Measure-to-measure Regression with Transformers

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

[1]University of Alberta, Alberta Machine Intelligence Institute, [2]Canada CIFAR AI Chair
[3]MIT, [4]The Broad Institute of MIT and Harvard

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 measure 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 methods 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 allow 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*

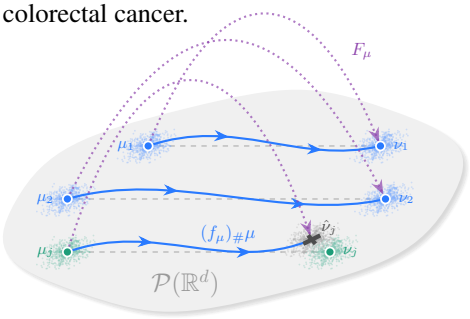


Figure 1: **Illustration of the *measure-to-measure* (M2M) regression problem.** In M2M regression, we treat entire distributions/measures as paired data points (e.g., $(\mu_i, \nu_i) \in \mathcal{P}(\mathbb{R}^d) \times \mathcal{P}(\mathbb{R}^d)$) to learn a continuous mapping between them. Leveraging 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 *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).

†Correspondence to: atanacko@ualberta.ca
https://github.com/Matthew-Vandergrift/m2m-transformers

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

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 and Paul, 2009], faithfully capturing their complex dynamics is a core modeling challenge, and one that calls for computational methods that explicitly account for such distribution-dependent behavior.

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). 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 and 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 and Panaretos, 2022, 2023, Bunne et al., 2023, Girshfeld and 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 and 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 pushforward, $\mu_t = (\Phi_t)_{\#}\mu_0$, 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 and 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.[2]

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 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,[3] $\mu_N := \frac{1}{N}\sum_{j=1}^{N} \delta_{x_j}$. More generally, for any measure μ on $\mathbb{R}^d$ and any $x \in \mathbb{R}^d$ we have,

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

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 $F_{\mu_N}(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$

[2] 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.

[3] 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.

parameterized by the empirical measure μ_N of the input tokens. This perspective has proved fruitful for the mathematical analysis of transformers, beginning 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) $x_1, \ldots, x_N \in \mathbb{R}^d$ which defines 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

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

Notably, the existence and regularity of such transport maps were recently formalized by Lavenant and 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 and Li, 2024, Edelman et al., 2022, Jiang et al., 2023, Wang and E, 2024, Petrov et al., 2024, Sander and 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 $f_\mu = F_\mu$, where F_μ is estimated using a transformer, $F^\theta_\mu(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$. 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.** Recall that we have a dataset of measure pairs $\{(\mu_i, \nu_i)\}_{i=1}^n$. By sampling an index i uniformly at random from $\{1, \ldots, n\}$ and subsequently drawing mini-batches $\tilde{\mu}_i \sim \mu_i$ and $\tilde{\nu}_i \sim \nu_i$, we use stochastic gradient descent to minimize the empirical loss

$$\mathcal{L}_{\text{M2M}}(\theta) = \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{\tilde{\mu}_i\sim\mu_i, \tilde{\nu}_i\sim\nu_i}\Big[\mathbf{D}\Big(\big(F^\theta_{\tilde{\mu}_i}\big)_{\#}\tilde{\mu}_i, \tilde{\nu}_i\Big)\Big]. \tag{3}$$

In practice, any differentiable 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 $\mathcal{W}_1, \mathcal{W}_2$. 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]. This mirrors Genevay et al. [2018], who minimize Sinkhorn divergences to learn a single map from fixed noise to data. In contrast, our static approaches learn measure-dependent pushforward maps amortized across many measure pairs. We label these methods `M2M-MMD`, `M2M-ED`, `M2M-`$\mathcal{W}_1$, and `M2M-`$\mathcal{W}_2$, respectively. We include detailed definitions of the respective distributional distances in Section A.5. We minimize (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 empirical 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, let $\gamma_i \in \Pi(\mu_i, \nu_i)$ denote a chosen coupling between the i-th source and target distributions. For a mini-batch-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 batched measure-level path $\tilde{\mu}_{i,t} = (\phi_t)_\#\gamma_i$. The parameters θ are then optimized by minimizing the following empirical loss:

$$\mathcal{L}_{\mathrm{TFM}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{(x,y)\sim\gamma_i} \int_0^1 \Big\|\chi_t^\theta[\tilde{\mu}_{i,t}](\phi_t(x,y)) - \frac{\partial}{\partial t}\phi_t(x,y)\Big\|^2 dt. \tag{5}$$

Since χ_t^θ takes $\tilde{\mu}_{i,t}$ as input, the measure acts as the condition, so standard guarantees for conditional generation [Dao et al., 2023, Zheng et al., 2023, Isobe et al., 2024] apply to (5). In particular, its minimizer generates the path $\tilde{\mu}_{i,t}$ for every pair, transporting μ_i at $t = 0$ to ν_i at $t = 1$.

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 $\tilde{\mu}_{i,t} = ((1 - t)\phi_{t=0} + t\phi_{t=1})_\sharp \gamma_i$. 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_j by solving the empirical ODE $\frac{d}{dt} z_t = \chi_t^\theta[\mu_t](z_t)$ 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.

