 0.0000}$	$\mathbf{0.0023 \pm 0.0001}$

Table 20. Generative performance for **Kernel interactions** system. Mean ± Std computed over 5 random seeds. **Bold** indicates the best performing method.

	MODEL	$\mathcal{W}_1$ (↓)	$\mathcal{W}_2$ (↓)	MMD (↓)	ED (↓)
LETTERS	CFM	0.3145 ± 0.0048	0.3630 ± 0.0054	0.0067 ± 0.0002	0.0146 ± 0.0007
	Meta−FM	0.2276 ± 0.0209	0.2716 ± 0.0236	0.0046 ± 0.0009	0.0128 ± 0.0026
	WFM	0.5802 ± 0.1671	0.6877 ± 0.1763	0.0373 ± 0.0242	0.1078 ± 0.0699
	M2M−ED	0.1708 ± 0.0413	0.2237 ± 0.0571	0.0019 ± 0.0008	0.0054 ± 0.0021
	M2M−MMD	0.1525 ± 0.0418	0.2005 ± 0.0610	0.0015 ± 0.0006	0.0051 ± 0.0023
	M2M−$\mathcal{W}_1$	0.1374 ± 0.0306	0.1679 ± 0.0440	0.0011 ± 0.0003	0.0039 ± 0.0012
	M2M−$\mathcal{W}_2$	0.1662 ± 0.0578	0.2049 ± 0.0761	0.0023 ± 0.0023	0.0061 ± 0.0052
	M2M−OTMSE	0.0943 ± 0.0060	0.1096 ± 0.0082	0.0008 ± 0.0002	0.0025 ± 0.0006
	M2M−TFM	$\mathbf{0.0569 \pm 0.0082}$	$\mathbf{0.0652 \pm 0.0089}$	$\mathbf{0.0003 \pm 0.0001}$	$\mathbf{0.0008 \pm 0.0002}$
NUMBERS	CFM	0.2336 ± 0.0025	0.2847 ± 0.0035	0.0036 ± 0.0001	0.0095 ± 0.0004
	Meta−FM	0.1889 ± 0.0079	0.2206 ± 0.0122	0.0025 ± 0.0003	0.0076 ± 0.0014
	WFM	0.5811 ± 0.1631	0.6940 ± 0.1718	0.0443 ± 0.0315	0.1266 ± 0.0875
	M2M−ED	0.1460 ± 0.0158	0.1870 ± 0.0301	0.0012 ± 0.0002	0.0035 ± 0.0004
	M2M−MMD	0.1284 ± 0.0240	0.1666 ± 0.0365	0.0010 ± 0.0003	0.0032 ± 0.0012
	M2M−$\mathcal{W}_1$	0.1163 ± 0.0217	0.1436 ± 0.0292	0.0008 ± 0.0002	0.0026 ± 0.0004
	M2M−$\mathcal{W}_2$	0.1432 ± 0.0427	0.1796 ± 0.0566	0.0013 ± 0.0006	0.0039 ± 0.0017
	M2M−OTMSE	0.0960 ± 0.0152	0.1145 ± 0.0140	0.0007 ± 0.0001	0.0025 ± 0.0007
	M2M−TFM	$\mathbf{0.0509 \pm 0.0052}$	$\mathbf{0.0604 \pm 0.0072}$	$\mathbf{0.0002 \pm 0.0001}$	$\mathbf{0.0006 \pm 0.0001}$
SHAPES	CFM	0.2510 ± 0.0036	0.2944 ± 0.0045	0.0050 ± 0.0002	0.0111 ± 0.0007
	Meta−FM	0.2201 ± 0.0312	0.2556 ± 0.0333	0.0048 ± 0.0011	0.0131 ± 0.0046
	WFM	0.5058 ± 0.1400	0.5663 ± 0.1490	0.0366 ± 0.0339	0.0973 ± 0.0803
	M2M−ED	0.1584 ± 0.0161	0.2004 ± 0.0246	0.0019 ± 0.0002	0.0050 ± 0.0008
	M2M−MMD	0.1555 ± 0.0145	0.1946 ± 0.0197	0.0019 ± 0.0002	0.0052 ± 0.0010
	M2M−$\mathcal{W}_1$	0.1561 ± 0.0173	0.1932 ± 0.0315	0.0019 ± 0.0006	0.0062 ± 0.0015
	M2M−$\mathcal{W}_2$	0.1695 ± 0.0176	0.2077 ± 0.0281	0.0023 ± 0.0004	0.0064 ± 0.0008
	M2M−OTMSE	0.1410 ± 0.0094	0.1702 ± 0.0124	0.0022 ± 0.0004	0.0059 ± 0.0015
	M2M−TFM	$\mathbf{0.0774 \pm 0.0064}$	$\mathbf{0.0907 \pm 0.0082}$	$\mathbf{0.0007 \pm 0.0001}$	$\mathbf{0.0018 \pm 0.0003}$
MULTI−OBJECT	CFM	0.2983 ± 0.0051	0.3436 ± 0.0062	0.0056 ± 0.0002	0.0160 ± 0.0006
	Meta−FM	0.2801 ± 0.0212	0.3280 ± 0.0257	0.0065 ± 0.0009	0.0186 ± 0.0032
	WFM	0.4232 ± 0.0768	0.4864 ± 0.0899	0.0128 ± 0.0045	0.0407 ± 0.0177
	M2M−ED	0.2634 ± 0.0411	0.3299 ± 0.0628	0.0045 ± 0.0011	0.0133 ± 0.0033
	M2M−MMD	0.2535 ± 0.0477	0.3150 ± 0.0602	0.0043 ± 0.0010	0.0136 ± 0.0034
	M2M−$\mathcal{W}_1$	0.2553 ± 0.0366	0.3150 ± 0.0551	0.0047 ± 0.0010	0.0150 ± 0.0023
	M2M−$\mathcal{W}_2$	0.2763 ± 0.0406	0.3430 ± 0.0678	0.0051 ± 0.0017	0.0155 ± 0.0033
	M2M−OTMSE	0.2022 ± 0.0103	0.2253 ± 0.0102	0.0030 ± 0.0001	0.0103 ± 0.0010
	M2M−TFM	$\mathbf{0.1268 \pm 0.0099}$	$\mathbf{0.1455 \pm 0.0128}$	$\mathbf{0.0014 \pm 0.0003}$	$\mathbf{0.0046 \pm 0.0009}$
AVERAGE	CFM	0.2744 ± 0.0030	0.3214 ± 0.0041	0.0052 ± 0.0002	0.0128 ± 0.0005
	Meta−FM	0.2292 ± 0.0150	0.2690 ± 0.0162	0.0046 ± 0.0005	0.0130 ± 0.0022
	WFM	0.5226 ± 0.0545	0.6086 ± 0.0494	0.0328 ± 0.0127	0.0931 ± 0.0303
	M2M−ED	0.1847 ± 0.0283	0.2352 ± 0.0431	0.0024 ± 0.0005	0.0068 ± 0.0014
	M2M−MMD	0.1725 ± 0.0312	0.2192 ± 0.0437	0.0022 ± 0.0005	0.0068 ± 0.0018
	M2M−$\mathcal{W}_1$	0.1663 ± 0.0235	0.2049 ± 0.0351	0.0021 ± 0.0004	0.0069 ± 0.0009
	M2M−$\mathcal{W}_2$	0.1888 ± 0.0380	0.2338 ± 0.0539	0.0027 ± 0.0011	0.0080 ± 0.0025
	M2M−OTMSE	0.1334 ± 0.0088	0.1549 ± 0.0095	0.0017 ± 0.0002	0.0053 ± 0.0009
	M2M−TFM	$\mathbf{0.0780 \pm 0.0020}$	$\mathbf{0.0904 \pm 0.0029}$	$\mathbf{0.0007 \pm 0.0001}$	$\mathbf{0.0020 \pm 0.0002}$

