



Politecnico
di Torino

université
PARIS-SACLAY



MSc thesis in Physics of Complex Systems

Flow Matching for Time-Series Forecasting: Scaling Laws in a Controlled Setting

Elia Parolari, 341901

Academic Year: 2025–2026

Home Institution: Politecnico di Torino

Hosting Institution: Université Paris-Saclay

Hosting Company: Capital Fund Management

Thesis advisor: Alfredo Braunstein (Politecnico di Torino)

External thesis advisor: Leonardo Petrini (CFM)

Contents

1	Introduction	3
1.1	Probabilistic Forecasting	3
1.2	Relevance	3
1.3	State of the Art	3
1.4	Motivation	4
1.4.1	Does probabilistic forecasting learn a better model of the world?	4
1.4.2	Can generative models be trusted for forecasting?	4
1.4.3	Why a controlled, low-dimensional, data-scarce setting?	4
1.4.4	Why this matters in practice	5
1.5	Contributions	5
1.6	Related Works	6
2	Background and Methods	6
2.1	Generative Models	6
2.2	Flow Matching	7
2.2.1	Flows and Flow Models	7
2.2.2	Flows and Velocity Fields	7
2.2.3	Probability Paths and Continuity Equation	7
2.2.4	Flow Matching Loss	8
2.2.5	Conditional Flow Matching Loss	8
2.2.6	Conditional Flows	9
2.2.7	Guided Flow Matching	9
2.2.8	Training	10
2.2.9	Sampling	10
2.2.10	Likelihood Estimation	10
2.3	Probabilistic Forecasting	10
2.3.1	General Framework	10
2.3.2	Markov Assumption	11
2.3.3	Stationarity and Time Homogeneity	11
2.3.4	Flow Matching for Probabilistic Forecasting	11
2.4	Experimental Pipeline	11
2.4.1	Synthetic Data Generation	11
2.4.2	Data Preprocessing	12
2.4.3	Trial-to-Trial Variability	12
2.4.4	Evaluation Metrics	12
3	Simulations	13
3.1	Gaussian Target Distribution	13
3.1.1	Setup	14
3.1.2	Results	14
3.2	AR(1) Process	20
3.2.1	Definitions	20
3.2.2	Distributions of Interests	20
3.2.3	Value Standardization	21
3.2.4	Value-Increments Standardization	22
3.2.5	Setup	22
3.2.6	Results Experiment n.1	23
3.2.7	Results Experiment n.2	27
4	Conclusions	29

References	30
A Algorithms	33
B Gaussian Target Velocity Field	34
C Self Consistency	35

1 Introduction

1.1 Probabilistic Forecasting

Forecasting is the practice of using available information to make informed predictions about future events or quantities. It is widely used in areas such as weather prediction, economics, energy planning, and demand estimation, where anticipating what may happen next supports better decisions today.

A traditional approach is single-point forecasting. In this setting, the focus is on producing one best estimate of the future outcome, such as “tomorrow’s temperature will be 18°C” or “next month’s sales will be 10,000 units.” While simple and easy to interpret, this type of forecast hides uncertainty and does not indicate how reliable the prediction is or what other outcomes are plausible.

Probabilistic forecasting addresses this limitation by describing a range of possible outcomes along with their likelihoods. Instead of a single number, it provides a probability distribution over future values, such as a 70% chance that temperature will fall between 16°C and 20°C, and a 10% chance it will exceed 22°C.

This richer representation is more complete because it communicates not only what is expected, but also how uncertain that expectation is. As a result, probabilistic forecasts enable better risk-aware decisions, allowing users to weigh possible scenarios rather than relying on a single potentially misleading prediction.

1.2 Relevance

Probabilistic forecasting is relevant in any field where decisions depend on uncertain future outcomes and where understanding risk is as important as predicting a single value.

In weather and climate science, it is essential because atmospheric systems are inherently uncertain [3]. Forecasts of temperature, precipitation, storms, or extreme events are more useful when expressed as likelihoods, enabling better preparation for rare but impactful conditions.

In energy systems, probabilistic forecasts help manage fluctuating demand and renewable generation [14]. Operators use probability distributions to plan capacity, storage, and backup resources more reliably.

In finance and economics, uncertainty is central. Probabilistic forecasts [11] are used to support portfolio management and policy decisions by quantifying potential variability rather than assuming a single future trajectory.

In healthcare and epidemiology, it is used to anticipate disease spread [24], hospital demand, and treatment outcomes, where uncertainty can have direct consequences on resource allocation and public health planning.

Across these domains, the key advantage is the same: decisions improve when uncertainty is explicitly modeled rather than ignored.

1.3 State of the Art

The field has advanced dramatically in recent years, largely thanks to progress in deep neural networks. A central role has been played by transformers [30], the same architecture behind large language models like ChatGPT, which have proven remarkably effective not only at generating text, but also at modeling complex temporal and spatial patterns in data. In weather forecasting, for instance, AI-based models like Google DeepMind’s GraphCast [17] now match or outperform traditional physics-based systems, producing global probabilistic forecasts in seconds rather than hours.

More recently, a new family of generative approaches, diffusion models [28] and flow matching [19], has pushed the frontier even further. These are the same techniques behind state-of-the-art image and video generators like Stable Diffusion, Midjourney, and OpenAI’s Sora.

That said, "state of the art" depends heavily on the field and on the metric being used. In some finance applications, robustness and interpretability matter more than raw predictive accuracy, so classical statistical models such as ARIMA, GARCH, and Kalman filters are still widely used alongside newer deep learning methods. [8, 32].

1.4 Motivation

Forecasting is one of the central problems in applied machine learning, and it is increasingly being approached through **generative models**.

This work applies **flow matching** [19], a recent framework for training continuous-time generative models, to synthetic timeseries. Despite the simplicity of this setting, it allows us to address a set of questions that are both conceptually fundamental and practically pressing.

1.4.1 Does probabilistic forecasting learn a better model of the world?

A point predictor only needs to capture a conditional mean (or some other single summary). A generative model, in contrast, must learn the entire conditional distribution of the future given the past. A natural question is whether this richer objective produces a model that is also *better* at point forecasting, when its samples are aggregated into a single estimate. In other words: does forcing the model to "understand" the full distribution improve performance on the simpler task that practitioners most often care about? This is a non-trivial question, since the generative objective is harder and may waste capacity on aspects of the distribution that are irrelevant for point prediction.

1.4.2 Can generative models be trusted for forecasting?

On images and text, the quality of generated samples is largely self-evident: a human can immediately tell whether a picture is realistic or whether a paragraph reads coherently. **Timeseries do not have this luxury**. A generated trajectory may *look* plausible while being statistically wrong: its marginals, autocorrelations, tail behaviour, or conditional distributions may differ substantially from the ground truth. The absence of an intuitive visual check makes the problem of *trust* much more acute. Before such models can be deployed in high-stakes domains, several questions must be answered:

- Can these models at least match the accuracy of dedicated point-estimate models on the point-estimate task?
- Are the probabilities they output **faithful**, in the sense that the learned distribution actually matches the true conditional distribution?
- How can this faithfulness be *tested* in a principled way?

1.4.3 Why a controlled, low-dimensional, data-scarce setting?

It may seem counterintuitive to study flow matching on toy problems when it is known to perform impressively on high-dimensional images and video. The reason is precisely that, in those domains, **faithfulness cannot be measured**: the true data distribution is unknown, so calibration can only be assessed indirectly through proxies (e.g. human evaluation). To genuinely answer the questions above, a setting is needed where:

- the **ground-truth distribution is known**, so that the model's output can be compared to it directly;
- the data is **low-dimensional**, so as to isolate the role of the model and the training procedure from the confounding effects of high-dimensional representation learning;

- the data is **scarce**, in stark contrast to the web-scale corpora on which large generative models are trained.

This regime exposes phenomena that remain invisible at scale, in particular **scaling laws with respect to dataset size** and the **role of noise**, both intrinsic to the data-generating process and injected by the generative model itself.

1.4.4 Why this matters in practice

This setting is not just a theoretical sandbox. In many real-world forecasting problems, and in **finance** in particular, practitioners routinely face exactly these conditions: limited and noisy data, low-dimensional signals, and a need for calibrated uncertainty rather than a single number. The success of generative models on images therefore does not automatically transfer; whether and how it does is itself a research question. Studying flow matching in a clean, ground-truthed environment provides concrete evidence about when probabilistic forecasting is worth the additional complexity, how much its outputs can be trusted, and how its performance scales with the resources available, questions whose answers are essential before such models can be responsibly deployed.

1.5 Contributions

This work does not introduce new flow matching methodology; instead, its contributions lie in providing a careful and reproducible empirical study of flow matching for probabilistic forecasting in a regime that has received relatively little attention in the literature, namely *small datasets*, *low-dimensional data*, and *known ground-truth distributions*. Specifically, the contributions of this report are the following:

- **A controlled benchmarking pipeline.** An experimental pipeline based exclusively on synthetic data with known transition distributions is designed, making it possible to evaluate the learned conditional distribution against ground truth using the Wasserstein-1 distance.
- **Empirical scaling laws.** For both an i.i.d. Gaussian target and an AR(1) process, the scaling of the Wasserstein-1 distance with the training-set size is reported and compared against theoretical predictions. This provides a quantitative benchmark for how close flow matching gets to the best achievable empirical estimator in this regime.
- **Probabilistic vs. point forecasting.** It is shown empirically that, on the AR(1) process, flow matching used as a point predictor (by averaging over sampled trajectories) matches the MSE of an optimal point estimator, while simultaneously learning a distribution that is faithful to the ground truth in Wasserstein sense.
- **Role of Data Preprocessing.** Various standardization and preprocessing techniques are evaluated and their results compared. Particular emphasis is placed on the critical role of preprocessing in flow matching, as the target distribution must ideally exhibit zero mean and unit standard deviation. However, achieving this objective on time series data presents considerable challenges, as naive data standardization can produce degenerate transition distributions. Two solutions are proposed to address this issue: one utilizing a conditional source distribution, and another employing joint prediction of both increments and future values.
- **Methodological rigor in a noisy regime.** Because the setting is small-scale, results are sensitive to seed and hyperparameter choices [6]. Each experiment is therefore systematically averaged over many random seeds, learning-rate and batch-size dependencies

are studied, and the number of seeds needed to estimate the mean Wasserstein distance to a target accuracy is quantified.

The flow matching framework itself, including the conditional flow matching loss, the use of guided velocity fields, and the likelihood formula, is not original to this work and is reviewed in Section 2 for completeness; references to the original papers are given throughout.

1.6 Related Works

Modern neural approaches to probabilistic forecasting include [26], where an autoregressive recurrent neural network is trained across large collections of related time series to learn a global probabilistic forecasting model. In [22], state-space models are used instead, combining latent dynamical structure with neural parameterizations to model temporal uncertainty.

More recent generative approaches extend these ideas using diffusion-based modeling. [23] introduce diffusion models for multivariate time series forecasting, scaling to high-dimensional real-world datasets with strongly correlated components. Building on related ideas, [9] propose stochastic interpolants, defining a continuous-time generative process that connects current observations to future predictive distributions.

Closest to the present work, [10] applies flow matching to multivariate time series forecasting, evaluating performance on both synthetic dynamical systems and real-world forecasting benchmarks.

2 Background and Methods

2.1 Generative Models

The goal of generative modeling is to draw new samples from an unknown distribution, given only a finite dataset of observations from it. Different assumptions about how the data is generated have led to several distinct families of models: Generative Adversarial Networks (GANs) [12], based on a game-theoretic formulation; energy-based models, rooted in statistical physics, such as Boltzmann machines [1]; and autoregressive models, which factorize the joint distribution into a product of conditionals that are directly maximized for likelihood. Transformers [30] are a key architecture for parameterizing these conditionals, though they can also be used to parametrize other generative modeling approaches (one needs to be careful in discerning the choice of the architecture and the paradigm for learning).

A particularly fruitful line of research builds on latent variable models, whose guiding idea is that the observed data is produced by an unobserved process that maps a simple prior, typically Gaussian, to the complex data distribution that one aims at learning. The formal starting tool behind this viewpoint is variational inference, which optimizes a tractable lower bound on the likelihood, the Evidence Lower Bound (ELBO) [5]. Variational Autoencoders (VAEs) [16] were the first deep generative models to fully exploit this framework, but are limited by the restricted expressiveness of their approximate posterior of the latent variable. Normalizing flows [25] address this by composing invertible transformations with tractable Jacobians, and their continuous-time extension, continuous normalizing flows (CNFs) [7], defines the transformation via an ODE¹. CNFs are highly expressive but costly to train, since each step requires solving an ODE, a bottleneck removed by flow matching [18], which enables simulation-free, scalable training.

In parallel, diffusion models [27, 13, 28] take a different route: instead of a deterministic latent-to-data map, they introduce stochasticity through a forward process that corrupts data

¹Unlike the original normalizing flows, which were introduced as flexible variational posteriors within the ELBO framework, CNFs are typically applied directly as generative models of the data distribution, trained by exact maximum likelihood rather than as components of a variational bound.

into noise and a learned reverse process that inverts it. This formulation is closely related to ideas from Langevin dynamics and nonequilibrium statistical mechanics, where stochastic evolution and time-reversal play a central role.