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):
    z += Transformer(z, t, θ) * dt
    t += dt
return z # approximates true ν_j
```

We provide pseudo-code for training and inference of the *static* models in Section A.5. We additionally consider an `M2M-OTMSE` variant of the static approach that pairs source and target samples through a mini-batch OT coupling and minimizes $\mathbb{E}_{(x,y)\sim\gamma_i} \|F^\theta_{\tilde{\mu}_i}(x) - y\|^2$. See Section A.6 for details regarding architecture and hyperparameters 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 nonlinear 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 and 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.

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 using 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 related direction uses similar 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 on unseen measures. We describe our individual experimental setups, implementation details, and findings in the following sections.

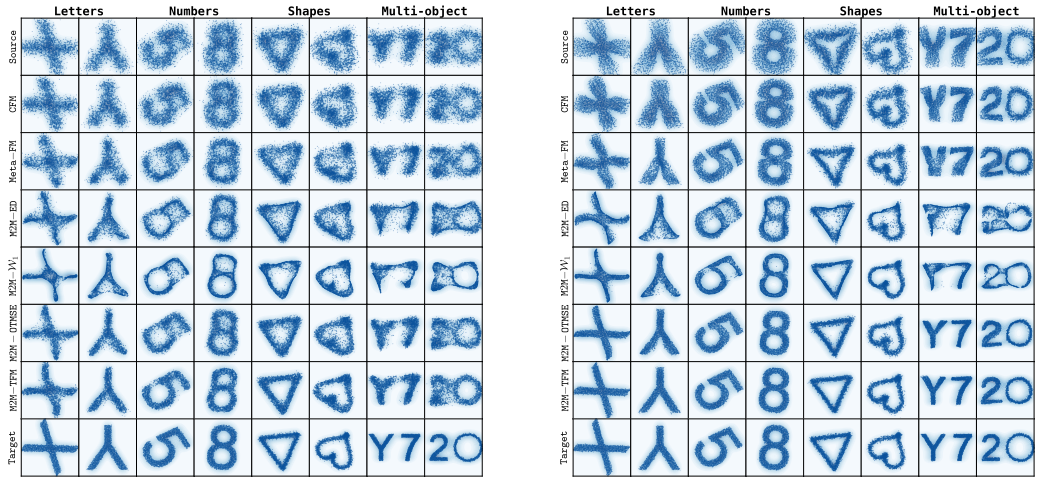
Baselines. We compare our approaches (`M2M-TFM`, `M2M-ED`, `M2M-`$\mathcal{W}_1$) to both non-measure-dependent and measure-dependent methods. We consider Conditional Flow Matching (`CFM`) [Lipman et al., 2022, Tong et al., 2023] as the non-measure-dependent baseline.[4] To deploy `CFM` in the M2M setting, we equivalently marginalize its loss over the training measure pairs and use the same couplings as `M2M-TFM`, with all other experimental conditions matched. Unlike `M2M-TFM`, however, `CFM` parameterizes the vector field with an MLP that acts on individual points, and therefore cannot capture measure-dependence.

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 (**i**, 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 $\mathcal{M} = \{(\mu_i, \nu_i)\}_{i=1}^M$ (with $M = 240$), 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.

[4] 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 measures outside of the training set.** 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 $\pm$ 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	0.2034 ± 0.0073	0.0062 ± 0.0005	0.1847 ± 0.0283	0.0068 ± 0.0014
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	$\mathbf{0.1300 \pm 0.0009}$	$\mathbf{0.0023 \pm 0.0001}$	$\mathbf{0.0780 \pm 0.0020}$	$\mathbf{0.0020 \pm 0.0002}$

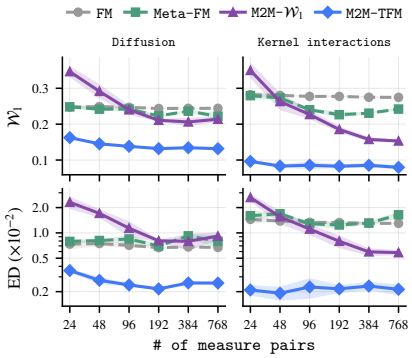


Figure 3: Ablation over number of training measure pairs on multi-measure objects.

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 outperform 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 additionally ablate the number of measure pairs used for training (Figure 3), and observe M2M-TFM outperforms all approaches for every number of measure pairs on both systems, performing well with as few as 24 pairs. All models are trained for a fixed number of epochs across all numbers of measure pairs (24 to 768), and we report the mean and standard deviation of $\mathcal{W}_1$ and ED over 3 seeds. We include extended quantitative results in Table 19 and Table 20, extended visualizations in Figure 6, and further ablations in Section C.1.

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.

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

	Kuramoto		FitzHugh-Nagumo		Atlas	
Method	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)
Baselines						
CFM	11.850 ± 0.085	7.215 ± 0.105	14.929 ± 0.056	0.651 ± 0.008	20.502 ± 0.080	6.729 ± 0.040
MF-Transformer	5.462 ± 0.098	3.315 ± 0.202	23.435 ± 0.254	3.305 ± 0.276	17.175 ± 0.026	8.721 ± 0.026
NMKV	23.441 ± 1.559	27.716 ± 3.117	22.191 ± 0.109	3.376 ± 0.097	21.133 ± 0.077	12.835 ± 0.131
Meta-FM	4.532 ± 0.199	3.258 ± 0.320	14.801 ± 0.175	1.028 ± 0.053	18.656 ± 0.093	7.641 ± 0.042
Static (ours)						
M2M-$\mathcal{W}_1$	5.204 ± 0.436	7.875 ± 0.872	14.757 ± 0.077	7.610 ± 0.163	20.358 ± 0.044	23.132 ± 0.086
M2M-ED	14.400 ± 0.065	17.456 ± 0.326	22.320 ± 0.390	3.653 ± 0.293	32.217 ± 0.268	12.429 ± 0.128
M2M-OTMSE	6.751 ± 1.154	10.964 ± 2.306	17.311 ± 0.297	11.812 ± 0.608	21.407 ± 0.367	25.246 ± 0.738
Dynamic (ours)						
M2M-TFM	$\mathbf{3.005 \pm 0.037}$	$\mathbf{2.299 \pm 0.092}$	$\mathbf{12.687 \pm 0.084}$	$\mathbf{0.548 \pm 0.033}$	$\mathbf{15.940 \pm 0.055}$	$\mathbf{6.633 \pm 0.130}$

Setup. For each McKean-Vlasov system, models are trained on 70 measure trajectories, and are evaluated on 10 held-out trajectories. Each training measure trajectory 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 pairs of 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 (unseen) measure μ_0 and must predict $\hat{\mu}_0, \hat{\mu}_1, \cdots, \hat{\mu}_{99}$, with evaluation metrics averaged over all time-points except the first. We train and evaluate on three modified McKean-Vlasov systems across three dimensional settings: a Kuramoto, Mean-Field Atlas, and FitzHugh-Nagumo system, each in 2-, 50-, and 100-dimensional instantiations. 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. Hyperparameters for all methods are in Section A.6.

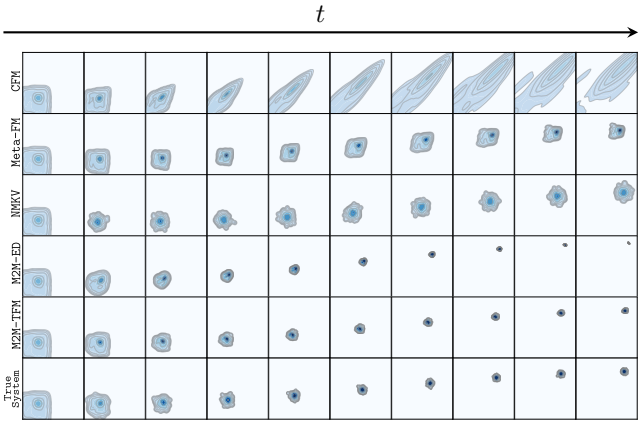


Figure 4: **Visualization of various model predictions on 2 dimensional Kuramoto system.** We sample new initial conditions then repeatedly apply the models, and the true dynamics. For static methods we only show visualization for M2M-ED.

Results. We report results in Table 2, on the 100-dimensional systems; results for all three processes in $2, 50$ and 100 dimensions are in Section C. M2M-TFM outperforms in almost all of the systems, reflecting expressivity and ability to scale more consistently with dimension. NMKV performs worse than one might expect, potentially because it is built for generalizing within a measure, rather than between measures. The Meta-FM baseline often exhibits worse performance, potentially due to dependence only on the initial time-point and requiring optimization of two models. We visualize the model predictions of the 2-dimensional Kuramoto system in Figure 4. In Section C.2, we provide extended results including; an ablation on the number of simulation steps for M2M-TFM, and mean cosine distances between the true and the learned vector fields for a subset of models.

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)[5] 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

[5]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.

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 (coefficient of determination) for each left-out test measure. Further experimental details are in Section A.3.

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. Table 3 has results for the task of predicting patient-specific treatment response for unseen patients. Although each patient has 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 generally 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, including the challenging real-world task of predicting patient-specific treatment response. Together, our novel methods provide practical, scalable, and expressive solutions for M2M regression.

Limitations & Future work. Our methods are currently limited to modeling simplified transport dynamics of physical processes. 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. Promising directions are to extend 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].

Another avenue for future research is to use our static approaches as one-step generative models, in line with recent advances in this direction [Deng et al., 2026, Zhu et al., 2026, Geng et al., 2026, Song and Dhariwal, 2024, Boffi et al., 2024, Potaptchik et al., 2026]. Since our *static* approaches already yield one-step transport maps from arbitrary source distributions, adapting our framework to generative modeling is straightforward. Applying M2M-TFM to higher-dimensional data, such as images, is also promising. Luo et al. [2026] expand the source measure beyond a single Gaussian, a setting that our M2M framework naturally accommodates, while Li and He [2025] show that transformers can be trained directly in pixel space. Lastly, our framework could be extended to other large-scale perturbation screen datasets, such as Zhang et al. [2025] and Huang et al. [2025].

Beyond empirical extensions, addressing certain theoretical questions for *dynamic* M2M regression with transformers remains an important open direction. For instance, future work could study the identifiability of nonlinear M2M operators learned via vector field models under our flow matching framework. 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 partially supported by the Eric and Wendy Schmidt Center at the Broad Institute of MIT and Harvard, and by the NSERC Postdoctoral Fellowship.

References

Abhinav K Adduri, Dhruv Gautam, Beatrice Bevilacqua, Alishba Imran, Rohan Shah, Mohsen Naghipourfar, Noam Teyssier, Rajesh Ilango, Sanjay Nagaraj, Mingze Dong, et al. Predicting cellular responses to perturbation across diverse contexts with state. *BioRxiv*, 2025.

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

Silas Alberti, Niclas Dern, Laura Thesing, and Gitta Kutyniok. Sumformer: Universal approximation for efficient transformers. In *Topological, Algebraic and Geometric Learning Workshops (TAGL)*, 2023.

Brandon Amos, Samuel Cohen, Giulia Luise, and Ievgen Redko. Meta optimal transport. *arXiv preprint arXiv:2206.05262*, 2022.