Table 21. 2 Dimensional McKean-Vlasov Systems

	Kuramoto		**FitzHugh-Nagumo**		**Atlas**	
Method	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)
CFM	1.490 ± 0.017	0.995 ± 0.024	1.576 ± 0.035	1.019 ± 0.044	1.370 ± 0.054	0.525 ± 0.044
MF-Transformer	1.357 ± 0.035	2.013 ± 0.071	2.112 ± 0.094	1.400 ± 0.098	26.215 ± 4.963	49.856 ± 9.916
NMKV	0.729 ± 0.079	0.985 ± 0.155	2.383 ± 0.021	2.304 ± 0.040	2.778 ± 0.025	3.261 ± 0.048
MFM	0.581 ± 0.069	0.641 ± 0.127	1.236 ± 0.168	0.728 ± 0.190	2.251 ± 0.293	1.875 ± 0.407
M2M-$\mathcal{W}_1$	$\mathbf{0.334 \pm 0.019}$	$\mathbf{0.415 \pm 0.036}$	$\mathbf{0.543 \pm 0.036}$	0.157 ± 0.026	0.811 ± 0.027	0.421 ± 0.034
M2M-ED	0.552 ± 0.061	0.689 ± 0.128	0.633 ± 0.107	0.274 ± 0.122	1.249 ± 0.115	0.869 ± 0.132
M2M-MSE	0.345 ± 0.037	0.430 ± 0.068	0.623 ± 0.032	0.206 ± 0.026	0.831 ± 0.031	0.476 ± 0.043
M2M-TFM	0.401 ± 0.038	0.518 ± 0.068	0.565 ± 0.027	$\mathbf{0.134 \pm 0.019}$	$\mathbf{0.715 \pm 0.025}$	$\mathbf{0.237 \pm 0.025}$