More recently, flows and diffusions have been unified under the framework of stochastic interpolants [2], providing a common language that encompasses and generalizes both.

2.2 Flow Matching

As stated in the previous section, the goal of a generative model is to produce samples from a target distribution q , which is typically unknown or difficult to sample from directly. Flow Matching addresses this by transforming samples $X_0 = x_0 \in \mathbb{R}^d$ from a simple source distribution p into samples $X_1 = x_1 \in \mathbb{R}^d$ from q . The following sections describe how Flow Matching achieves this and how it learns it from data. This is a minimal summary based on [19], to which the reader is referred for a more rigorous and complete treatment.

2.2.1 Flows and Flow Models

Definition 2.1 (Flow). A **flow** is a map $\psi : [0, 1] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ implementing $(t, x) \rightarrow \psi_t(x)$.

Definition 2.2 (Flow Model). A **flow model** is process defined by $X_t = \psi_t(X_0)$ with $X_0 \sim p$.

A flow tells us where an input x will be mapped at time t . The corresponding flow model is a stochastic process constructed by using as input a random variable X_0 from a source distribution p . It can be shown that the resulting process is a continuous time Markov chain.

2.2.2 Flows and Velocity Fields

Definition 2.3 (Velocity Field). A **velocity field** is a map $u : [0, 1] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ implementing $(t, x) \mapsto u_t(x)$.

A flow ψ can be characterized by a velocity field through the following ODE:

$$\begin{cases} \frac{d}{dt}\psi_t(x) = u_t(\psi_t(x)) & \text{(flow ODE)} \\ \psi_0(x) = x & \text{(initial condition)} \end{cases} \quad (1)$$

The flow and velocity field are dual representations of the same geometric object²: given an initial point x , integrating the velocity field forward in time yields the trajectory position $\psi_t(x)$. This complementarity is captured by the fundamental relationship:

$$u_t(x) = \dot{\psi}_t(\psi_t^{-1}(x)) \quad (2)$$

2.2.3 Probability Paths and Continuity Equation

Definition 2.4 (Probability Path). A **probability path** is a time-dependent probability distribution $(p_t)_{t \in [0, 1]}$.

A flow model as in definition 2.2 naturally generates the probability path p_t describing the law of X_t . From a velocity perspective, we say that $u_t(x)$ *generates* p_t if $X_t = \psi_t(X_0) \sim p_t$. The following theorem taken from [31] can be used to check such condition.

Theorem 2.1 (Mass Conservation). Let p_t be a probability path and u_t a locally Lipschitz-integrable vector field. Then, the following two statements are equivalent:

²More rigorously the flow and the velocity should both be regular enough, more specifically they are assumed to be diffeomorphisms. For a more detailed discussion see [19].

1. $u_t(x)$ generates p_t
2. $\frac{\partial p_t}{\partial t}(x) + \nabla \cdot (p_t u_t)(x) = 0$

The condition in point 2 of the theorem above, is known as the **continuity equation**. This fundamental equation describes the conservation of probability mass over time as it is transported by the velocity field $u_t(x)$.

2.2.4 Flow Matching Loss

Flow Matching considers a flow model described by an unknown flow ψ and unknown velocity u whose corresponding probability path p_t is such that:

$$\begin{cases} p_{t=0} = p, & \text{(source distribution)} \\ p_{t=1} = q. & \text{(target distribution)} \end{cases} \quad (3)$$

i.e. such process bridges the source and the target distribution as desired. Since ψ and u are not known, one introduces a parametric version of the velocity u^θ and aims at minimizing the *Flow Matching Loss*³:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{\substack{t \sim \mathcal{U}(0,1) \\ X_t \sim p_t}} \left[\|u_t(X_t) - u_t^\theta(X_t)\|^2 \right] \quad (4)$$

The parametrized velocity obtained can then be used to transport samples from the source distribution to the target distribution.

2.2.5 Conditional Flow Matching Loss

The Flow Matching Loss in eq. (4) is a purely theoretical object that cannot be computed, since u is unknown (otherwise the problem would already be solved). The idea is to work around this issue by introducing a conditioning variable. We start by rewriting p_t using the law of total probability:

$$p_t(x) = \int p_{t|1}(x | x_1) q(x_1) dx_1. \quad (5)$$

At this point we make the *modeling assumption* that, although the full process is unknown, the conditional density $p_{t|1}$ is known; that is, given the endpoint of the process, the distribution at intermediate times is prescribed. Under the independent coupling hypothesis⁴, $p_{t|1}$ must satisfy:

$$\begin{cases} p_{t=0|1}(x | x_1) = p(x), \\ p_{t=1|1}(x | x_1) = \delta(x - x_1), \end{cases} \quad (6)$$

in order for p_t to satisfy eq. (3).

The relationship between the velocity field that *generates* $p_{t|1}$, denoted $u_t(x | x_1)$, and the original velocity field is:

$$u_t(x) = \int u_t(x | x_1) p_{1|t}(x_1 | x) dx_1. \quad (7)$$

Hence, one can interpret the velocity field at a given position x and time t as an average over conditional velocity fields, weighted by the probability of reaching x_1 given the current position x .

³This loss can be written more generally using a generic dissimilarity measure. Every experiment presented later uses the usual l_2 -norm, hence the choice of this specific loss.

⁴More precisely, one introduces the joint distribution of the process at $t = 0$ and $t = 1$, denoted $\pi_{0,1}(x_0, x_1)$ and known as the *coupling*. One then imposes $p_{t=0}(x | x_1) = \pi_{0|1}(x | x_1)$, which reduces to $p(x)$ under the independent coupling hypothesis.

Now one can define the *Conditional Flow Matching Loss*:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{\substack{t \sim \mathcal{U}(0,1) \\ X_1 \sim q \\ X_t \sim p_{t|1}(\cdot | X_1)}} \left[\left\| u_t(X_t | X_1) - u_t^\theta(X_t) \right\|^2 \right] \quad (8)$$

The following theorem justifies why one is interested in this loss.

Theorem 2.2. The gradients of the Flow Matching loss and the Conditional Flow Matching loss coincide:

$$\nabla \mathcal{L}_{\text{FM}}(\theta) = \nabla \mathcal{L}_{\text{CFM}}(\theta). \quad (9)$$

In particular, one can minimize the Conditional Flow Matching loss, and the minimizer will also be a minimizer of the Flow Matching loss. For a proof of the theorem, see [19].

2.2.6 Conditional Flows

In order to compute the Conditional Flow Matching Loss in eq. (8), one needs to be able to compute $u_t(x | x_1)$ generating $p_{t|1}$. Conditional flows offer a principled and computationally efficient way of accessing these quantities.

Consider the conditional flow model: $X_t = \psi_t(X_0 | x_1)$ with $X_0 \sim p$. If $\psi_t(x | x_1)$ satisfies the boundary conditions:

$$\begin{cases} \psi_{t=0}(x | x_1) = x & \text{(identity mapping at } t = 0), \\ \psi_{t=1}(x | x_1) = x_1 & \text{(reaches target at } t = 1), \end{cases} \quad (10)$$

then the induced distribution $p_{t|1}(x | x_1)$ satisfies eq. (6). The corresponding velocity field generating this distribution is:

$$u_t(x | x_1) = \dot{\psi}_t(\psi_t(x | x_1) | x_1), \quad (11)$$

which can be computed explicitly from the flow map. The Conditional Flow Matching Loss then assumes the form:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{\substack{t \sim \mathcal{U}(0,1) \\ X_0 \sim p \\ X_1 \sim q}} \left[\left\| \dot{\psi}_t(X_0 | X_1) - u_t^\theta(X_t) \right\|^2 \right], \quad X_t = \psi_t(X_0 | X_1). \quad (12)$$

This framework is closely related to the *stochastic interpolants* framework proposed by [2], and one can show equivalence under suitable regularity assumptions on the flow.

A natural question that one may have is how to choose a suitable conditional flow for our problem. A common choice⁵ is to choose $\psi_t(x | x_1) = tx + (1-t)x_1$ together with a standard normal source distribution, leading to a conditional distribution $p_{t|1}(x | x_1) = \mathcal{N}(x; tx_1, (1-t)^2)$.

2.2.7 Guided Flow Matching

In most practical applications, one aims to generate samples under specific guidance conditions, such as generating images from textual prompts. Mathematically, this corresponds to learning to sample from a conditional distribution $q(\cdot | Y = y)$, where $y \in \mathcal{Y}$ denotes a set of conditioning labels.

The conditional flow matching framework can be extended to guided generation by assuming that the conditional path remains independent of Y . This yields the Guided Conditional Flow Matching Loss:

⁵This choice minimizes the dynamic Optimal Transport problem with quadratic cost [31].

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{\substack{t \sim \mathcal{U}(0,1) \\ (X_0, X_1, Y) \sim \pi_{0,1,Y}}} \left[\left\| \dot{\psi}_t(X_0 | X_1) - u_t^\theta(X_t | Y) \right\|^2 \right], \quad X_t = \psi_t(X_0 | X_1). \quad (13)$$

The crucial distinction from the unconditional formulation is that the parametrized velocity field u_t^θ now explicitly depends on the guidance signal Y , enabling the model to condition its generation process on the provided label information. Note also that the coupling can be decomposed as $\pi(x_0, x_1, y) = \pi(x_0|y)q(x_1|y)$ with a source distribution that can in practice depend on the guidance. In section 3.2 this will be of crucial importance.

2.2.8 Training

Training is performed by minimizing the loss in eq. (13). Given access to a dataset $\{(x_1^\mu, y^\mu)\}_{\mu=1}^M$ of i.i.d. samples drawn from $X_1, Y \sim \pi_{1,Y}$, and assuming independent coupling between source and target distributions, i.e., $\pi_{0,1,Y} = p \pi_{1,Y}$, the expectation in the loss can be approximated via Monte Carlo sampling:

$$\mathcal{L}_{\text{CFM}}(\theta) \approx \frac{1}{M} \sum_{\mu=1}^M \left\| \dot{\psi}_{t^\mu}(x_0^\mu | x_1^\mu) - u_{t^\mu}^\theta(x_t^\mu | y^\mu) \right\|^2, \quad (14)$$

with:

$$t^\mu \stackrel{\text{i.i.d.}}{\sim} \mathcal{U}(0,1), \quad x_0^\mu \stackrel{\text{i.i.d.}}{\sim} p, \quad x_t^\mu = \psi_{t^\mu}(x_0^\mu | x_1^\mu) \quad (15)$$

The parameters θ are then optimized via (stochastic) gradient descent to minimize this empirical loss approximation. Pseudocode can be found in algorithm 1.

2.2.9 Sampling

Once the velocity field u_t^θ is trained, new samples are generated by numerically integrating the Neural ODE

$$\frac{dx_t}{dt} = u_t^\theta(x_t | y), \quad x_0 \sim p, \quad (16)$$

from $t = 0$ to $t = 1$. The simplest and most common choice is to use Euler integration [20]. Pseudocode can be found in algorithm 2.

2.2.10 Likelihood Estimation

One important benefit of using flows as generative models is that they allow the tractable computation of exact likelihoods through the formula [7]:

$$\log q(x_1) = \log p(x_0) - \int_0^1 \nabla \cdot u_t^\theta(x_t | y) dt, \quad x_t = \psi_t(x_0), \quad (17)$$

In practice, this is evaluated by integrating the Neural ODE *backwards in time* from x_1 at $t = 1$ to x_0 at $t = 0$, while simultaneously accumulating the divergence term along the trajectory. Pseudocode can be found in algorithm 3.

2.3 Probabilistic Forecasting

2.3.1 General Framework

Definition 2.5 (Timeseries). A *discrete-time, continuous-space* timeseries is a sequence of random variables $\mathcal{V} = (V_1, V_2, \dots)$ where each V_t takes value in $\mathbb{R}^d$.

Definition 2.6 (Trajectory). A **trajectory** starting at t_1 and ending at $t_2 > t_1$ is a realization of $t_2 - t_1 + 1$ subsequent random variables $V_{t_1:t_2} = (V_{t_1}, \dots, V_{t_2})$, denoted as $v_{t_1:t_2} = (v_{t_1}, \dots, v_{t_2})$.

The goal of probabilistic forecasting is to characterize the distribution of the future $V_{T+1:T+h}$ given an observed past $V_{1:T} = v_{1:T}$, i.e. to infer the **predictive distribution**

$$p(v_{T+1}, \dots, v_{T+h} \mid v_1, \dots, v_T). \quad (18)$$

2.3.2 Markov Assumption

Definition 2.7 (Markov chain of order p). A timeseries $\mathcal{V}$ is a *Markov chain of order p* if, for every t ,

$$p(v_t \mid v_{t-1}, \dots, v_1) = p(v_t \mid v_{t-1}, \dots, v_{t-p}). \quad (19)$$

In words: conditional on the entire past, the present depends on history only through the p most recent observations. Under this assumption, the joint predictive distribution in eq. (18) factorizes as

$$p(v_{T+1}, \dots, v_{T+h} \mid v_1, \dots, v_T) = \prod_{i=1}^h p(v_{T+i} \mid v_{T+i-1}, \dots, v_{T+i-p}). \quad (20)$$

Therefore, inferring (18) reduces to learning the **transition distribution** (19).

