Mitigating Memorization in Closed-Form Flow Matching with Bootstrap Aggregating

Circée Chalayer[12], Quentin Bertrand[12], Emilie Morvant[1], and Rémi Emonet[123] (✉)

[1] Université Jean Monnet Saint-Étienne, CNRS, Institut d'Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023 Saint-Étienne, France
[2] Inria [3] Institut Universitaire de France

Abstract. Diffusion and flow matching are a class of generative models that generate new samples by solving ordinary or stochastic differential equations with a learned score/velocity field. Interestingly, the optimal velocity field admits a closed-form formula that can be computed for finite datasets. Generating samples following the optimal velocity field can only reproduce samples from the training set, *i.e.*, memorize the training set. Neural networks, trained to match the velocity field, introduce inductive bias that can partially mitigate the issue. However, these models can still exhibit memorization, unlike the benign overfitting observed in discriminative tasks. In this paper, we propose a bootstrap aggregating (bagging) method for flow matching to reduce memorization. By conceptually averaging over resampled training subsets, our approach effectively reduces memorization. We derive a closed-form bagging formulation compatible with exact flow matching, enabling efficient implementation. Experiments confirm reduced memorization and better generalization without architectural changes or auxiliary objectives.

Keywords: Conditional Flow Matching · Generalization · Memorization · Bagging · Bootstrap Aggregating.

1 Introduction

Deep generative models, such as diffusion models [30,15,33] and flow matching models [23,1,25], can now generate realistic samples across a wide range of modalities. These models are capable of generating new content in domains such as images [34], audio [5], video [36], and text [9,18]. Despite their groundbreaking empirical performance, the generalization mechanisms underlying these models remain poorly understood. While modern deep generative models can produce novel samples and empirically exhibit clear signs of generalization [16], partial memorization of the training data by large generative models has also been observed in several empirical studies (see, *e.g.*, [7,31,32,8]). Moreover, this phenomenon has been investigated theoretically in the context of flow matching and diffusion models (see, *e.g.*, [29,14,3,22,2]).

In contrast to what is observed in discriminative learning, a key challenge of generative models is the lack of benign overfitting: perfectly minimizing the flow

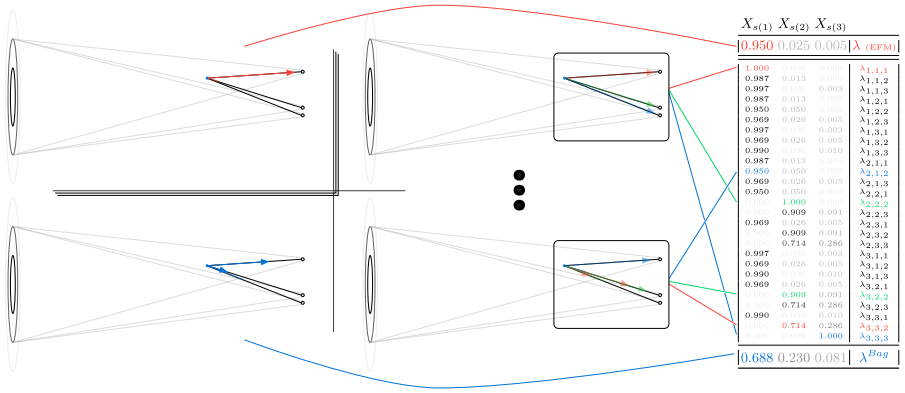

Fig. 1: Bagging of the closed-form flow matching, illustrated with $N = 3$ points in the training set. Dashed, arrowless lines are vectors of small magnitude scaled for visualization. The closed-form solution of flow matching points almost exactly to the closest training point (top left), inducing pure memorization. The bagging approach first (virtually) computes closed-form solutions of bootstrap samples (right), and then averages the resulting velocity fields (bottom left). This article explores the effect of bagging on memorization in generative models and proposes an approach to efficiently work around the combinatorial nature of bagging.

matching or diffusion training objective yields a model that can only generate training samples [2], leading to the so-called *memorization* of the training set. Understanding why practical models nevertheless generate novel samples has therefore become an important research question.