Erick Armingol, Adam Officer, Olivier Harismendy, and Nathan E Lewis. Deciphering cell–cell interactions and communication from gene expression. *Nature Reviews Genetics*, 2021.

Lazar Atanackovic, Xi Zhang, Brandon Amos, Mathieu Blanchette, Leo J Lee, Yoshua Bengio, Alexander Tong, and Kirill Neklyudov. Meta flow matching: Integrating vector fields on the wasserstein manifold. In *International Conference on Learning Representations*, 2025.

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

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *arXiv preprint arXiv:2406.07507*, 2024.

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

Valérie Castin, Pierre Ablin, and Gabriel Peyré. How smooth is attention? In *International Conference on Machine Learning*, 2024.

Valérie Castin, Pierre Ablin, José Antonio Carrillo, and Gabriel Peyré. A unified perspective on the dynamics of deep transformers. *arXiv preprint arXiv:2501.18322*, 2025.

Shi Chen, Zhengjiang Lin, Yury Polyanskiy, and Philippe Rigollet. Quantitative clustering in mean-field transformer models, 2025a. arXiv:2504.14697.

Shi Chen, Zhengjiang Lin, Yury Polyanskiy, and Philippe Rigollet. Critical attention scaling in long-context transformers. In *International Conference on Learning Representations*, 2026.

Ziang Chen, Yury Polyanskiy, and Philippe Rigollet. 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.

Sinho Chewi, Jonathan Niles-Weed, and Philippe Rigollet. *Statistical optimal transport*, volume 2364 of *Lecture Notes in Mathematics*. Springer, Cham, 2025.

David Chiang, Peter Cholak, and Anand Pillay. Tighter bounds on the expressivity of transformer encoders. In *International Conference on Machine Learning*, 2023.

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

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Atray Dixit, Oren Parnas, Biyu Li, Jenny Chen, Charles P Fulco, Livnat Jerby-Arnon, Nemanja D Marjanovic, Danielle Dionne, Tyler Burks, Raktima Raychowdhury, et al. Perturb-seq: dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. *Cell*, 167(7), 2016.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning*, 2022.

Kilian Fatras, Younes Zine, Szymon Majewski, Rémi Flamary, Rémi Gribonval, and Nicolas Courty. Minibatch optimal transport distances; analysis and applications. *arXiv preprint arXiv:2101.01792*, 2021.

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Nic Fishman, Gokul Gowri, Peng Yin, Jonathan Gootenberg, and Omar Abudayyeh. Generative distribution embeddings. *arXiv preprint arXiv:2505.18150*, 2025.

Nic Fishman, Gokul Gowri, Paolo LB Fischer, Marinka Zitnik, Omar Abudayyeh, and Jonathan Gootenberg. Distribution-conditioned transport. *arXiv preprint arXiv:2603.04736*, 2026.

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

Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International conference on artificial intelligence and statistics*, pages 1608–1617. PMLR, 2018.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Neural Information Processing Systems*, 2026.

Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. The emergence of clusters in self-attention dynamics. In *Neural Information Processing Systems*, 2024a.

Borjan Geshkovski, Philippe Rigollet, and Domènec Ruiz-Balet. Measure-to-measure interpolation using transformers. *arXiv preprint arXiv:2411.04551*, 2024b.

Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. A mathematical perspective on transformers. *Bulletin of the American Mathematical Society*, 62(3), 2025.

Jonathan Geuter, Clément Bonet, Anna Korba, and David Alvarez-Melis. DDEQs: Distributional deep equilibrium models through wasserstein gradient flows. In *International Conference on Artificial Intelligence and Statistics*, 2025a.

Jonathan Geuter, Gregor Kornhardt, Ingimar Tomasson, and Vaios Laschos. Universal neural optimal transport. In *International Conference on Machine Learning*, 2025b.

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

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

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

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

Vincent Guan, Lazar Atanackovic, and Kirill Neklyudov. A call to lagrangian action: Learning population mechanics from temporal snapshots. *arXiv preprint arXiv:2605.08550*, 2026.

Doron Haviv, Aram-Alexandre Pooladian, Dana Pe'er, and Brandon Amos. Wasserstein flow matching: Generative modeling over families of distributions. In *International Conference on Machine Learning*, 2025.

Vincent Tao Hu, Wei Zhang, Meng Tang, Pascal Mettes, Deli Zhao, and Cees Snoek. Latent space editing in transformer-based flow matching. In *AAAI conference on artificial intelligence*, 2024.

Ann C Huang, Tsung-Han S Hsieh, Jiang Zhu, Jackson Michuda, Ashton Teng, Soohong Kim, Elizabeth M Rumsey, Sharon K Lam, Ikenna Anigbogu, Philip Wright, 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.

Noboru Isobe, Masanori Koyama, Jinzhe Zhang, Kohei Hayashi, and Kenji Fukumizu. Extended flow matching: a method of conditional generation with generalized continuity equation. *arXiv preprint arXiv:2402.18839*, 2024.

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

Haotian Jiang, Qianxiao Li, Zhong Li, and Shida Wang. A brief survey on the approximation theory for sequence modelling, 2023. arXiv preprint arXiv:2302.13752.

Bowen Jing, Hannes Stärk, Tommi Jaakkola, and Bonnie Berger. Generative modeling of molecular dynamics trajectories. *Neural Information Processing Systems*, 2024.

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

Anastasis Kratsios, Behnoosh Zamanlooy, Tianlin Liu, and Ivan Dokmanić. Universal approximation under constraints is possible with transformers. In *International Conference on Learning Representations*, 2022.

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

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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

Gaoxiang Luo, Frank Cole, Sihang Zhang, Yuxiang Wan, Yulong Lu, and Ju Sun. Flow matching for multimodal distributions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23260–23271, 2026.

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

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

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

Khai Nguyen, Dang Nguyen, Quoc Nguyen, Tung Pham, Hung Bui, Dinh Phung, Trung Le, and Nhat Ho. On transportation of mini-batches: A hierarchical approach. In *International Conference on Machine Learning*, 2022.

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

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

Stefan Peidli, Tessa D Green, Ciyue Shen, Torsten Gross, Joseph Min, Samuele Garda, Bo Yuan, Linus J Schumacher, Jake P Taylor-King, Debora S Marks, et al. scperturb: harmonized single-cell perturbation data. *Nature Methods*, 2024.

Qiangwei Peng, Zihan Wang, Junda Ying, Yuhao Sun, Qing Nie, Lei Zhang, Tiejun Li, and Peijie Zhou. Wfr-fm: Simulation-free dynamic unbalanced optimal transport. *arXiv preprint arXiv:2601.06810*, 2026.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. 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.

Alexander Petersen and Hans-Georg Müller. Fréchet regression for random objects with euclidean predictors. *The Annals of Statistics*, 2019.

Aleksandar Petrov, Philip HS Torr, and Adel Bibi. Prompting a pretrained transformer can be a universal approximator, 2024. arXiv preprint arXiv:2402.14753.

Katarina Petrović, Lazar Atanackovic, Viggo Moro, Kacper Kapuśniak, Ismail Ilkan Ceylan, Michael M. Bronstein, Joey Bose, and Alexander Tong. Curly flow matching for learning non-gradient field dynamics. In *Neural Information Processing Systems*, 2026.

Peter Potaptchik, Adhi Saravanan, Abbas Mammadov, Alvaro Prat, Michael S Albergo, and Yee Whye Teh. Meta flow maps enable scalable reward alignment. *arXiv preprint arXiv:2601.14430*, 2026.

Santanu Subhash Rathod, Pietro Liò, and Xiao Zhang. Splineflow: Flow matching for dynamical systems with b-spline interpolants. *arXiv preprint arXiv:2601.23072*, 2026.

Martin Rohbeck, Edward De Brouwer, Charlotte Bunne, Jan-Christian Huetter, Anne Biton, Kelvin Y. Chen, Aviv Regev, and Romain Lopez. Modeling complex system dynamics with flow matching across time and conditions. In *International Conference on Learning Representations*, 2025.

Kristiyan Sakalyan, Alessandro Palma, Filippo Guerranti, Fabian J Theis, and Stephan Günnemann. Modeling microenvironment trajectories on spatial transcriptomics with nicheflow. *arXiv preprint arXiv:2511.00977*, 2025.

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

Michael E Sander, Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022.

Nian Si, Jose Blanchet, Soumyadip Ghosh, and Mark Squillante. Quantifying the empirical wasserstein distance to a set of measures: Beating the curse of dimensionality. *Neural Information Processing Systems*, 2020.

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

Jimeng Su, Ying Song, Zhipeng Zhu, Xinyue Huang, Jibiao Fan, Jie Qiao, and Fengbiao Mao. Cell–cell communication: new insights and clinical implications. *Signal transduction and targeted therapy*, 2024.

Zoltán Szabó, Bharath K. Sriperumbudur, Barnabás Póczos, and Arthur Gretton. Learning theory for distribution regression. *Journal of Machine Learning Research*, 2016.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Alexander Y. Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free schrödinger bridges via score and flow matching. In *International Conference on Artificial Intelligence and Statistics*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Neural information processing systems*, 2017.