2.3.3 Stationarity and Time Homogeneity

Two additional standard assumptions are made:

- **Time homogeneity:** the transition distribution does not depend on t , only on the p most recent values, i.e.

$$p(v_t \mid v_{t-1}, \dots, v_{t-p}) = p(v \mid u_1, \dots, u_p) \quad \text{for all } t. \quad (21)$$

- **Stationarity:** the joint statistical properties of the process are invariant under time shifts.

These assumptions guarantee that the dynamics *repeat in time*, which is what allows past observations to be informative about the future and, in practice, makes it possible to learn the transition distribution from a single trajectory.

2.3.4 Flow Matching for Probabilistic Forecasting

In the context of probabilistic forecasting, given a timeseries $\mathcal{V}$, flow matching can be applied by choosing as target distribution q the transition distribution in eq. (21), and as dataset the collection of observations $\{(v_t, v_{t-p:t-1})\}_{t=p+1}^T$, where the past p values $v_{t-1:t-p}$ play the role of the guidance variable.

One can also alternatively look at the intervals $\Delta_t = V_t - V_{t-1}$ and train flow matching to sample from the conditional distribution of Δ_t given $V_{t-p:t-1} = v_{t-p:t-1}$ and then use the sampled increments to reconstruct the future values X_t by adding them to the current value.

2.4 Experimental Pipeline

2.4.1 Synthetic Data Generation

All experiments are carried out on **synthetically generated data**. This gives us access to the true data-generating distribution, which is precisely the distribution we aim to learn by sampling from it via flow matching.

Having access to the true distribution allows us to compute **theoretical metrics** that assess the actual quality of the generated samples, for instance by measuring how close the empirical

distribution built from flow-matching samples is to the true one. In addition, a held-out test dataset that is not seen during training can be used to evaluate the model.

For time series, however, this **evaluation is problematic**. The test set is typically obtained by holding out the final portion of a trajectory and treating it as an unseen future. Such a future, though, **does not capture the full underlying process**, as it is only a single possible realization: it might be a typical realization, or an atypical one. For point forecasting this is not a real issue (since in that case the output of the model is a single trajectory too), but for probabilistic forecasting it becomes problematic, since one is forced to assess the quality of the forecasted future (made of multiple sampled trajectories) against a single realization.

Knowing the true process, on the other hand, allows us to genuinely assess the quality of the probabilistic forecast.

Moreover, combining theoretical and validation metrics provides a way to benchmark the latter, by asking: **how well do single-trajectory evaluation metrics capture the quality of a model that learns the full distribution, compared to the ground-truth metrics?**

2.4.2 Data Preprocessing

Flow matching learns to transport samples from a source distribution to a target distribution. To accomplish this, it is crucial that the region of typical samples of the target distribution matches that of the source distribution. In fact, if typical realizations of the target distribution are far from typical realizations of the source, then samples must be transported over long distances, making it difficult to learn a good velocity field. A common way to address this issue is to standardize the data by subtracting the mean and dividing by the standard deviation. This works well for i.i.d. datasets, but it becomes tricky when dealing with time series. Solutions to this problem are proposed and discussed in details in section 3.2.

2.4.3 Trial-to-Trial Variability

As highlighted in [6], neural networks are highly sensitive to sources of variability such as dataset, hyperparameter choices, and weight initialization. Ignoring these factors can lead to biased or misleading conclusions, particularly when comparing different models or architectures. Ideally, results should be averaged over multiple trials that vary the training and evaluation splits, model initialization, and other stochastic components. In practice, however, this is often computationally prohibitive. The experiments presented in this paper are conducted on small datasets and in a small-model parameter regime, which makes it feasible to account for this variability by running each experiment with multiple random seeds, thereby yielding more stable and reliable results.

2.4.4 Evaluation Metrics

As emphasized in [29], different evaluation metrics for generative models can be largely independent of one another: good performance under one criterion does not necessarily imply good performance under another. It is therefore essential to rely on several complementary metrics in order to avoid drawing misleading conclusions. A comprehensive overview of metrics commonly used in probabilistic forecasting can be found in [11].

Theoretical metric: Wasserstein-1 distance. When the true data-generating distribution is available, the Wasserstein-1 distance [21] is used as a primary theoretical benchmark, since it provides a principled way to compare the learned and true distributions. For 1-dimensional distributions the Wasserstein-1 distance between two samples μ_1, μ_2 is defined as:

$$W(\mu_1, \mu_2) = \int_{\mathbb{R}} |F_1(x) - F_2(x)| dx \quad (22)$$

where F_1 and F_2 are the empirical CDFs. A useful result for what follows is the analytical expression for the expected Wasserstein distance between a Gaussian distribution and an empirical distribution drawn from it. In [4], the authors consider $\mathcal{N}(\mu, \sigma^2)$ and the empirical distribution $\hat{P}_N$ built from N i.i.d. samples. Using the CLT, one obtains the scaling

$$\mathbb{E} \left[W \left(\mathcal{N}(\mu, \sigma^2), \hat{P}_N \right) \right] \approx \frac{1.36\sigma}{\sqrt{N}} \quad (23)$$

i.e. it follows a power law with exponent equal to $-1/2$.

Empirical metrics. In practical settings the true distribution is not accessible, and evaluation must be based on a validation trajectory $v_{T+1:T+h}$. For each step we draw samples

$$s_t^{(1)}, \dots, s_t^{(N)} \sim q_\theta(\cdot \mid v_{t-1}, \dots, v_{t-p}), \quad (24)$$

where q_θ denotes the one-step transition distribution induced by integrating the learned velocity field u_θ . Based on these samples we consider the following three metrics, each capturing a different aspect of forecast quality.

1. Mean Squared Error of the predictive mean. This measures the accuracy of the point forecast given by the sample mean $\bar{s}_t = \frac{1}{N} \sum_{j=1}^N s_t^{(j)}$:

$$\text{MSE} = \frac{1}{h} \sum_{i=1}^h \|v_{T+i} - \bar{s}_{T+i}\|^2. \quad (25)$$

If one has access to the process generating the data, it is possible to replace v_{T+i} with the true expectation of the process, yielding an MSE between the sample mean and an oracle point forecaster. This metric will hereafter be referred to as the *oracle MSE*.

2. Average negative log-likelihood (cross-entropy). This evaluates how much probability mass the learned distribution assigns to the observed trajectory:

$$\text{NLL} = -\frac{1}{h} \sum_{i=1}^h \log q_\theta(v_{T+i} \mid v_{T+i-1}, \dots, v_{T+i-p}). \quad (26)$$

3. Average Continuous Ranked Probability Score (CRPS). The CRPS is a proper scoring rule that jointly assesses calibration and sharpness of the predictive distribution. For a univariate predictive CDF F and observation y it is defined as

$$\text{CRPS}(F, y) = \int_{\mathbb{R}} (F(x) - \mathbb{1}\{x \geq y\})^2 dx = \mathbb{E}_{X \sim F}[|X - y|] - \frac{1}{2} \mathbb{E}_{X, X' \sim F}[|X - X'|], \quad (27)$$

where the latter formulation admits a straightforward sample-based estimator. Averaging over the forecast horizon gives

$$\overline{\text{CRPS}} = \frac{1}{h} \sum_{i=1}^h \text{CRPS}(q_\theta(\cdot \mid v_{T+i-1}, \dots, v_{T+i-p}), v_{T+i}). \quad (28)$$

3 Simulations

3.1 Gaussian Target Distribution

This is a preliminary experiment on i.i.d. data with no guidance. The goal is to establish baseline results that can guide the choice of hyperparameters in more complex settings. In each training session the dataset is standardized before feeding it to flow matching. Samples are drawn in the normalized space and then de-normalized. Metrics are computed in the original space.

3.1.1 Setup

The most relevant parameters used in this experiments are reported in table 1.

Parameter	Value
Source distribution	$p = \mathcal{N}(0, 1)$
Target distribution	$q = \mathcal{N}(\mu = 5, \sigma^2 = 1.5^2)$
Dataset size	$N \in \{32, 64, 128, 256, 512, 1024, 2048, 4096\}$
Velocity parametrization	linear, non-linear (MLP, 4 hidden units)
Batch size	$B \in \{32, 64, 128, 256, 512, 1024, 2048, 4096\}$
Learning rate	$\eta \in \{0.1, 0.01, 0.001\}$
Optimizer	SGD
Training iterations	1000
Trials	$n = 100$

Table 1: Experimental setup for the Gaussian target distribution. Parameter values specified as ranges indicate that all combinations were tested independently, each with n different random seeds (number of trials).

3.1.2 Results

Batch size. For both the linear and non-linear models, using larger batch sizes during training helps achieve a lower final Wasserstein distance, as shown in fig. 1. This is consistent with the fact that the loss in eq. (8) is an expectation approximated à la Monte Carlo, and a larger batch size corresponds to a more accurate approximation. On the other hand, smaller batch sizes are usually beneficial during training as they introduce noise that helps escape local minima. In Flow Matching, however, additional noise already arises from the sampling of times and of the source distribution, which may already be sufficient to prevent the optimization from getting stuck. This is a good news from a computational point of view since batches can be processed in parallel.

Learning rate and number of iterations. The dynamics of the loss and of the Wasserstein distance are reported in fig. 2 for the linear model and in fig. 3 for the non-linear model with 4 hidden units, under a fixed number of iterations and varying learning rates. For the linear model, the final value of the loss is essentially the same across all learning rates and training-set sizes; larger training-set sizes correspond to less noisy loss curves, i.e. a more stable training. The dynamics of the Wasserstein distance are also similar across learning rates. A good choice is $\eta = 0.01$, which is large enough to avoid stopping too early, as instead happens in fig. 2a with $\eta = 0.001$.

For the non-linear model, whose results are shown in fig. 3, a learning rate of $\eta = 0.1$ unlocks a lower region of the loss. This region, however, appears to correspond to an overfitting regime: looking at the corresponding Wasserstein dynamics, once the loss starts decreasing after the initial plateau, the Wasserstein distance starts to increase, especially for small training-set sizes. This indicates that care must be taken when introducing non-linearity in the velocity field, and that having access to a reliable evaluation metric, to be used, for instance, for early stopping, may be of crucial importance. For the non-linear model as well, $\eta = 0.01$ appears to be a sweet spot.

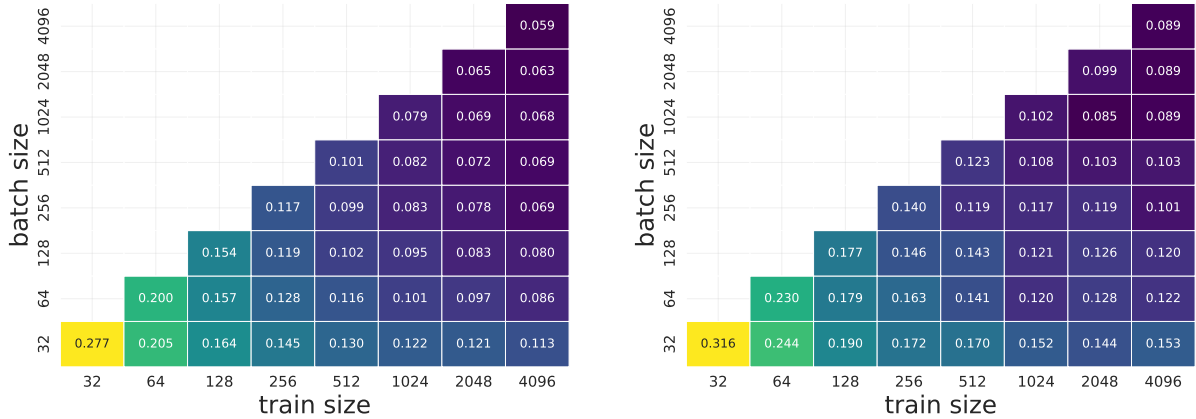
Additional plots are reported in section B, where the velocity field itself is visualized during training. The linear model learns an essentially uniform, zero velocity field: indeed, for this simple problem, once the data is standardized the source already matches the target, so the

optimal velocity is trivial. In contrast, the non-linear model learns a non-uniform field that tends to spread samples apart. This behaviour is consistent with the other results reported below.

Scaling of the Wasserstein distance with training-set size. Figure 4 shows the Wasserstein distance at the final iteration (i.e. once training is complete) as a function of the training-set size in log-log space, at the fixed (best) learning rate $\eta = 0.01$. The theoretical scaling (see section 2.4), is also reported as a reference. Both the linear and the non-linear models achieve a good score, with the linear model slightly outperforming the non-linear one in this simple setting.

Mean and standard deviation dynamics. The information content of a Gaussian is fully captured by its mean and standard deviation. It is therefore informative to compare the true mean and standard deviation against: (1) the empirical mean and standard deviation of the training dataset, which serve as a proxy for the best achievable score, and (2) the mean and standard deviation of samples generated via Flow Matching. This comparison is shown in fig. 5 in log-log space at the best learning rate. While the mean is learned equally well by the linear and the non-linear models, the non-linear model appears to struggle slightly more in matching the correct standard deviation. This behavior, as mentioned earlier, can be explained as a consequence of overfitting that leads to a velocity field pushing samples away from the origin as confirmed by the plots in section B.