Several hypotheses have been proposed to explain this phenomenon. These include the *inductive bias* of the neural network architectures [16], the *implicit bias* of the optimization procedures [4], noise in the training process [35], and the *exposure bias*, *i.e.*, the mismatch between intermediate distributions encountered during training and those encountered during inference [28]. Each of these hypotheses has led to a growing line of work aiming to design methods that mitigate memorization and improve generalization in generative models. In particular, several recent works attempt to construct closed-form generative models that can generalize without training, either by explicitly incorporating inductive biases such as equivariance and locality [17,26,27], or by introducing additional layers of stochasticity during the sampling process [29].

Contributions. In this paper, we propose a new approach to mitigate memorization in flow matching models based on the principle of *bootstrap aggregating* (bagging [6]). Bagging reduces the variance of a predictor by averaging models trained on different resamples of the training dataset, resulting in a reduction of the influence of individual samples and improving stability with respect to small dataset perturbations. The usual application of bagging to flow matching would require training multiple models on different resampled datasets and av-

eraging their predictions, which would be computationally expensive. Moreover, given the inductive biases of neural architectures and the implicit regularization induced by optimization procedures mentioned above, training multiple models may provide only limited additional benefits. Instead, we propose to explore the use of bagging directly at the *closed-form level*. Rather than retraining multiple models, we average the closed-form of the flow associated with resamples of the dataset. Intuitively, our approach reduces the dependence of the learned flow on individual training samples and introduces a form of stochasticity that mitigates memorization. The intuition behind our approach is illustrated in Figure 1.

Organization of the paper. Section 2 provides an introduction to conditional flow-matching, emphasizing the distinction between the closed-form (CFF) and matching methods that train a model to estimate the (CFM and EFM). It then recalls the principle of bootstrap aggregating that averages predictions from models obtained by bootstrap samples, *i.e.*, datasets created by resampling the training set. Section 3 introduces the core of our contribution, showing how bagging can be applied directly to the closed-form of the flow, implemented by sampling or by a fast alternate solution. Section 4 studies how well the approach can improve generalization and reduce memorization. Due to space constraints, we push some interesting connected discussions, but not directly necessary, to the supplementary material B.

2 Background

2.1 Conditional Flow Matching in a Nutshell

In flow matching[3], we assume a source distribution p_0 and a data distribution p_{data}. For simplicity, we consider $p_0 = \mathcal{N}(0, \mathrm{I_d})$ as the unit Gaussian source distribution (more general p_0 can be considered). We observe a training dataset $X := (X_i)_{i=1}^N \in (\mathbb{R}^d)^N$ consisting of N data points independently drawn from p_{data}. We denote by $\hat{p}_{data} := \frac{1}{N}\sum_{i=1}^N \delta_{X_i}$ the empirical training distribution.

The objective of flow-based models (and diffusion, in a stochastic version) is to rely on a time-dependent velocity field $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ to transport p_0 to $\hat{p}_{data}$. The problem is expressed as solving, on $[0,1]$, the following ordinary differential equation:

$$\begin{cases} x(0) = x_0 \sim p_0 \\ \dot{x}(t) = u(x(t), t). \end{cases} \tag{1}$$

With the proper velocity field u, if the initial condition x_0 is drawn from p_0, the law of $x(1)$, associated with the terminal time $t = 1$, should follow $\hat{p}_{data}$. The intermediate distributions of $x(t)$, for every $t \in [0,1]$, define a probability path, denoted by $p(\cdot|t)$, that progressively transforms p_0 into $\hat{p}_{data}$. Once such a velocity field u

[3] The reader can refer to [24,12,13] for introductions to flow matching.

is known, one generates new samples by first sampling x_0 from p_0, then solving the ordinary differential equation, and finally outputting $x(1)$ as a generated sample.

A conditional flow model constructs this velocity field by *(i)* first defining a conditioning variable z independent of t, typically $z \sim \hat{p}_{data}$, *(ii)* then choosing a conditional probability path $p^{\text{cond}}_{X_i}(\cdot|t) := p(\cdot|z = X_i, t) = \mathcal{N}(tX_i, (1-t)^2 \mathrm{I_d})$.