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

Zhiting Wei, Yiheng Wang, Yicheng Gao, Shuguang Wang, Ping Li, Duanmiao Si, Yuli Gao, Siqi Wu, Danlu Li, Kejing Dong, et al. Benchmarking algorithms for generalizable single-cell perturbation response prediction. *Nature Methods*, 2025.

Haoming Yang, Ali Hasan, Yuting Ng, and Vahid Tarokh. Neural McKean-Vlasov processes: Distributional dependence in diffusion processes. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, 2024.

Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *IEEE/CVF conference on computer vision and pattern recognition*, 2019.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020.

María Ramos Zapatero, Alexander Tong, James W Opzoomer, Rhianna O'Sullivan, Ferran Cardoso Rodriguez, Jahangir Sufi, Petra Vlckova, Callum Nattress, Xiao Qin, Jeroen Claus, et al. Trellis tree-based analysis reveals stromal regulation of patient-derived organoid drug responses. *Cell*, 186(25):5606–5619, 2023.

Jesse Zhang, Airol A Ubas, Richard de Borja, Valentine Svensson, Nicole Thomas, Neha Thakar, Ian Lai, Aidan Winters, Umair Khan, Matthew G Jones, et al. Tahoe-100m: A giga-scale single-cell perturbation atlas for context-dependent gene function and cellular modeling. *BioRxiv*, pages 2025–02, 2025.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *IEEE/CVF international conference on computer vision*, 2021.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky TQ Chen. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023.

Huaisheng Zhu, Teng Xiao, Shijie Zhou, Zhimeng Guo, Hangfan Zhang, Siyuan Xu, and Vasant G Honavar. Simple distillation for one-step diffusion models. In *Neural Information Processing Systems*, 2026.

Aleksandr Zimin, Anastasiia Kutakh, Yury Polyanskiy, and Philippe Rigollet. Learning gaussian mixture models via transformer measure flows. In *ICML 2025 Workshop on Methods and Opportunities at Small Scale*, 2025.

A Additional Experimental Details

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* versions of the McKean-Vlasov Systems from Yang et al. [2024], specifically the Kuramoto model, the Mean-Field Atlas system, and the FitzHugh-Nagumo model each of which we modify. Our goal in modifying these systems is to ensure that given different initial measures the resulting end states of the system are different. This was done because a naive extension of these systems to higher dimensions corresponded to lower diversity in the final states when changing the initial conditions. 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 $X_{j,t}$ for a specific j denotes another particle in the system at time t, i.e $i \neq j$, and that W_i is a Wiener random process, sampled individually for each i. Additionally we use $X_{i,t}^{\dim d}$ to denote the dth dimension of a particle, i.e one of the entries in the D dimensional vector.

We simulate the evolution of the Kuramoto system following equation 6 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 $t = 5.0$.

$$dX_{i,t} = \left(\sin(X_{i,t}) + (2/N)\sum_{j=1}^{N}(\sin(X_{j,t} - X_{i,t}))\right)dt + \sigma * dW_i \tag{6}$$

For the modified FitzHugh-Nagumo system we use a different equation for the first dimension, $X_{i,t}^1$ and the other dimensions $X_{i,t}^2$. Dimension 1 is simulated according to equation 7, and all other dimensions are simulated using equation 8. We simulate starting from 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 $b_i \in [0.5, 0.8], c_i \in [0.5, 1.0], d_i \in [0.3, 0.7]$ and $\tau_i \in [1.0, 10.0]$. We simulate 100 time-points evenly spaced between the initial time and final time. We note that for $D = 2$ we simulate to time 4.0 and for $D > 2$ we simulate until time 10.0.

$$\mathbf{b} = [b_1, \cdots, b_D], b_i = (0.5 + \frac{i-1}{D-1}0.3)) \ \mathbf{c} = [c_1, \cdots, c_D], c_i = (0.5 + \frac{i-1}{D-1}0.5))$$
$$\mathbf{d} = [d_1, \cdots, d_D], d_i = (0.3 + \frac{i-1}{D-1}0.4)) \ \boldsymbol{\tau} = [\tau_1, \cdots, \tau_D], \tau_i = (1.0 + \frac{i-1}{D-1}9.0))$$
$$I(t) = 0.1 \times \sin(10 \times t) \quad \lambda = 0.4, a = 0.2$$
$$dX_{i,t}^{\text{dim1}} = \left(aX_{i,t}^{\text{dim1}}(X_{i,t}^{\text{dim1}} - \lambda)(1 - X_{i,t}^{\text{dim1}}) - \frac{1}{N}\sum_{j=1}^{n} X_{j,t}^{\text{dim}d} + I(t) \right) dt$$
$$- \left(X_{i,t}^{\text{dim1}} - \frac{1}{N}\sum_{j=1}^{N} X_{j,t}^{\text{dim1}} \right) dt \tag{7}$$
$$dX_{i,t}^{\text{dim}d} = \left(\frac{-\mathbf{b}^{\text{dim}d} \times X_{i,t}^{\text{dim}d} + \mathbf{c}^{\text{dim}d} \times X_{i,t}^{\text{dim1}} + \mathbf{d}^{\text{dim}d}}{\boldsymbol{\tau}^{\text{dim}d}} \right) dt \tag{8}$$

For the MeanField Atlas we simulate using equation 9. 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.

$$dX_{i,t}^{\text{dim } d} = \gamma \left(\frac{1}{N}\sum_{j=1}^{N} 1_{[x_{i,t} - Xj,t \geq 0]} \right) * dt + \sigma * dW_{i,t} \tag{9}$$
$$\text{with } \gamma(x) = 5 * (0.5 - x) + (X_{i,t} - 0.01 * X_{i,t}^3) + 1.5 \sin(X_{i,t}^{\text{dim } d-1})$$

Lastly we test on two additional and very simple systems which are easy to visualize and easy to infer the correct behaviour for. The first system we call GoToMean, and it attracts every particle to the mean of the system at each time point. We simulate using the dynamics in Equation (10). We simulate starting from initial conditions consisting of 334 particles drawn from $U_{[-10,10]}$ and 166 particles drawn from $\mathcal{N}(A, 0.1)$ with A sampled from $U_{[-10,10]}$. This is a system without noise. We simulate 100 time points from time $t = 0$ to $t = 10.0$.

$$dX_{i,t} = \left((1/N)\sum_{j=1}^{N} X_{j,t} - X_{i,t} \right) dt \tag{10}$$

The second simple system we call GoToZero attracts every particle to origin at each time point. We simulate using the dynamics in Equation (11). We start initial conditions consisting of 334 particles drawn from $U_{[-1,1]}$ and 166 particles drawn from $\mathcal{N}(A, 0.01)$ with A sampled from $U_{[-1,1]}$. We simulate 100 time points from time $t = 0$ to $t = 10.0$.

$$dX_{i,t} = -X_{i,t}\, dt \tag{11}$$

We provide results for these systems in Section C.2.

A.2.2 Extra Training Details

We shuffle data-points at each time point such that the datasets produced by simulating each of these systems do not have a one to one correspondence between particle i and index i at each time point. 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 measure trajectory 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 from μ_0 to μ_{99} formed by connecting all the learned maps between adjacent time points. Additionally since we have 500

particles in each measure we sample a batch of 128 particles at each training step. 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 predicted trajectory.

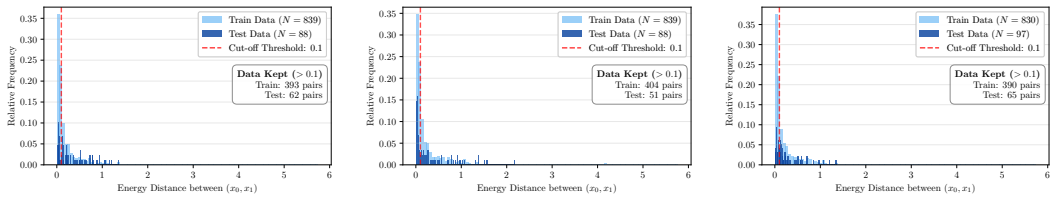
A.3 Patient-derived Organoid (PDO) Dataset

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 5 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 5: 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 multi-measure objects experiments, we use a densely-connected MLP based architecture. For the McKean-Vlasov and biological experiments, we use a 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 an empirical distribution, i.e a matrix input, into a single vector output. For the McKean-Vlasov experiments, we do follow the setup from Atanackovic et al. [2025] and use the graph convolutional network (GCN). 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], `NMKV`, we use the empirical measure architecture with a modified 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 empirical measure and 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 published model, with one key modification. We add a sinusoidal timestep embedding, concatenation to the input, i.e appended to each particle, and passed into the model's embedding layer. The rest of the model is unchanged other than the embedding layer width increasing to handle the time concatenation. We do this because predicting the evolution of multi-time point McKean Vaslov systems requires knowing which of 99 measure pairs the model is currently in. We train this model to regress towards their proposed finite-difference approximation target. The approximation target in the McKean Vlasov setting corresponds to the following,

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

Notably the setup in this work assumes paired particles and we do not use paired particles, nor do we add OT coupling to this method as such it is expected that this method will struggle in this setting.

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. We provide pseudo-code for training and inference of the *static* method in Algorithm 3 and Algorithm 4 respectively.

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 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

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

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 $p = 2$).

Algorithm 3 *Static* M2M (*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, F_μ^θ(·)
x, y = sample_measures(M, b, m)
# tuple ([b, m, d], [b, m, d])
# if using non-distributional loss add:
# x, y = ot_couplings(x, y)
y_pred = Transformer(x, θ)
loss = D(y, y_pred)
θ = update(loss, θ)
```