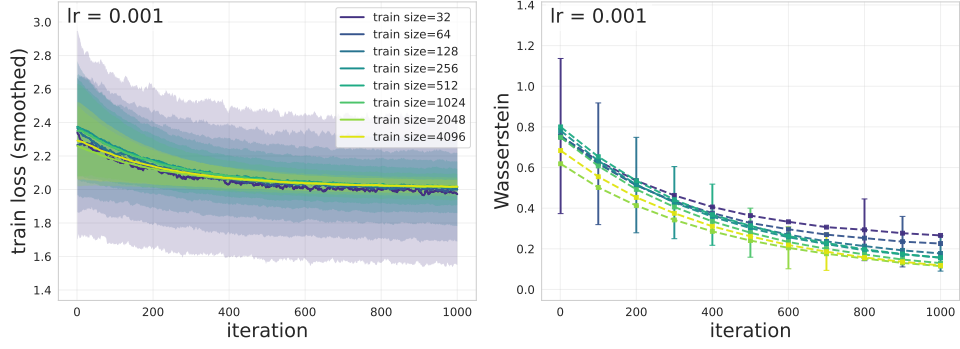
Error on the mean Wasserstein. Denote with w_N^i the Wasserstein distance computed for a model trained with N data points at the i -th trial. The error in estimating $\bar{w}_N$ scales with the number of trials n as $\frac{\sigma_N}{\sqrt{n}}$. A target error r can be achieved using $n(r) = (\frac{\sigma_N}{r})^2$ trials. The curves of this quantity are presented in fig. 6. In particular note that in order to increase the accuracy of one order of magnitude (e.g. from 0.01 to 0.001) one has to increase the number of trials by two order of magnitudes.



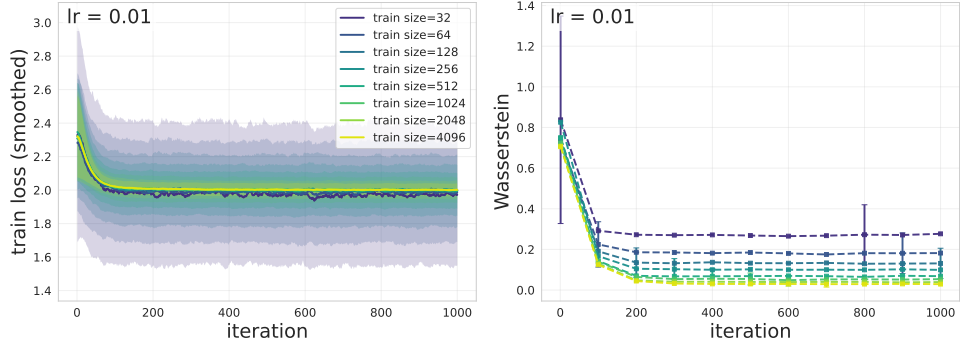
(a) Linear velocity field.

(b) MLP (4 hidden units).

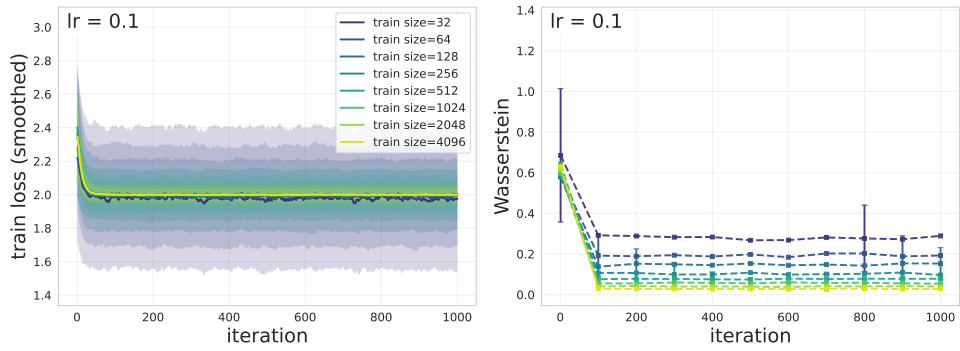
Figure 1: Heatmap of the Wasserstein distance between samples from the true target distribution and the distribution learned with Flow Matching; lower (darker) values are better. The horizontal axis reports the dataset size used for training, and the vertical axis the batch size. The metric is averaged over the learning rates used and the random seeds. Figure 1a uses a linear parametrization of the velocity, while Figure 1b uses a non-linear parametrization with 4 hidden units. Both figures show that a larger batch size achieves a lower Wasserstein distance.



(a) Linear velocity field, $\eta = 0.001$.

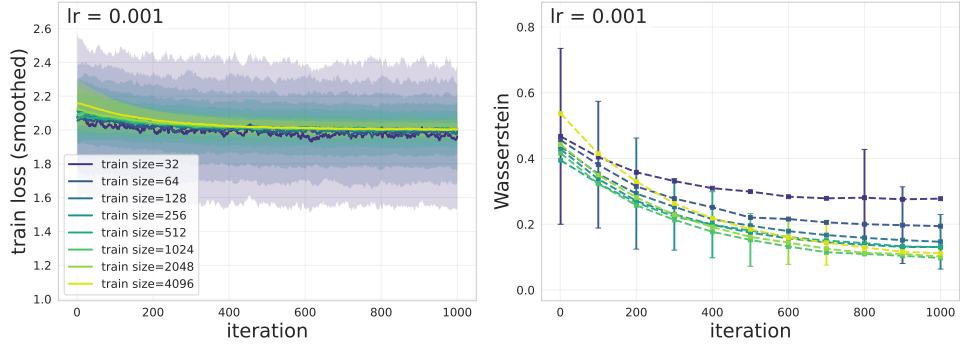


(b) Linear velocity field, $\eta = 0.01$.

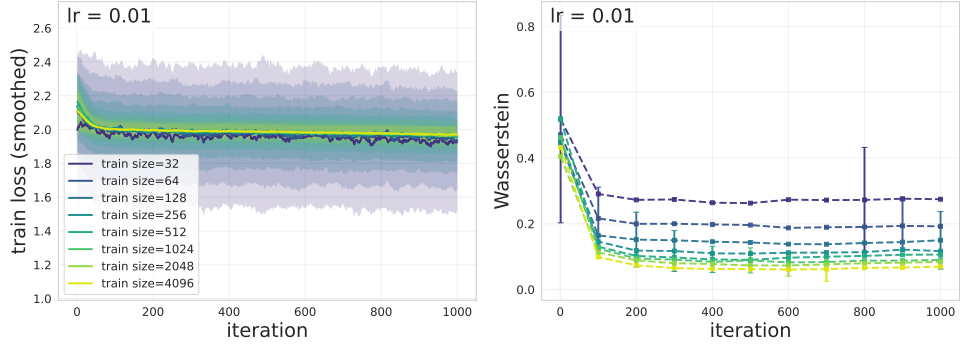


(c) Linear velocity field, $\eta = 0.1$.

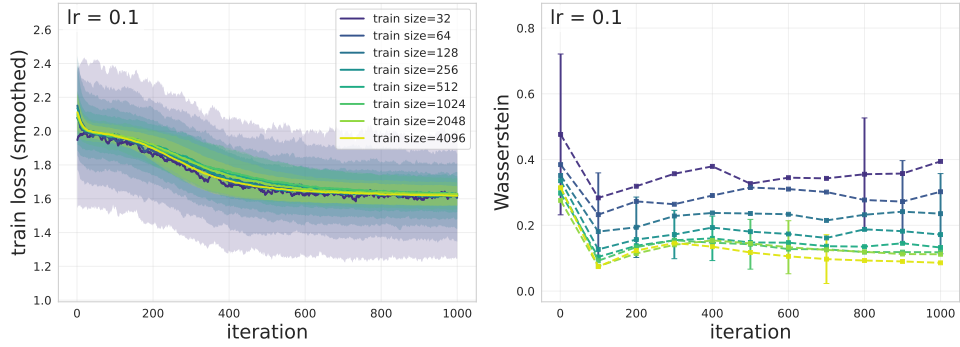
Figure 2: Training loss (left) and Wasserstein distance (right) versus iteration for the **linear** velocity parametrization, at three learning rates and several training-set sizes (largest available batch size used in each case). Shaded regions and error bars denote ± 1 standard deviation across random seeds. The final Wasserstein distance is approximately identical across all learning rates; what changes is the number of iterations needed to reach a stable plateau.



(a) MLP (4 hidden units), $\eta = 0.001$.

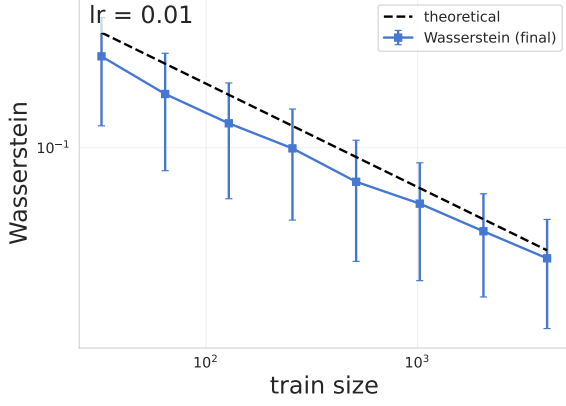


(b) MLP (4 hidden units), $\eta = 0.01$.

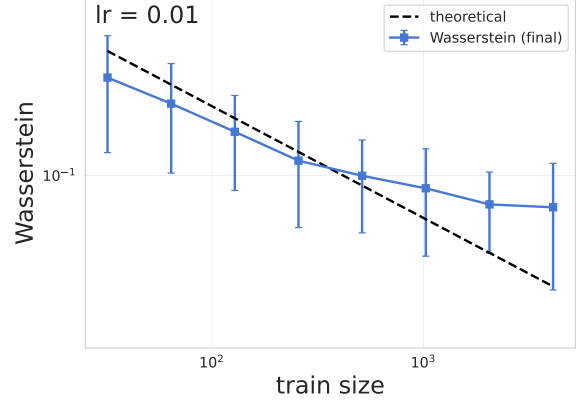


(c) MLP (4 hidden units), $\eta = 0.1$.

Figure 3: Training loss (left) and Wasserstein distance (right) versus iteration for the **non-linear** velocity parametrization, at three learning rates and several training-set sizes (largest available batch size used in each case). Shaded regions and error bars denote ± 1 standard deviation across random seeds. For $\eta = 0.01$, the loss reaches a lower plateau while the Wasserstein distance starts increasing. This is a typical sign of overfitting, which was not observed with the simpler linear model.

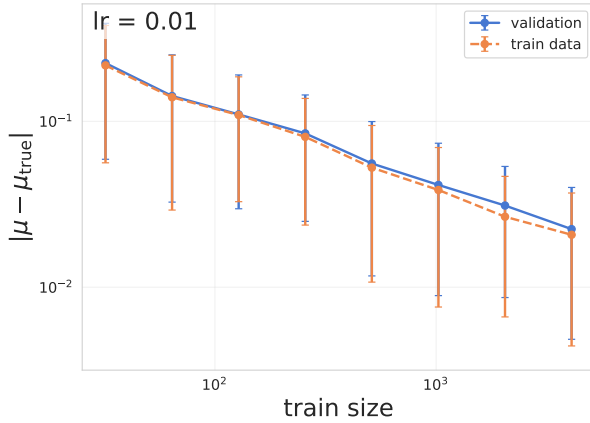


(a) Linear velocity field.

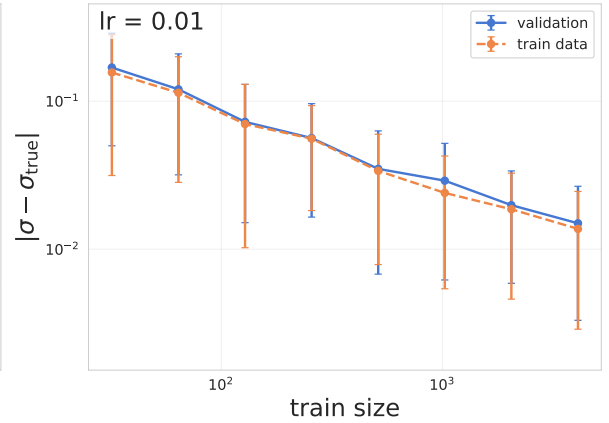


(b) MLP (4 hidden units).

Figure 4: Final Wasserstein distance versus training-set size at $\eta = 0.01$, comparing linear (left) and non-linear (right) velocity parametrizations. Lower is better; axes are in log-logs scale; error bars are computed across random seeds. The black dashed line correspond to the theoretical expectation of the Wasserstein distance computed according to eq. (23). The Wasserstein distance decreases monotonically with training size, but with divergent scaling behaviors: linear velocity scales more favorably, whereas nonlinear velocity saturates, suggesting diminishing returns at larger dataset sizes.



(a) Linear velocity field.