Table 22. 50 Dimensional McKean-Vlasov Systems

	Kuramoto		**FitzHugh-Nagumo**		**Atlas**	
Method	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)
CFM	7.783 ± 0.120	4.762 ± 0.106	11.476 ± 0.081	1.515 ± 0.051	12.926 ± 0.106	4.585 ± 0.046
MF-Transformer	10.733 ± 0.463	18.197 ± 0.972	15.640 ± 0.439	6.172 ± 0.840	18.752 ± 0.241	21.204 ± 0.472
NMKV	5.482 ± 0.215	7.287 ± 0.422	13.633 ± 0.182	7.455 ± 0.346	23.206 ± 8.421	31.884 ± 16.888
MFM	3.760 ± 0.137	3.658 ± 0.222	13.067 ± 0.219	4.467 ± 0.400	17.426 ± 0.469	11.314 ± 0.794
M2M-$\mathcal{W}_1$	2.709 ± 0.190	3.609 ± 0.378	11.275 ± 0.244	6.352 ± 0.486	14.867 ± 0.172	16.972 ± 0.325
M2M-ED	8.134 ± 0.281	5.968 ± 0.522	16.120 ± 0.423	4.086 ± 0.362	18.863 ± 0.198	8.032 ± 0.204
M2M-MSE	5.953 ± 0.619	10.102 ± 1.238	13.928 ± 0.356	8.741 ± 0.694	14.922 ± 0.177	17.608 ± 0.356
M2M-TFM	$\mathbf{2.196 \pm 0.039}$	$\mathbf{1.791 \pm 0.067}$	$\mathbf{9.671 \pm 0.102}$	$\mathbf{0.692 \pm 0.114}$	$\mathbf{10.996 \pm 0.087}$	$\mathbf{3.863 \pm 0.058}$

Table 23. 100 Dimensional McKean-Vlasov Systems

	Kuramoto		**FitzHugh-Nagumo**		**Atlas**	
Method	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)
CFM	11.312 ± 0.119	6.978 ± 0.135	19.418 ± 0.108	2.435 ± 0.044	19.434 ± 0.115	7.132 ± 0.060
MF-Transformer	13.657 ± 0.422	21.019 ± 0.975	25.044 ± 0.711	8.012 ± 1.000	24.509 ± 0.355	23.359 ± 0.707
NMKV	9.978 ± 0.702	14.130 ± 1.306	20.049 ± 0.320	11.618 ± 0.670	28.830 ± 1.785	30.867 ± 3.830
MFM	7.303 ± 0.204	6.231 ± 0.322	18.683 ± 0.240	5.173 ± 0.177	24.891 ± 0.464	14.720 ± 0.496
M2M-$\mathcal{W}_1$	3.684 ± 0.122	4.786 ± 0.246	15.688 ± 0.136	8.271 ± 0.266	21.838 ± 0.996	26.020 ± 1.946
M2M-ED	13.524 ± 0.403	14.461 ± 2.041	22.113 ± 0.691	4.405 ± 0.791	27.282 ± 0.183	10.783 ± 0.059
M2M-MSE	8.636 ± 0.916	14.709 ± 1.833	18.266 ± 0.481	10.137 ± 0.835	21.329 ± 0.429	25.211 ± 0.855
M2M-TFM	$\mathbf{3.229 \pm 0.055}$	$\mathbf{2.584 \pm 0.095}$	$\mathbf{14.710 \pm 0.113}$	$\mathbf{1.121 \pm 0.158}$	$\mathbf{17.345 \pm 0.059}$	$\mathbf{6.649 \pm 0.096}$

Table 24. Ablation over the number of inference steps per marginal for the 100 dimensional McKean Vlasov systems

	Kuramoto 100D	
Method	$\mathcal{W}_1(\downarrow)$	$ED(\downarrow)$
M2M-TFM 1 steps per marginal	3.219 ± 0.052	2.633 ± 0.087
M2M-TFM 5 steps per marginal	3.224 ± 0.054	2.589 ± 0.093
M2M-TFM 10 steps per marginal	3.226 ± 0.055	2.586 ± 0.094
M2M-TFM 20 steps per marginal	3.228 ± 0.055	2.585 ± 0.095
M2M-TFM 50 steps per marginal	3.228 ± 0.055	2.584 ± 0.095
M2M-TFM 100 steps per marginal	3.229 ± 0.055	2.584 ± 0.095