Algorithm 4 *Static* M2M (*Inference / sampling*)

```
# Inputs:
#   Test Measure: μ_j ∈ R^{N_{μ_j} × d} ,
#   θ: Transformer parameters, F_μ^θ(·)
x = μ_j # [1, N_{μ_j}, d]
y = Transformer(x, θ)
return y # approximates true ν_j
```

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 between predicted and target empirical distributions. We refer the reader to Atanackovic et al. [2025] for a definition of this metric.

Cosine Distance. Given two vectors a, b we compute cosine distance as

$$d_{\cos}(a, b) \;=\; 1 - \frac{\langle a, b \rangle}{\|a\|_2 \, \|b\|_2}.$$

We use this in the McKean–Vlasov systems experiments in which we take the average cosine distance between two measures. Thus we have to apply it to two empirical measures, A and B each consisting of n vectors a_i and b_i. To do this we compute

$$d_{mean\,\cos}(a, b) \;=\; \frac{1}{n} \sum_{i=1}^{n} 1 - \frac{\langle a_i, b_i \rangle}{\|a_i\|_2 \, \|b_i\|_2}.$$

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 of all methods is 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 begins by augmenting each point with sinusoidal spatial Fourier features (using a learned random matrix with 128 frequencies). This is then fed into an linear input projection layer to the hidden dimension size. 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 experiments), 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 and 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 and Ba, 2017] using its 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 evaluated on 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 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 $[0.001, 0.0001, 0.00001, 0.000001]$, and for `Meta-FM` we test the cross product of this list for both the encoder and decoder learning rates. For `Meta-FM` we also tune the number of nearest neighbours $k \in [1, 10, 50]$. 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 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. The hyperparameters used by the static approach

for the multi-measure object experiments in Section 5.1 are listed in table Table 7. For the McKean Vlasov experiments they are listed in Table 8, and for derived organoid dataset experiments they are listed in Table 9. For the McKean Vaslov system experiment, unless otherwise written all methods use a linear learning rate schedule with a start factor 1.0 and an end factor of 0.01.