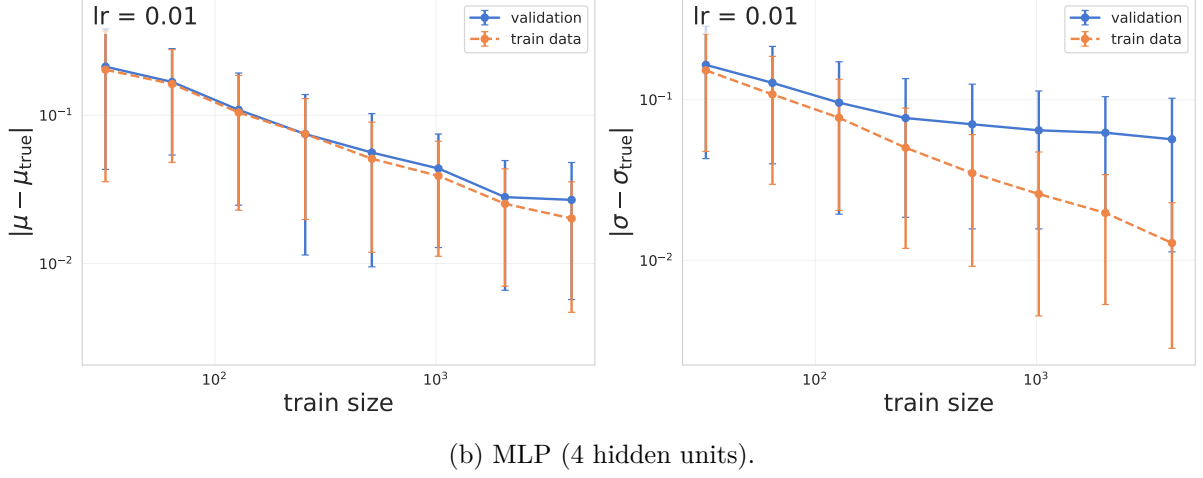


Figure 5: Distance between the true first two moments (mean and standard deviation) and the train sample mean and standard deviation (dashed orange line) and Flow Matching sample mean and standard deviation (blue line) versus training-set size, at $\eta = 0.01$. Top row: linear velocity field; bottom row: MLP with 4 hidden units. Axes are in log-log scale; errors are computed over trials. While the mean scales equally well for both models, the non-linear parametrization of the velocity field struggles to capture the standard deviation of the target Gaussian.

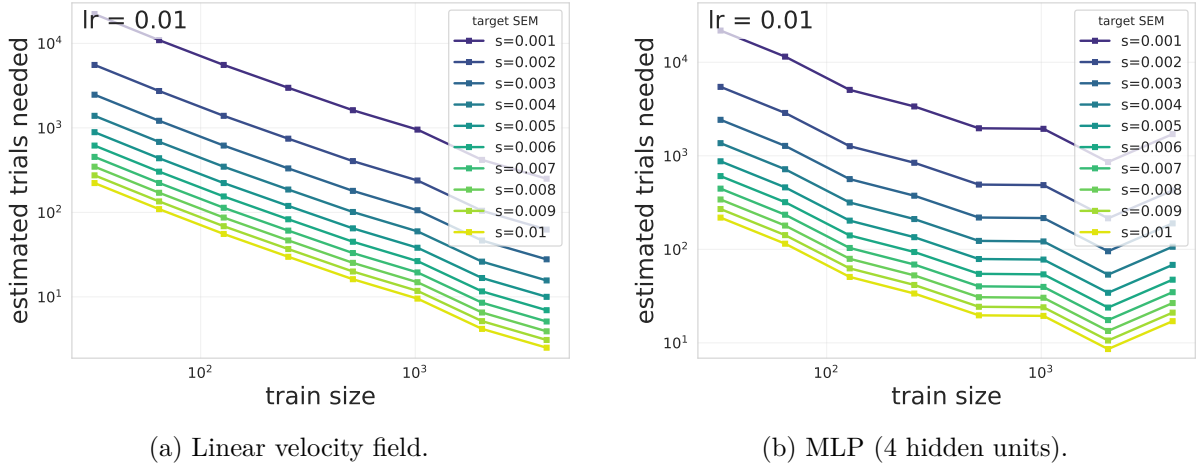


Figure 6: Number of random-seed trials required to reach the target error on the Wasserstein mean given a training-set size, at $\eta = 0.01$. Lower values indicate higher reliability of training for that configuration. Left: linear velocity field; right: non linear velocity field (MLP with 4 hidden units). Axis are in log-log scale.

3.2 AR(1) Process

3.2.1 Definitions

Definition 3.1 ($AR(1)$). An autoregressive process of order 1, denoted $AR(1)$ is a stochastic process defined by the equation

$$\begin{cases} X_t = \phi X_{t-1} + \xi_t, & \xi_t \sim \mathcal{N}(0, \sigma^2) \\ X_0 = x_0 \end{cases} \quad (29)$$

with $|\phi| < 1$ ensuring stationary.

Definition 3.2 (Increments). Given an $AR(1)$ process, we denote with

$$\Delta_t = X_t - X_{t-1}, \quad t \geq 1 \quad (30)$$

the increment of the process at time t .

3.2.2 Distributions of Interests

Stationary distribution Using the recursion in eq. (29), we can write the process as

$$X_t = \phi^t x_0 + \sum_{i=0}^{t-1} \phi^i \xi_{t-i} \quad (31)$$

Since X_t is a sum of Gaussians, its distribution is still Gaussian, hence it is enough to compute the mean

$$\mathbb{E}[X_t] = \phi^t x_0 + \sum_{i=0}^{t-1} \phi^i \mathbb{E}[\xi_{t-i}] = \phi^t x_0 \rightarrow 0 \quad \text{as } t \rightarrow \infty \quad (32)$$

and variance

$$\mathbb{V}[X_t] = \sum_{i=0}^{t-1} \phi^{2i} \mathbb{V}[\xi_{t-i}] = \sum_{i=0}^{t-1} \phi^{2i} \sigma^2 \rightarrow \frac{\sigma^2}{1 - \phi^2} \quad \text{as } t \rightarrow \infty \quad (33)$$

to show that

$$X_t \xrightarrow{d} \mathcal{N}(0, s_x^2) \quad \text{as } t \rightarrow \infty \quad (34)$$

with $s_x = \frac{\sigma}{\sqrt{1 - \phi^2}}$

Transition distribution The transition distribution is time-independent and given by

$$X_t | X_{t-1} = x_{t-1} \sim \mathcal{N}(\phi x_{t-1}, \sigma^2) \quad (35)$$

Following directly the definition of the process.

Stationary distribution of the increments Plugging eq. (29) in eq. (30) one can rewrite the increments as

$$\Delta_t = X_t - X_{t-1} = (\phi - 1)X_{t-1} + \xi_t \quad (36)$$

Since X_{t-1} is Gaussian, and the noise is Gaussian, also the increments are Gaussian and again in order to determine the distribution is sufficient to compute the mean

$$\mathbb{E}[\Delta_t] = (\phi - 1)\mathbb{E}[X_{t-1}] + \mathbb{E}[\xi_t] \rightarrow 0 \quad \text{as } t \rightarrow \infty \quad (37)$$

and the variance

$$\mathbb{V}[\Delta_t] = (\phi - 1)^2 \mathbb{V}[X_{t-1}] + \mathbb{V}[\xi_t] \quad (38)$$

$$= (\phi - 1)^2 \mathbb{V}[X_{t-1}] + \sigma^2 \quad (39)$$

$$\rightarrow (\phi - 1)^2 \frac{\sigma^2}{1 - \phi^2} + \sigma^2 \quad (40)$$

$$= \frac{2\sigma^2}{1 + \phi} \quad \text{as } t \rightarrow \infty \quad (41)$$

showing that

$$\Delta_t \xrightarrow{d} \mathcal{N}(0, s_\delta^2) \quad \text{as } t \rightarrow \infty \quad (42)$$

with $s_\delta = \sqrt{\frac{2}{1+\phi}}\sigma$

Transition distribution of the increments Instead of learning the transition distribution one can equivalently study the transition distribution for the intervals since $X_t = X_{t-1} + \Delta_t$. It follows directly from eq. (36) that

$$\Delta_t | X_{t-1} = x_{t-1} \sim \mathcal{N}((\phi - 1)x_{t-1}, \sigma^2) \quad (43)$$

3.2.3 Value Standardization

Value standardization consists in rescaling the process using the mean and standard deviation calculated over a training trajectory $\mathcal{T} = \{X_0, \dots, X_T\}$. Assuming that the statistics calculated over the dataset converges to the one of the stationary distribution, this procedure leads to the process

$$Z_t = \frac{X_t}{s_x} \quad (44)$$

Which now satisfy the recursion

$$Z_t = \phi Z_{t-1} + \frac{\xi_t}{s_x} \quad (45)$$

Clearly this makes the stationary distribution of Z_T a standard normal as desired, but one has to look at the effect of this standardization on the distribution of interest, namely the transition distribution and the transition distribution of the increments.

Transition distribution under value standardization Following directly from eq. (45) the transition distribution is given by

$$Z_t | Z_{t-1} = z_{t-1} \sim \mathcal{N}\left(\phi z_{t-1}, \frac{\sigma^2}{s_x^2}\right) \quad (46)$$

and the new variance is

$$\frac{\sigma^2}{s_x^2} = 1 - \phi^2 \quad (47)$$

Which converges to 0 as $\phi \rightarrow 1$. This could be problematic for flow matching, as it has to learn a distribution that is very peaked and concentrated.

Transition distribution of the increments under value standardization Similarly, the new conditional distribution of the intervals

$$R_t = Z_t - Z_{t-1} = (\phi - 1)Z_{t-1} + \frac{\xi_t}{s_x} \quad (48)$$

suffer the same problem since it is clear that also in this case

$$R_t | Z_{t-1} = z_{t-1} \sim \mathcal{N}\left((\phi - 1)z_{t-1}, \frac{\sigma^2}{s_x^2}\right) \quad (49)$$

3.2.4 Value-Increments Standardization

Another possible way is to standardize the values and the increments separately, each with its own standard deviation, i.e. we consider the new random variables

$$Z_t = \frac{X_t}{s_x} \quad (50)$$

$$R_t = \frac{\Delta_t}{s_\delta} \quad (51)$$

then it follows:

$$R_t = \frac{\Delta_t}{s_\delta} = \frac{X_t - X_{t-1}}{s_\delta} = \frac{s_x}{s_\delta} (Z_t - Z_{t-1}) = \frac{s_x}{s_\delta} \left[(\phi - 1)Z_{t-1} + \frac{\xi_t}{s_x} \right] \quad (52)$$

hence

$$R_t | Z_{t-1} = z_{t-1} \sim \mathcal{N} \left(-\sqrt{\frac{1-\phi}{2}} z_{t-1}, \frac{1+\phi}{2} \right) \quad (53)$$

Now the standard deviation is under control and bounded in $\left[\frac{1}{2}, 1\right]$.

3.2.5 Setup

The previous section highlighted several ways to approach the problem of training flow matching for probabilistic forecasting: learning the distribution of the next values, learning the distribution of the next increment, and choosing how to handle standardization. Each of these choices can affect the performance of flow matching. The goal of this section is precisely to clarify these aspects and compare the various methods. To address the standardization problem, two possible solutions are proposed and tested in two different experiments.

The first proposal follows [15]. The data is kept in the original space and the fixed source $\mathcal{N}(0, I)$ is replaced with a *learnable, conditioning-dependent* source distribution. It is expected, and confirmed experimentally, that this helps flow matching performance. The conditional source used is

$$p(\cdot | y) = \mathcal{N}(\mu_\theta(y), \sigma_\theta^2(y)) \quad (54)$$

with $\mu_\theta(y)$ and $\sigma_\theta^2(y)$ chosen to be linear functions with learnable parameters. As suggested in [15], a regularization on the loss function is added. This first experiment compares flow matching trained on standardized values, predicting the next value using the standard normal source, with flow matching trained using the conditional source without standardizing the data, both employing a linear parametrization of the velocity. The most relevant parameters used in this experiment are reported in table 2. The theoretical metrics considered are the Wasserstein distance and the oracle MSE. The empirical metrics, computable with a test dataset, are instead the negative log-likelihood, the MSE, and the CRPS, all evaluated on the validation set. For a detailed discussion of how these metrics are computed, see section 2.4.

The second proposal instead relies on jointly predicting the next values and the increments at the same time. This is very convenient to implement with flow matching, since it does not require training two different models but a single one, sampling from $q(\Delta_t, X_t | X_{t-1} = x_{t-1})$, or, using the previous notation, sampling from $q(R_t, Z_t | Z_{t-1} = z_{t-1})$, where values and increments are standardized independently. Clearly, this joint distribution is non-zero only for increments and values that are compatible. This allows for an a posteriori check of whether the model has learned to be self-consistent, by comparing the samples of the next value obtained with Z_t with those obtained using R_t to reconstruct the next value, and verifying whether they differ or not. A discrepancy would indicate that flow matching is failing to learn. The intuition behind this method is that flow matching is allowed to view the same problem from two different points of view, leading to a richer representation that can potentially improve performance.