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**
Hidden dimension	512
Number of layers	5
Number of attention heads	4
Learning Rate	1×10^{-4}
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**
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 hyperparameters for Meta-FM in Table 17. For the McKean Vaslov system experiment, unless otherwise written all methods use a linear learning rate schedule with a start factor 1.0 and an end factor of 0.01.

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
Hidden dimension	512
Number of layers	5
Learning Rate	1×10^{-6}
Dropout	No
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
Hidden dimension Decoder	512
Hidden dimension Encoder	512
Number of layers	5
Number of nearest neighbours	50
Learning Rate Decoder	1×10^{-4}
Learning Rate Encoder	1×10^{-6}
Dropout	No
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
Hidden dimension	512
Number of layers	5
Number of attention heads	4
Learning Rate	1×10^{-6}
Cosine annealing LR schedule	$\eta_{\min} = 10^{-7}$
Dropout	No
Time embedding dimension	128
Steps per marginal at inference	100
Measure Batch Size	16
Particle Batch Size	128
OT during Training	False
Iterations (1 gradient step)	100,000

Table 15: Hyperparameters for `NMKV` [Yang et al., 2024] model in McKean-Vlasov experiments.

Hyperparameter	Value
Hidden dimension	512
Number of layers	5
Learning Rate	1×10^{-5}
Exponential LR schedule	$\gamma = 0.9998$ per step
Particle Batch Size	128
Measure Batch Size	16
OT during Training	False
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

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 run 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. 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 6.

C.2 McKean-Vlasov Systems

We include tables with the metrics for all 9 systems, this includes the $2, 50$, and 100 dimension versions of our modified Kuramoto, Atlas, and FitzHugh-Nagumo systems. The results for the 2 dimensional systems are listed in Section C.2, for the 50 dimensional systems in Section C.2, and for 100 dimensional systems in Section C.2. In Table 26 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 100 dimensional Kuramoto system. Lastly we provide a cosine distance metric between the true vector field and the learned vector field by the model. We provide these metrics for a subset of models on the $2, 50$ and 100 dimensional systems in table 27 , table 28, and table 29. As defined in Section A.5 the cosine reported here is the average cosine distance between all particles in two measures. This is further averaged over all 100 time-points in the system. To compute this we query the model with the true marginal and compare the returned vector field to the true vector field that we use for simulation. This produces 100 mean cosine distances and we take the average over all of them for our final value. Lastly we provide performance metrics for the two simple systems described in Section A.2.1, we provide these metrics in tables Section C.2, Section C.2 and Section C.2. We observe that for the GoToZero system, which does not require measure-dependence, that the non-measure dependent baseline often outperforms or performs on par with M2M-TFM, this is in contrast to the GoToMean system where M2M-TFM and the static M2M methods do generally outperform.



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

Figure 6: Extended visualization of model predictions on multi-measure objects for unseen measures.

Table 19: Prediction 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: Prediction 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: Ablation over the number of samples in each measure pair on the multi-measure objects experiment. Mean ± Std computed over 3 random model seeds and 8 "*object*" silhouettes. **Bold** indicates the best performing method for each number of samples.

	Diffusion		Kernel interactions	
Method	$\mathcal{W}_1$ (↓)	ED (↓)	$\mathcal{W}_1$ (↓)	ED (↓)
512 samples				
CFM	0.2445 ± 0.0011	0.0066 ± 0.0002	0.2774 ± 0.0010	0.0133 ± 0.0005
Meta-FM	0.2449 ± 0.0048	0.0104 ± 0.0003	0.2074 ± 0.0079	0.0094 ± 0.0011
M2M-$\mathcal{W}_1$	0.2015 ± 0.0180	0.0067 ± 0.0014	0.1033 ± 0.0321	0.0025 ± 0.0009
M2M-TFM	$\mathbf{0.1567 \pm 0.0024}$	$\mathbf{0.0038 \pm 0.0000}$	$\mathbf{0.0617 \pm 0.0023}$	$\mathbf{0.0013 \pm 0.0001}$
1024 samples				
CFM	0.2470 ± 0.0020	0.0070 ± 0.0003	0.2771 ± 0.0020	0.0131 ± 0.0004
Meta-FM	0.2246 ± 0.0123	0.0079 ± 0.0014	0.2199 ± 0.0076	0.0119 ± 0.0015
M2M-$\mathcal{W}_1$	0.1836 ± 0.0193	0.0059 ± 0.0016	0.1351 ± 0.0405	0.0050 ± 0.0021
M2M-TFM	$\mathbf{0.1310 \pm 0.0015}$	$\mathbf{0.0023 \pm 0.0001}$	$\mathbf{0.0748 \pm 0.0024}$	$\mathbf{0.0018 \pm 0.0001}$
2048 samples				
CFM	0.2456 ± 0.0009	0.0068 ± 0.0001	0.2746 ± 0.0027	0.0127 ± 0.0005
Meta-FM	0.2254 ± 0.0010	0.0074 ± 0.0004	0.2325 ± 0.0177	0.0132 ± 0.0020
M2M-$\mathcal{W}_1$	0.2030 ± 0.0150	0.0080 ± 0.0016	0.1803 ± 0.0104	0.0078 ± 0.0017
M2M-TFM	$\mathbf{0.1324 \pm 0.0022}$	$\mathbf{0.0022 \pm 0.0001}$	$\mathbf{0.0788 \pm 0.0076}$	$\mathbf{0.0018 \pm 0.0003}$
4096 samples				
CFM	0.2481 ± 0.0015	0.0072 ± 0.0000	0.2755 ± 0.0003	0.0131 ± 0.0002
Meta-FM	0.2218 ± 0.0068	0.0073 ± 0.0010	0.2231 ± 0.0086	0.0114 ± 0.0008
M2M-$\mathcal{W}_1$	0.1917 ± 0.0250	0.0074 ± 0.0022	0.1604 ± 0.0555	0.0068 ± 0.0034
M2M-TFM	$\mathbf{0.1282 \pm 0.0002}$	$\mathbf{0.0020 \pm 0.0000}$	$\mathbf{0.0835 \pm 0.0023}$	$\mathbf{0.0021 \pm 0.0002}$

Table 22: Unbalanced numbers of source and target samples. $\mathcal{W}_1$ (↓) between predicted and target measures for models trained with different numbers of samples in the source and target measures. **Bold** indicates the best performing method for each setting.

# samples		Diffusion				Kernel interactions			
Source	Target	CFM	Meta-FM	M2M-$\mathcal{W}_1$	M2M-TFM	CFM	Meta-FM	M2M-$\mathcal{W}_1$	M2M-TFM
1000	2000	0.2443	0.2295	0.2402	**0.1458**	0.2760	0.2337	0.1939	**0.1068**
	4000	0.2459	0.2693	0.2062	**0.1675**	0.2757	0.2616	0.1763	**0.1267**
	6000	0.2459	0.2366	0.1890	**0.1618**	0.2735	0.2293	0.2180	**0.1375**
3000	4000	0.2454	0.2192	0.1693	**0.1271**	0.2784	0.2383	0.1075	**0.0850**
	6000	0.2453	0.2235	0.2148	**0.1244**	0.2773	0.2311	0.2055	**0.0813**
5000	6000	0.2443	0.2369	0.1718	**0.1247**	0.2723	0.2503	0.1725	**0.0834**

Table 23: **Prediction performance comparison on 2-dimensional McKean-Vlasov systems.** Mean±Std computed over 6 random seeds. **Bold** indicates the best performing method.

	Kuramoto		FitzHugh-Nagumo		Atlas	
Method	$\mathcal{W}_1$	ED	$\mathcal{W}_1$	ED	$\mathcal{W}_1$	ED
CFM	1.454 ± 0.021	0.944 ± 0.024	0.381 ± 0.001	0.122 ± 0.002	1.400 ± 0.012	0.488 ± 0.009
MF-Transformer	1.450 ± 0.251	2.399 ± 0.486	1.291 ± 0.073	0.960 ± 0.117	1.230 ± 0.012	0.612 ± 0.026
NMKV	0.714 ± 0.027	0.952 ± 0.051	1.116 ± 0.002	0.762 ± 0.003	3.067 ± 0.056	3.805 ± 0.112
Meta-FM	0.285 ± 0.016	$\mathbf{0.211 \pm 0.025}$	0.312 ± 0.006	0.080 ± 0.003	0.808 ± 0.054	0.285 ± 0.044
M2M-ED	0.502 ± 0.025	0.716 ± 0.049	0.386 ± 0.033	0.172 ± 0.036	0.892 ± 0.107	0.539 ± 0.133
M2M-OTMSE	0.319 ± 0.029	0.391 ± 0.055	0.344 ± 0.019	0.145 ± 0.019	0.977 ± 0.059	0.685 ± 0.087
M2M-$\mathcal{W}_1$	0.373 ± 0.026	0.482 ± 0.043	0.355 ± 0.030	0.164 ± 0.034	0.900 ± 0.068	0.549 ± 0.083
M2M-TFM	$\mathbf{0.252 \pm 0.022}$	0.269 ± 0.040	$\mathbf{0.288 \pm 0.007}$	$\mathbf{0.070 \pm 0.006}$	$\mathbf{0.678 \pm 0.035}$	$\mathbf{0.225 \pm 0.026}$

Table 24: **Prediction performance comparison on 50-dimensional McKean-Vlasov systems.** Mean±Std computed over 6 random seeds. **Bold** indicates the best performing method.

Method	Kuramoto		FitzHugh-Nagumo		Atlas	
	$\mathcal{W}_1$	ED	$\mathcal{W}_1$	ED	$\mathcal{W}_1$	ED
CFM	8.292 ± 0.056	5.200 ± 0.070	8.885 ± 0.027	0.341 ± 0.008	14.165 ± 0.047	4.514 ± 0.013
MF-Transformer	3.744 ± 0.096	2.262 ± 0.193	14.939 ± 0.145	1.855 ± 0.041	11.502 ± 0.023	5.517 ± 0.032
NMKV	12.721 ± 0.706	15.914 ± 1.393	12.681 ± 0.143	3.338 ± 0.122	14.270 ± 0.038	9.339 ± 0.082
Meta-FM	2.783 ± 0.067	1.770 ± 0.186	8.912 ± 0.053	0.538 ± 0.020	11.793 ± 0.082	4.809 ± 0.119
M2M-ED	8.422 ± 0.244	9.244 ± 0.596	11.783 ± 0.442	1.058 ± 0.252	21.925 ± 0.494	9.000 ± 0.264
M2M-OTMSE	4.504 ± 0.572	7.223 ± 1.143	10.859 ± 0.166	5.282 ± 0.497	14.802 ± 0.225	17.254 ± 0.440
M2M-$\mathcal{W}_1$	3.237 ± 0.304	4.693 ± 0.608	10.251 ± 0.108	5.190 ± 0.178	15.111 ± 0.169	17.841 ± 0.329
M2M-TFM	$\mathbf{1.934 \pm 0.018}$	$\mathbf{1.388 \pm 0.020}$	$\mathbf{8.323 \pm 0.054}$	$\mathbf{0.330 \pm 0.015}$	$\mathbf{10.438 \pm 0.047}$	$\mathbf{4.136 \pm 0.102}$

Table 25: **Prediction performance comparison on 100-dimensional McKean-Vlasov systems.** Mean±Std computed over 6 random seeds. **Bold** indicates the best performing method.

Method	Kuramoto		FitzHugh-Nagumo		Atlas	
	$\mathcal{W}_1$	ED	$\mathcal{W}_1$	ED	$\mathcal{W}_1$	ED
CFM	11.850 ± 0.085	7.215 ± 0.105	14.929 ± 0.056	0.651 ± 0.008	20.502 ± 0.080	6.729 ± 0.040
MF-Transformer	5.462 ± 0.098	3.315 ± 0.202	23.435 ± 0.254	3.305 ± 0.276	17.175 ± 0.026	8.721 ± 0.026
NMKV	23.441 ± 1.559	27.716 ± 3.117	22.191 ± 0.109	3.376 ± 0.097	21.133 ± 0.077	12.835 ± 0.131
Meta-FM	4.532 ± 0.199	3.258 ± 0.320	14.801 ± 0.175	1.028 ± 0.053	18.656 ± 0.093	7.641 ± 0.042
M2M-ED	14.400 ± 0.065	17.456 ± 0.326	22.320 ± 0.390	3.653 ± 0.293	32.217 ± 0.268	12.429 ± 0.128
M2M-OTMSE	6.751 ± 1.154	10.964 ± 2.306	17.311 ± 0.297	11.812 ± 0.608	21.407 ± 0.367	25.246 ± 0.738
M2M-$\mathcal{W}_1$	5.204 ± 0.436	7.875 ± 0.872	14.757 ± 0.077	7.610 ± 0.163	20.358 ± 0.044	23.132 ± 0.086
M2M-TFM	$\mathbf{3.005 \pm 0.037}$	$\mathbf{2.299 \pm 0.092}$	$\mathbf{12.687 \pm 0.084}$	$\mathbf{0.548 \pm 0.033}$	$\mathbf{15.940 \pm 0.055}$	$\mathbf{6.633 \pm 0.130}$

Table 26: Ablation over the number of inference steps per marginal for the 100 dimensional McKean Vlasov systems

Method	Kuramoto 100D	
	$\mathcal{W}_1(\downarrow)$	$ED(\downarrow)$
M2M-TFM (Steps=1)	2.868 ± 0.058	2.630 ± 0.116
M2M-TFM (Steps=5)	2.974 ± 0.034	2.327 ± 0.094
M2M-TFM (Steps=10)	2.992 ± 0.035	2.316 ± 0.092
M2M-TFM (Steps=20)	2.998 ± 0.037	2.305 ± 0.092
M2M-TFM (Steps=50)	3.003 ± 0.037	2.300 ± 0.092
M2M-TFM (Steps=100)	3.005 ± 0.037	2.299 ± 0.092

Table 27: Cosine Distance between true and learned vector field for 2 dimensional systems averaged over 6 seeds.