Parameter	Value
AR(1) coefficient	$\phi \in \{0.25, 0.5, 0.75\}$
Noise standard dev.	$\sigma = 1$
Train dataset size	$N \in \{32, 64, 128, 256, 512, 1024, 2048, 4096\}$
Evaluation dataset size	512
Velocity parametrization	linear
Batch size	$\min(N, 1024)$
Learning rate	$\eta \in \{0.1, 0.01, 0.001\}$
Optimizer	SGD
Training iterations	2000
Trials	N -dependent

Table 2: Parameter values specified as ranges indicate that all combinations were tested independently. The number of trials per training-set size is 32:100, 64:100, 128:100, 256:100, 512:80, 1024:50, 2048:50, 4096:40.

This second experiment compares different standardization strategies: standardizing values and predicting values, standardizing values and increments while predicting increments, and the joint prediction of values and increments. Moreover, different parametrizations of the velocity are considered, introducing non-linearities. The parameters used for this experiment are reported in table 3. For computational reasons, only the Wasserstein distance and the oracle MSE have been considered as metrics in this experiment.

Parameter	Value
AR(1) coefficient	$\phi \in \{0.1, 0.5, 0.9\}$
Noise standard dev.	$\sigma = 100$
Train dataset size	$N \in \{32, 64, 128, 256, 512, 1024, 2048, 4096\}$
Evaluation dataset size	128
Velocity hidden units	$h \in \{0, 4, 8, 16, 32, 64\}$
Batch size	$\min(N, 1024)$
Learning rate	$\eta = 0.01 \rightarrow 0.0001$ (cosine annealing)
Optimizer	SGD
Training iterations	10.000
Trials	20

Table 3: Parameter values specified as ranges indicate that all combinations were tested independently.

3.2.6 Results Experiment n.1

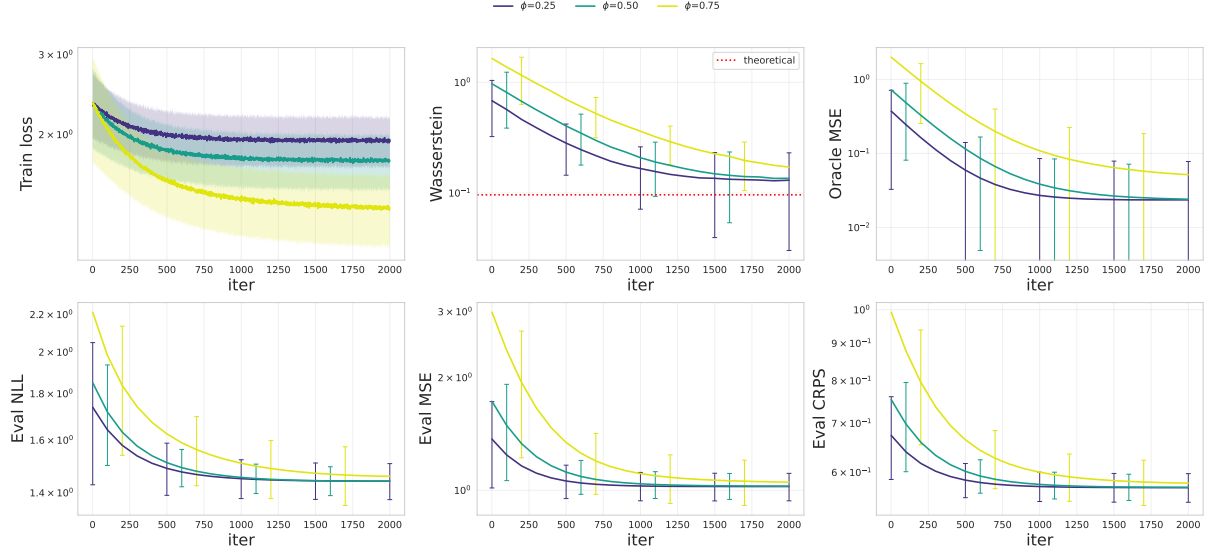
Learning dynamics. Figure 7 compares the learning dynamics of the model trained with a standard normal source against the one trained with the guided source. Both the models reach convergence. The key observation is that the guided source makes the loss value and the Wasserstein value essentially invariant to ϕ .

Scaling laws. Figure 8 illustrates how the theoretical metrics scale with training set size, revealing two key observations. First, both the Wasserstein distance and the oracle MSE exhibit

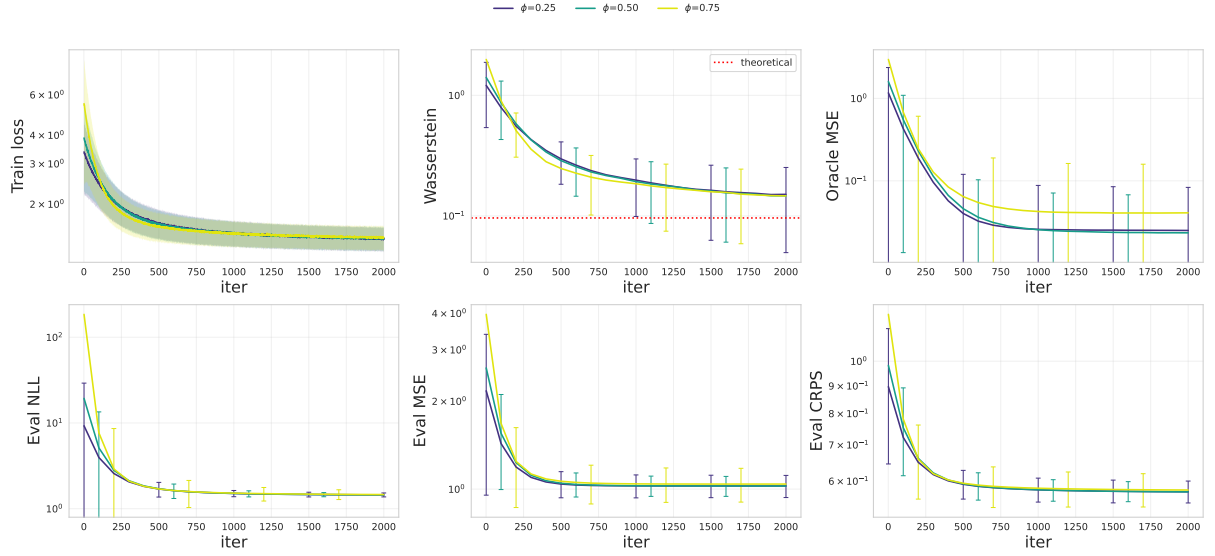
reduced sensitivity to ϕ under the guided source distribution, indicating more uniform performance across the parameter space. Second, to benchmark the flow-matching model as a point predictor, an optimal linear point forecaster of the form $\hat{y} = \alpha x$ is additionally trained, matching the analytical form of the true conditional mean, with α learned from data. A comparison of the oracle MSE between the flow-matching predictions and this optimal linear teacher shows that the two scaling curves coincide. Taken together, these results directly address the original research questions. Flow matching not only recovers the full predictive distribution accurately (as evidenced by the favorable Wasserstein scaling) but, when used as a point predictor, it also attains the performance of the optimal teacher forecaster. Moreover, the improved behavior under the guided source distribution highlights the benefit of adopting a data-driven preprocessing strategy.

Regime shifts stability. Figure 9 illustrates the stability of the trained model under regime shifts. Specifically, models are trained, for several training set sizes, on data generated by an $AR(1)$ process with $\phi = 0.5$, and then evaluated on validation sets produced by $AR(1)$ processes with different values of ϕ . As a function of the validation ϕ , all metrics exhibit a convex shape with a minimum at the true value $\phi = 0.5$ for train size large enough. The Wasserstein distance changes most sharply around the true value, whereas the effect is more moderate for the other metrics. Overall, this indicates that flow matching is sensitive to changes in the underlying data-generating process. This sensitivity could potentially be exploited to produce forecasts accompanied by a confidence score, for instance derived from the log-likelihood. The underlying idea is that the model is aware of regime shifts and can signal them, thereby providing an additional layer of reliability to its predictions.

A practical takeaway. By tracking both the theoretical metrics and their empirical counterparts, which are computable on a test set and therefore directly applicable to real-world data, a side-by-side comparison reveals that the two families exhibit closely aligned behavior, both in their training dynamics and in their sensitivity to regime shifts. This observation translates into a practical guideline: on real data, where theoretical metrics are inaccessible, the empirical metrics can serve as reliable proxies for monitoring model convergence and detecting regime shifts, thereby providing a principled basis for trusting the model’s predictions.



(a) Standard normal source distribution.



(b) Guided source distribution.

Figure 7: Learning dynamics across different values of the autoregressive parameter ϕ . The panel reports the training loss, evaluation loss, Wasserstein distance, evaluation negative log-likelihood, evaluation and oracle MSE, and evaluation CRPS. Panel (a) shows results for a model trained with a standard normal source distribution, while panel (b) shows results for a model trained with the proposed guided source distribution. Shaded regions and error bars represent variability across different training set sizes and independent trials. The learning rate is fixed at $\eta = 0.001$. In the plot showing the Wasserstein distance the dashed line corresponds to the theoretical expectation computed according to eq. (23). The y axis are in log scale.

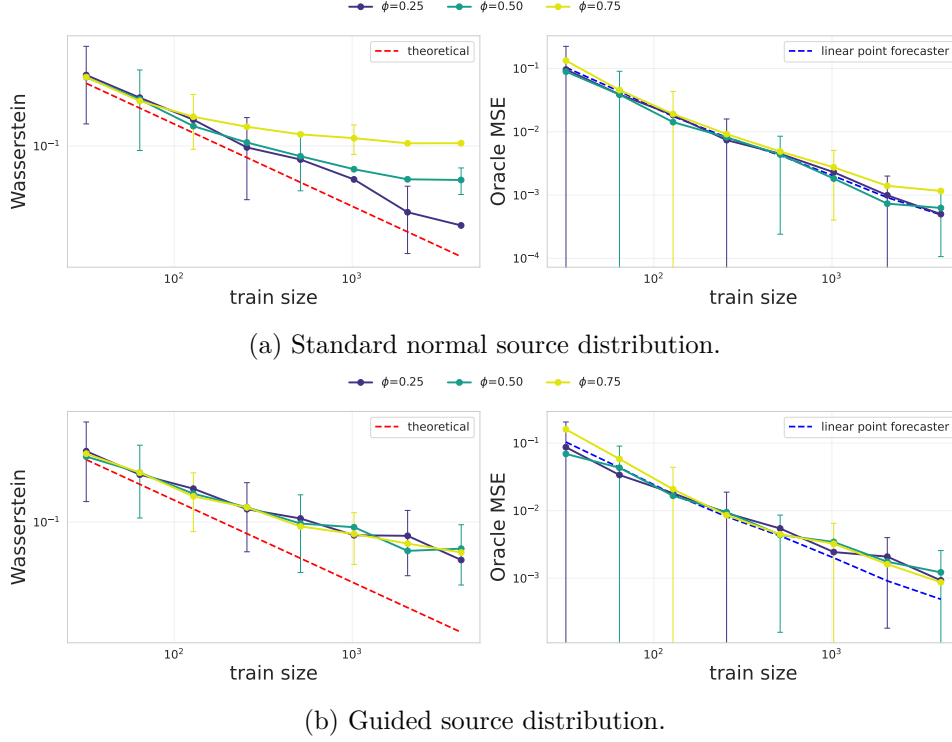


Figure 8: Scaling laws of the Wasserstein distance and oracle MSE as a function of the training set size, shown in log-log space, for different values of the autoregressive parameter ϕ . The learning rate is fixed to $\eta = 0.001$. Panel (a) corresponds to the model trained with a standard normal source distribution, while panel (b) corresponds to the model trained with the guided source distribution. Error bars are computed over independent trials. In the Wasserstein panels, the dashed line indicates the theoretical scaling prediction computed according to eq. (23). In the MSE panels, the dashed line denotes the MSE achieved by a teacher optimal student producing a single point estimate, included as a reference. Note in particular that using a guided source distribution reduces the effect of ϕ on the scaling of the Wasserstein distance, making it essentially ϕ -independent.