Method	Atlas	FitzHugh-Nagumo	Kuramoto
NMKV	1.436 ± 0.004	0.493 ± 0.002	0.557 ± 0.010
CFM	0.169 ± 0.002	0.098 ± 0.004	0.208 ± 0.001
Meta-FM	0.100 ± 0.006	0.074 ± 0.005	0.118 ± 0.004
M2M-TFM	0.062 ± 0.001	0.070 ± 0.001	0.084 ± 0.001

Table 28: Cosine Distance between true and learned vector field for 50 dimensional systems averaged over 6 seeds.

Method	Atlas	FitzHugh-Nagumo	Kuramoto
NMKV	1.248 ± 0.003	0.319 ± 0.003	0.528 ± 0.003
CFM	0.438 ± 0.001	0.087 ± 0.001	0.230 ± 0.002
Meta-FM	0.356 ± 0.005	0.102 ± 0.003	0.185 ± 0.003
M2M-TFM	0.241 ± 0.003	0.048 ± 0.002	0.127 ± 0.001

Table 29: Cosine Distance between true and learned vector field for 100 dimensional systems averaged over 6 seeds.

Method	Atlas	FitzHugh-Nagumo	Kuramoto
NMKV	1.199 ± 0.003	0.422 ± 0.002	0.456 ± 0.001
CFM	0.480 ± 0.002	0.124 ± 0.001	0.252 ± 0.002
Meta-FM	0.476 ± 0.002	0.182 ± 0.010	0.215 ± 0.003
M2M-TFM	0.299 ± 0.003	0.071 ± 0.003	0.149 ± 0.001

Table 30: **Prediction performance comparison on two simple 2-dimensional McKean-Vlasov systems (GoToMean and GoToZero).** Mean±Std computed over 3 random seeds. **Bold** indicates the best or tied for best performing method.

	GoToZero		GoToMean	
Method	$\mathcal{W}_1$	ED	$\mathcal{W}_1$	ED
CFM	$\mathbf{0.014 \pm 0.001}$	0.003 ± 0.001	1.782 ± 0.018	1.162 ± 0.016
MF-Transformer	0.127 ± 0.027	0.166 ± 0.050	2.059 ± 0.490	2.350 ± 0.836
NMKV	0.394 ± 0.000	0.218 ± 0.001	1.142 ± 0.099	1.082 ± 0.166
Meta-FM	0.026 ± 0.004	0.022 ± 0.007	0.563 ± 0.052	0.414 ± 0.057
M2M-MSE	$\mathbf{0.014 \pm 0.000}$	$\mathbf{0.002 \pm 0.000}$	0.851 ± 0.165	0.815 ± 0.090
M2M-$\mathcal{W}_1$	0.015 ± 0.002	$\mathbf{0.002 \pm 0.001}$	$\mathbf{0.355 \pm 0.003}$	$\mathbf{0.372 \pm 0.008}$
M2M-TFM	0.016 ± 0.001	0.005 ± 0.001	0.465 ± 0.044	0.555 ± 0.076

Table 31: **Prediction performance comparison on two simple 50-dimensional McKean-Vlasov systems (GoToMean and GoToZero).** Mean±Std computed over 3 random seeds. **Bold** indicates the best performing method.

	GoToZero		GoToMean	
Method	$\mathcal{W}_1$	ED	$\mathcal{W}_1$	ED
CFM	0.242 ± 0.002	$\mathbf{0.014 \pm 0.001}$	10.727 ± 0.086	6.981 ± 0.222
MF-Transformer	2.467 ± 0.015	0.900 ± 0.010	26.676 ± 0.554	11.834 ± 1.191
NMKV	7.441 ± 0.298	4.475 ± 0.225	9.700 ± 0.720	6.919 ± 1.213
Meta-FM	0.287 ± 0.007	0.104 ± 0.011	6.056 ± 0.215	3.518 ± 0.212
M2M-MSE	0.272 ± 0.000	0.116 ± 0.002	12.942 ± 1.270	21.576 ± 2.543
M2M-$\mathcal{W}_1$	$\mathbf{0.238 \pm 0.002}$	0.062 ± 0.006	5.267 ± 0.424	6.273 ± 0.834
M2M-TFM	0.266 ± 0.011	0.043 ± 0.008	$\mathbf{3.804 \pm 0.010}$	$\mathbf{2.149 \pm 0.065}$

Table 32: **Prediction performance comparison on two simple 100-dimensional McKean-Vlasov systems (GoToMean and GoToZero).** Mean±Std computed over 3 random seeds. **Bold** indicates the best performing method.

	GoToZero		GoToMean	
Method	$\mathcal{W}_1$	ED	$\mathcal{W}_1$	ED
CFM	0.548 ± 0.003	$\mathbf{0.070 \pm 0.002}$	21.315 ± 0.241	11.518 ± 0.226
MF-Transformer	3.687 ± 0.034	1.340 ± 0.019	40.543 ± 0.603	19.544 ± 1.521
NMKV	17.954 ± 0.159	11.714 ± 0.145	28.114 ± 0.499	16.447 ± 0.742
Meta-FM	0.417 ± 0.008	0.121 ± 0.013	10.136 ± 0.084	5.920 ± 0.178
M2M-MSE	0.388 ± 0.003	0.161 ± 0.006	19.104 ± 1.470	32.062 ± 2.919
M2M-$\mathcal{W}_1$	$\mathbf{0.348 \pm 0.001}$	0.103 ± 0.002	6.520 ± 0.578	6.899 ± 1.155
M2M-TFM	0.439 ± 0.068	0.085 ± 0.036	$\mathbf{5.535 \pm 0.105}$	$\mathbf{2.761 \pm 0.182}$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide experimental results validating our claims as well as connections to existing theoretical observations.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a thorough limitations and future work in the conclusion section in the main body of the manuscript.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification: Our work does not contribute theoretical results, rather it proposes an set of approaches which are only empirically evaluated.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes],

Justification: We provide detailed descriptions of our experimental setup including the datasets used, and descriptions of how synthetic data was generated. We provide detailed hyperparameter tables, and architectural descriptions of our models.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: For the camera ready version of this work we will link to our public repository containing all code for the experiments in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We explain in our experimental setup our data splits, and provide references for the existing datasets for how they were obtained. We provide hyperparameters and state our optimizer (adam) along with the default hyperparameter values we use for it.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All empirical experiments are conducted over 3 to 6 random seeds and error bars (standard deviation) of performance metrics are reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide an overview of the compute resources used to run our empirical experiments, train models, and reproduce our results in the appendix (in the Computational Requirements section).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, the authors have reviewed and conform to the code of NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work focuses on methodology for M2M regression and its empirical validation; as a result, it does not present immediate foreseeable societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Likewise, since this work focuses on the empirical validation of theoretical methods, it does not pose such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All code and papers used and compared to in this work are cited and referenced.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not have new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve institutional review board (IRB) approvals or equivalent for research with human subjects, nor does it conduct research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.