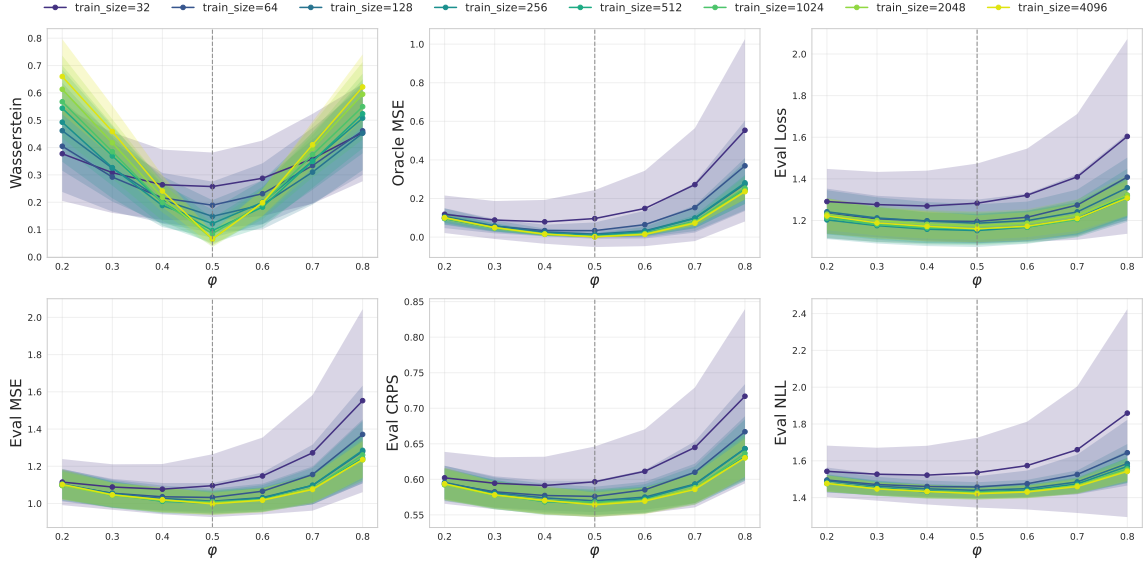


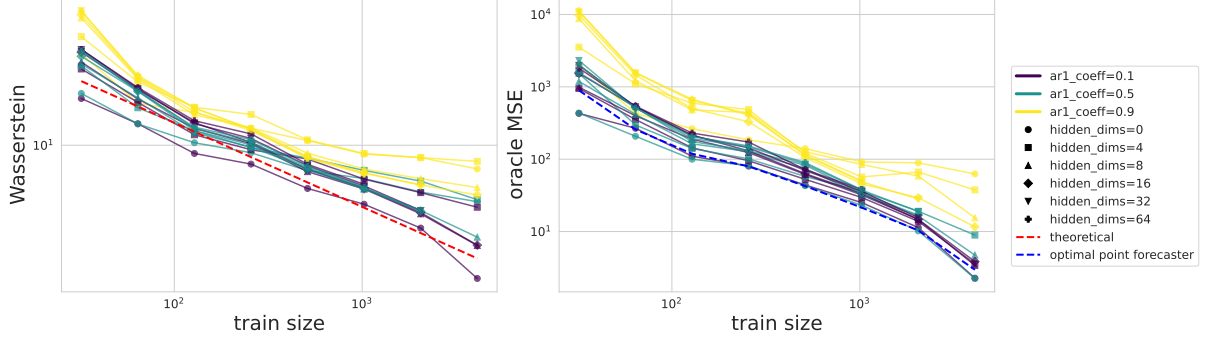
Figure 9: Validation metrics evaluated on datasets generated by an $AR(1)$ process with autoregressive parameter ϕ (x-axis), for several training set sizes. Each line corresponds to a model trained on data generated with the true parameter $\phi = 0.5$ (highlighted by the vertical dashed line) and evaluated under distribution shift induced by varying ϕ at test time. Shaded regions indicates the standard deviation over multiple trials.

3.2.7 Results Experiment n.2

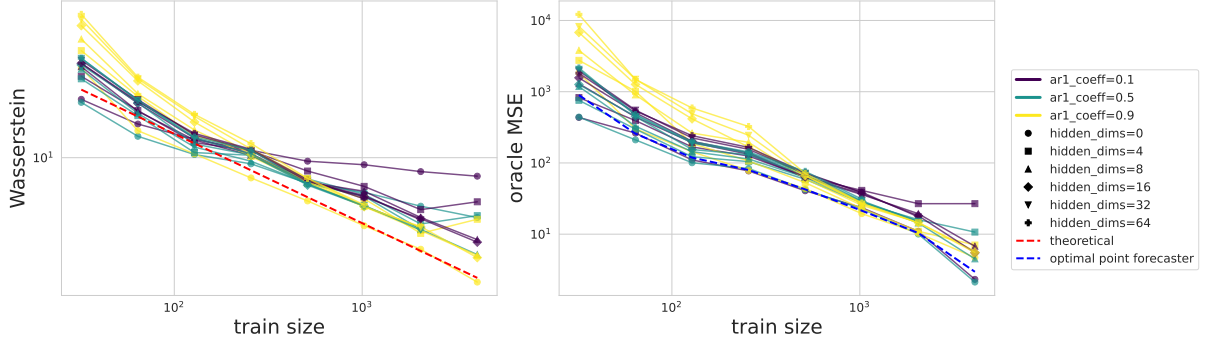
The results of the experiment are reported in fig. 10, from which one key observation can be made. The scaling laws of the Wasserstein distance for the model trained on standardized values and the one trained using increments are symmetric in the autoregressive parameter: the former performs better for small ϕ , while the latter performs better for large ϕ . Consequently, the model that uses both eventually performs close to the theoretical prediction⁶ for both large and small ϕ . To achieve this, it employs a non-linear parametrization of the velocity: this is the price to be paid for solving the more complex problem of joint prediction. Unlike using a conditional source which requires a careful tuning of the learning rate based on the scale on which the training data lives, this approach is scale-agnostic and adapts better to multiple scales. Regarding the MSE, the separation between different ϕ values is more pronounced in the case of standardized values, whereas it shrinks in the other two cases when a non-linear model is eventually used.

See appendix C for the plot showing the self-consistency of the predictions when using the model trained on jointly predicting values and increments.

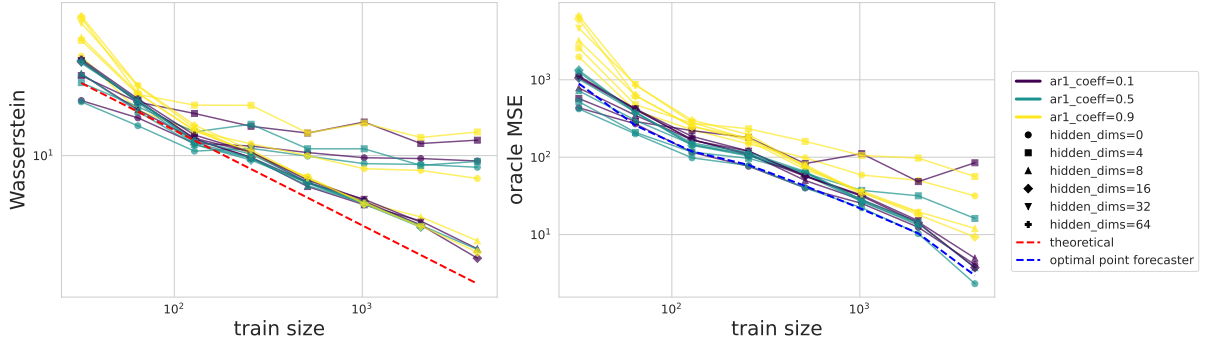
⁶See section 3.2.6 for a note on the two reference values that appear in the plots.



(a) Flow matching trained using standardized values.



(b) Flow matching trained using standardized increments.



(c) Flow matching trained using jointly standardized values and increments.

Figure 10: Scaling laws of the Wasserstein distance and oracle MSE as a function of the training set size, shown in log-log space, for different values of the autoregressive parameter ϕ (line color) and different parametrization of the velocity (marker style). Panel (a) corresponds to flow matching trained using standardized values. Panel (b) uses instead standardized intervals while panel (c) corresponds to the method jointly predicting intervals and values and the prediction considered is obtained by averaging over the two samples.

4 Conclusions

This work studied probabilistic time-series forecasting with flow matching in a deliberately controlled setting: synthetic data, known ground-truth transition distributions, low dimensionality, small training sets and small models. This regime mirrors the conditions under which forecasting systems are typically deployed in practice, particularly in finance, and crucially allows the quality of the learned distribution to be evaluated directly against the ground truth, something impossible in the high-dimensional domains where flow matching is usually applied and where faithfulness can only be assessed indirectly. This provides a principled way to address the question of *trust* in generative forecasters.

Two questions guided the analysis:

1. Does probabilistic forecasting match the accuracy of dedicated point-estimate models on the point-estimate task?
2. How faithful are the learned probabilities with respect to the ground truth?

The experiments on the AR(1) process show that flow matching used as a point predictor matches the MSE of an optimal linear point forecaster, while the full learned distribution converges to the true one in Wasserstein-1 distance at a rate close to the theoretical benchmark of [4]. Learning the full conditional distribution therefore does not deteriorate point-prediction performance, and the additional information is genuinely informative: the learned negative log-likelihood is sensitive to distribution shifts and can in principle be used as a confidence signal (e.g. by flagging as low-confidence any forecast whose NLL exceeds the 95th percentile measured on train-set). In this controlled setting, a generative forecaster can be trusted to do no worse than a point predictor while delivering a faithful predictive distribution.

Two methodological messages emerge. First, in this small-data regime results are highly sensitive to seed, learning rate and batch size, so reliable conclusions require multiple seeds, convergence checks, and comparison with theoretical benchmarks: the absence of an intuitive visual check for timeseries makes systematic quantitative validation indispensable. Second, standardization and type of prediction (increments vs values) is a critical modeling choice as shown by the difference performances that one achieves on processes with different persistence (the ϕ parameter).

Overall, the experiments support a moderately optimistic view of flow matching in the small-data, low-dimensional regime: it is competitive with classical point predictors while providing a faithful estimate of the predictive distribution, at the cost of a careful experimental methodology. The success of generative models on high-dimensional data does not transfer automatically, but with appropriate care it can transfer meaningfully.

Limitations. The targets are Gaussian or conditionally Gaussian, the dimension is one, the time-series are first-order Markov and stationary, and the models are small. It is not obvious that the same picture would hold for heavy-tailed, multimodal, multivariate, or non-stationary processes.

Perspectives. Natural extensions of this work concern adding complexity gradually. Examples include non-Gaussian (heavy-tailed or multimodal) transitions, higher-order or non-linear dynamics, multivariate time series, regime-switching models and real-world data.

References

- [1] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. “A learning algorithm for boltzmann machines”. In: *Cognitive Science* 9.1 (1985), pp. 147–169. ISSN: 0364-0213. DOI: [https://doi.org/10.1016/S0364-0213\(85\)80012-4](https://doi.org/10.1016/S0364-0213(85)80012-4). URL: <https://www.sciencedirect.com/science/article/pii/S0364021385800124>.
- [2] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. “Stochastic interpolants: A unifying framework for flows and diffusions, 2023”. In: URL <https://arxiv.org/abs/2303.08797> 3 (2023), p. 1.
- [3] Myles R. Allen and David A. Stainforth. “Towards objective probabilistic climate forecasting”. In: *Nature* 419.6903 (2002), pp. 228–228. DOI: 10.1038/nature01092a. URL: <https://doi.org/10.1038/nature01092a>.
- [4] Eustasio del Barrio, Evarist Giné, and Carlos Matrán. “Central Limit Theorems for the Wasserstein Distance Between the Empirical and the True Distributions”. In: *Annals of Probability - ANN PROBAB* 27 (Apr. 1999). DOI: 10.1214/aop/1022677394.
- [5] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. “Variational Inference: A Review for Statisticians”. In: *Journal of the American Statistical Association* 112.518 (Apr. 2017), pp. 859–877. ISSN: 1537-274X. DOI: 10.1080/01621459.2017.1285773. URL: <http://dx.doi.org/10.1080/01621459.2017.1285773>.
- [6] Xavier Bouthillier et al. *Accounting for Variance in Machine Learning Benchmarks*. 2021. arXiv: 2103.03098 [cs.LG]. URL: <https://arxiv.org/abs/2103.03098>.
- [7] Ricky TQ Chen et al. “Neural ordinary differential equations”. In: *Advances in neural information processing systems* 31 (2018).
- [8] Weisi Chen et al. “Deep Learning for Financial Time Series Prediction: A State-of-the-Art Review of Standalone and Hybrid Models”. In: *Computer Modeling in Engineering and Sciences* 139 (Dec. 2023), pp. 187–224. DOI: 10.32604/cmescs.2023.031388.
- [9] Yifan Chen et al. *Probabilistic Forecasting with Stochastic Interpolants and Föllmer Processes*. 2024. arXiv: 2403.13724 [cs.LG]. URL: <https://arxiv.org/abs/2403.13724>.
- [10] Ahmed ElGazzar and Marcel van Gerven. *Probabilistic Forecasting via Autoregressive Flow Matching*. 2026. arXiv: 2503.10375 [cs.LG]. URL: <https://arxiv.org/abs/2503.10375>.
- [11] Tilmann Gneiting and Matthias Katzfuss. “Probabilistic Forecasting”. In: *Annual Review of Statistics and Its Application* 1 (Jan. 2014), pp. 125–151. DOI: 10.1146/annurev-statistics-062713-085831.
- [12] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML]. URL: <https://arxiv.org/abs/1406.2661>.
- [13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: <https://arxiv.org/abs/2006.11239>.
- [14] Tao Hong and Shu Fan. “Probabilistic electric load forecasting: A tutorial review”. In: *International Journal of Forecasting* 32.3 (2016), pp. 914–938. ISSN: 0169-2070. DOI: <https://doi.org/10.1016/j.ijforecast.2015.11.011>. URL: <https://www.sciencedirect.com/science/article/pii/S0169207015001508>.
- [15] Junwan Kim et al. *Better Source, Better Flow: Learning Condition-Dependent Source Distribution for Flow Matching*. 2026. arXiv: 2602.05951 [cs.CV]. URL: <https://arxiv.org/abs/2602.05951>.
- [16] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *CoRR* abs/1312.6114 (2013). URL: <https://api.semanticscholar.org/CorpusID:216078090>.

- [17] Remi Lam et al. “Learning skillful medium-range global weather forecasting”. In: *Science* 382.6677 (2023), pp. 1416–1421. DOI: 10.1126/science.adi2336. eprint: <https://www.science.org/doi/pdf/10.1126/science.adi2336>. URL: <https://www.science.org/doi/abs/10.1126/science.adi2336>.
- [18] Yaron Lipman et al. “Flow matching for generative modeling”. In: *arXiv preprint arXiv:2210.02747* (2022).
- [19] Yaron Lipman et al. *Flow Matching Guide and Code*. 2024. arXiv: 2412.06264 [cs.LG]. URL: <https://arxiv.org/abs/2412.06264>.
- [20] Xingchao Liu, Chengyue Gong, and Qiang Liu. *Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow*. 2022. arXiv: 2209.03003 [cs.LG]. URL: <https://arxiv.org/abs/2209.03003>.
- [21] Gabriel Peyré and Marco Cuturi. *Computational Optimal Transport*. 2020. arXiv: 1803.00567 [stat.ML]. URL: <https://arxiv.org/abs/1803.00567>.
- [22] Syama Sundar Rangapuram et al. “Deep State Space Models for Time Series Forecasting”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf.
- [23] Kashif Rasul et al. *Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting*. 2021. arXiv: 2101.12072 [cs.LG]. URL: <https://arxiv.org/abs/2101.12072>.
- [24] Nicholas G. Reich et al. “A collaborative multiyear, multimodel assessment of seasonal influenza forecasting in the United States”. In: *Proceedings of the National Academy of Sciences* 116.8 (2019), pp. 3146–3154. DOI: 10.1073/pnas.1812594116. eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.1812594116>. URL: <https://www.pnas.org/doi/abs/10.1073/pnas.1812594116>.
- [25] Danilo Rezende and Shakir Mohamed. “Variational Inference with Normalizing Flows”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1530–1538. URL: <https://proceedings.mlr.press/v37/rezende15.html>.
- [26] David Salinas, Valentin Flunkert, and Jan Gasthaus. *DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks*. 2019. arXiv: 1704.04110 [cs.AI]. URL: <https://arxiv.org/abs/1704.04110>.
- [27] Jascha Sohl-Dickstein et al. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: 1503.03585 [cs.LG]. URL: <https://arxiv.org/abs/1503.03585>.
- [28] Yang Song et al. *Score-Based Generative Modeling through Stochastic Differential Equations*. 2021. arXiv: 2011.13456 [cs.LG]. URL: <https://arxiv.org/abs/2011.13456>.
- [29] Lucas Theis, Aäron van den Oord, and Matthias Bethge. *A note on the evaluation of generative models*. 2016. arXiv: 1511.01844 [stat.ML]. URL: <https://arxiv.org/abs/1511.01844>.
- [30] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a8Paper.pdf.
- [31] Cédric Villani. *Optimal Transport: Old and New*. Vol. 338. Grundlehren der mathematischen Wissenschaften (Fundamental Principles of Mathematical Sciences). Springer, 2008. DOI: 10.1007/978-3-540-71050-9.

- [32] Wei Jie Yeo et al. “A comprehensive review on financial explainable AI”. In: *Artificial Intelligence Review* 58.6 (2025), p. 189. DOI: 10 . 1007 / s10462 - 024 - 11077 - 7. URL: <https://doi.org/10.1007/s10462-024-11077-7>.

A Algorithms

Algorithm 1 Training a guided flow matching model.

Require: Dataset $\mathcal{D} = \{(x_1^\mu, y^\mu)\}_{\mu=1}^M$, source distribution p , conditional path $\psi_t(\cdot \mid x_1)$, velocity network $u_t^\theta(\cdot \mid y)$, learning rate η , batch size B .

```

1: while not converged do
2:   Sample a mini-batch  $\mathcal{B} \subset \{1, \dots, M\}$  of size  $B$ 
3:   for each  $\mu \in \mathcal{B}$  do
4:      $t^\mu \sim \mathcal{U}(0, 1)$ 
5:      $x_0^\mu \sim p$ 
6:      $x_t^\mu \leftarrow \psi_{t^\mu}(x_0^\mu \mid x_1^\mu)$ 
7:      $v_{\text{target}}^\mu \leftarrow \dot{\psi}_{t^\mu}(x_0^\mu \mid x_1^\mu)$ 
8:   end for
9:    $\hat{\mathcal{L}}(\theta) \leftarrow \frac{1}{B} \sum_{\mu \in \mathcal{B}} \|v_{\text{target}}^\mu - u_{t^\mu}^\theta(x_t^\mu \mid y^\mu)\|^2$ 
10:   $\theta \leftarrow \theta - \eta \nabla_\theta \hat{\mathcal{L}}(\theta)$ 
11: end while
12: return  $\theta$ 

```

Algorithm 2 Sampling via forward Euler integration of the learned Neural ODE.

Require: Trained velocity field u_t^θ , source distribution p , guidance signal y , number of integration steps N .

```

1:  $h \leftarrow 1/N$  ▷ step size
2:  $x \sim p$  ▷ draw initial noise
3: for  $n = 0, 1, \dots, N-1$  do
4:    $t \leftarrow n h$ 
5:    $x \leftarrow x + h \cdot u_t^\theta(x \mid y)$ 
6: end for
7: return  $x$  ▷ generated sample at  $t = 1$ 

```

Algorithm 3 Likelihood estimation via backward Euler integration of the learned Neural ODE.

Require: Trained velocity field u_t^θ , source log-density $\log p$, guidance signal y , target point x_1 , number of integration steps N .

```

1:  $h \leftarrow 1/N$  ▷ step size
2:  $x \leftarrow x_1$ 
3:  $\ell \leftarrow 0$  ▷ accumulated divergence
4: for  $n = N, N-1, \dots, 1$  do
5:    $t \leftarrow n h$ 
6:    $\ell \leftarrow \ell + h \cdot \nabla \cdot u_t^\theta(x \mid y)$ 
7:    $x \leftarrow x - h \cdot u_t^\theta(x \mid y)$  ▷ backward Euler step
8: end for
9: return  $\log q(x_1) \leftarrow \log p(x) - \ell$ 

```

B Gaussian Target Velocity Field

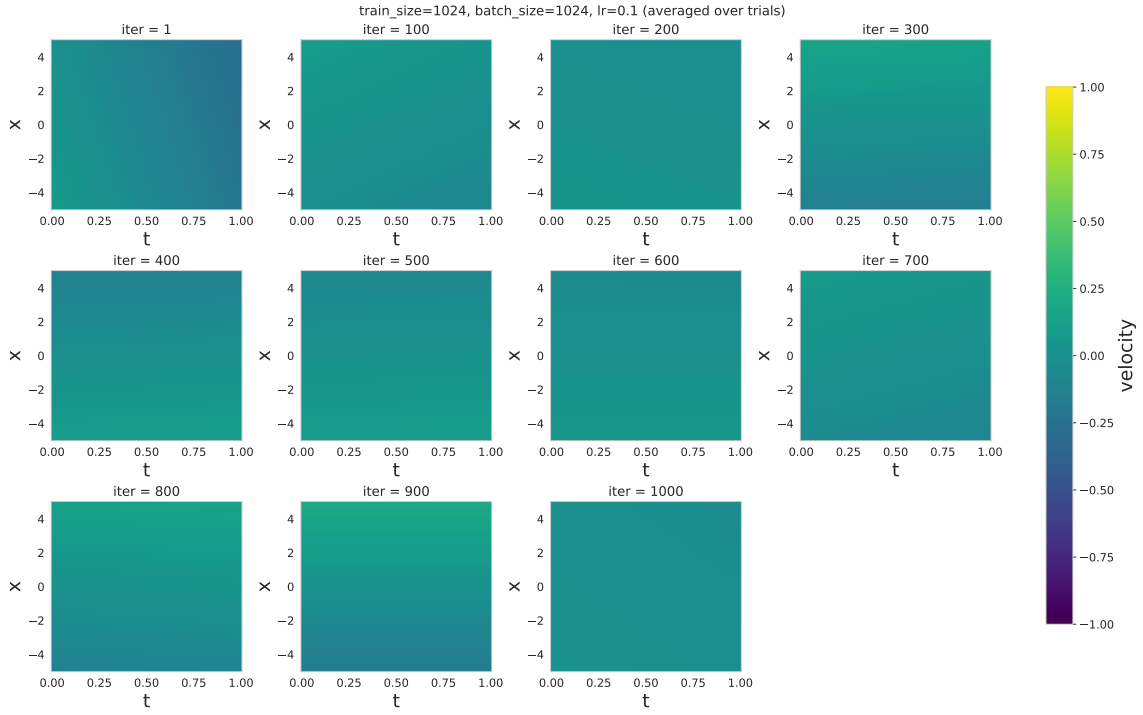


Figure 11: Evolution of the learned linear velocity field during training for a Gaussian target distribution.

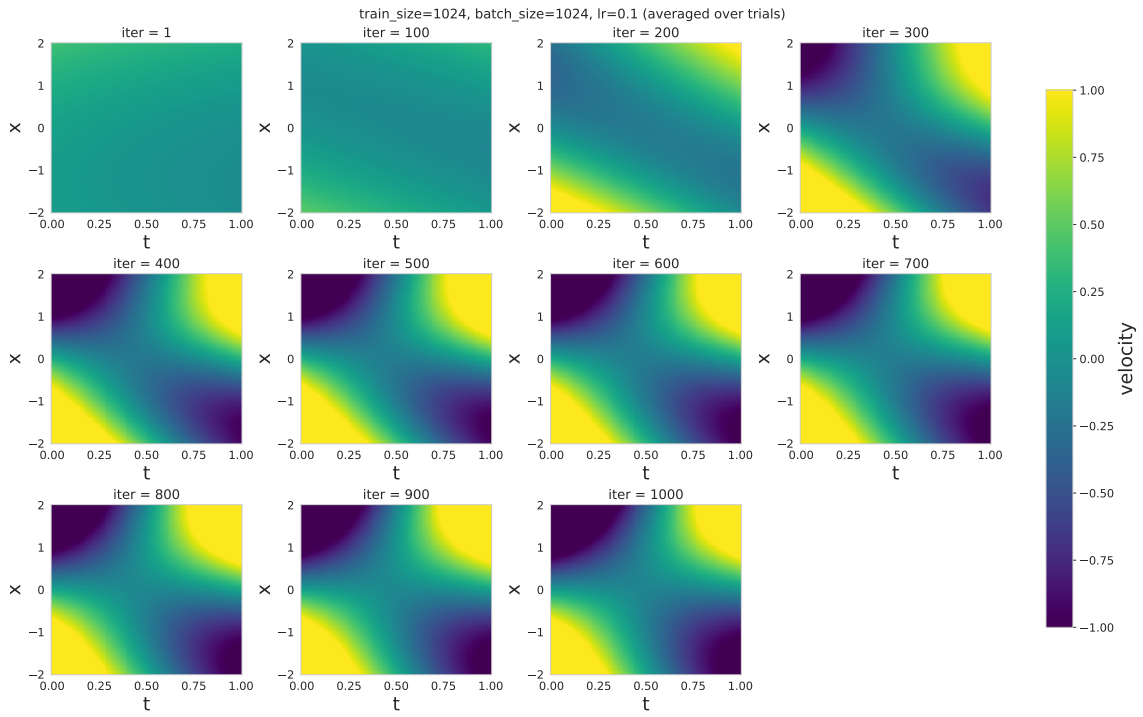


Figure 12: Evolution of the learned non-linear velocity field during training for a Gaussian target distribution.

C Self Consistency

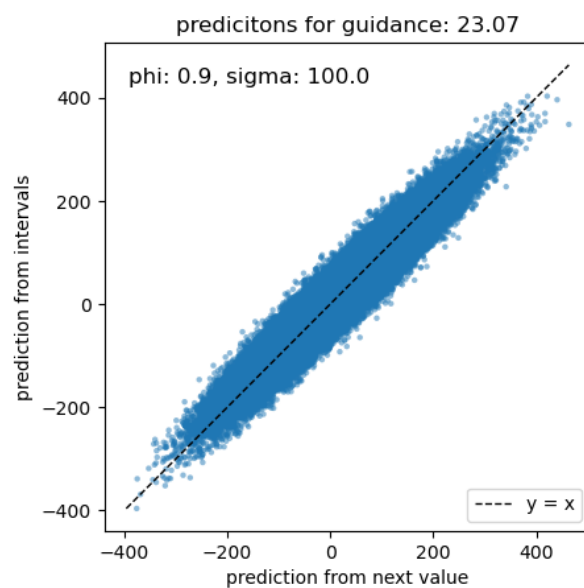


Figure 13: Samples from flow matching trained to predict values and intervals jointly for a given value of the guidance (conditioning variable). The model has learned to be self-consistent: prediction from intervals and from values coincide.