Given such a conditional probability path $p^{\text{cond}}_{X_i}(\cdot|t)$, the continuity equation (Sec. 3.5 of [24]) defines a corresponding conditional velocity field transporting p_0 to $\hat{p}_{data}$. With *(i)* and *(ii)*, this field is simply defined by

$$u^{\text{cond}}_{X_i}(x,t) \;:=\; u^{\text{cond}}(x, z = X_i, t) \;= \frac{X_i - x}{1-t}. \tag{2}$$

Marginalizing the conditional path over z implies a probability path $p(\cdot|t)$, and thus defines an optimal velocity field transporting p_0 to $\hat{p}_{data}$ (Thm. 1 of [23]), denoted by u^*_X and defined by

$$u^*_X(x,t) \;:=\; \mathbb{E}_{z=X_i|x,t}\, u^{\text{cond}}_{X_i}(x,t). \tag{3}$$

Flow Matching (**FM**) consists in training a model (typically a neural network) to match the velocity field governing a given flow-based model. In principle, one could approximate u^* with a neural network $u_\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ with parameters θ by minimizing

$$\mathcal{L}_{FM}(\theta) := \mathbb{E}_{\substack{t\sim\mathcal{U}([0,1]) \\ x_t\sim p(\cdot|t)}} \left\| u_\theta(x_t,t) - u^*_X(x_t,t) \right\|^2.$$

However, this target field u^*_X is usually intractable, and this approach has initially been dismissed. Indeed, *Conditional Flow Matching* (**CFM**), introduced by Lipman *et al.* [23], rather trains a model to fit the conditional velocity field, using

$$\begin{aligned} \mathcal{L}_{CFM}(\theta) &:= \mathbb{E}_{\substack{x_0\sim p_0 \\ X_i\sim\hat{p}_{data} \\ t\sim\mathcal{U}([0,1])}} \left\| u_\theta(x_t,t) - u^{\text{cond}}_{X_i}(x_t,t) \right\|^2, \\ \text{with}\quad x_t &:= (1-t)x_0 + tX_i, \\ \text{and}\quad u^{\text{cond}}_{X_i}(x_t,t) &= \frac{X_i - x_t}{1-t} = X_i - x_0. \end{aligned}$$

While the regression target is stochastic, it can be shown that $\mathcal{L}_{CFM}$ and $\mathcal{L}_{FM}$ have the same minimizers. In practice, the CFM objective is easy to compute since sampling from p_0, from $\mathcal{U}([0,1])$ and from the empirical training distribution $\hat{p}_{data}$ is straightforward.

We call *Closed-Form Flow* (**CFF**), the exact minimizer of $\mathcal{L}_{FM}$ (and $\mathcal{L}_{CFM}$). Indeed, if we put no architectural constraints on u_θ, then the flow matching problem decomposes into problems at every position (x,t) that are all independent.

The optimal velocity field of Equation (3) can be expressed by the closed-form[4]:

$$u_X^*(x,t) \;=\; \sum_{i=1}^{N} \lambda(X,x,t)_i \; \frac{X_i - x}{1-t}, \tag{4}$$

where $\lambda(X,x,t)_i$ denotes the softmax weight of the point X_i in the dataset X and is defined for all $i \in \{1,\dots,N\}$ by

$$\begin{aligned} \lambda(X,x,t)_i \;&=\; \frac{p^{\text{cond}}_{X_j}(x|t)}{\sum_{k=1}^{N} p^{\text{cond}}_{X_k}(x|t)} \\ &=\; \text{softmax}\left(\left(\frac{-\| \, x - tX_j\|^2}{2(1-t)^2}\right)_{j=1}^{N}\right)_i . \end{aligned} \tag{5}$$

Note that such a velocity field $u_X^*(x,t)$ is optimal for the empirical probability distribution $\hat{p}_{data}$ and not for the true data distribution p_{data}. *Empirical Flow Matching* (EFM) [2] trains a model to regress against this closed-form expression of the optimal empirical velocity field.

An important property emerges when t tends to 1. In that regime, the softmax weights concentrate on the closest training sample, and the magnitude of $u_X^*(x,t)$ diverges for any point x that does not coincide with one of the data points. Consequently, integrating the ordinary differential equation (1) leads to trajectories that converge to the training samples themselves (see Thm. 4.6 of [14]). This reveals a fundamental paradox: perfectly minimizing the conditional flow matching objective would lead to the field $u_\theta = u_X^*(x,t)$ which effectively memorizes the training dataset. This memorization "by design" restricts generation to training points only (*i.e.*, points from $\hat{p}_{data}$) and limits the ability to generate points from the true unknown data distribution p_{data}.

Interestingly, from Equations (2) and (4) the optimal velocity field $u_X^*(x,t)$ can be interpreted as a weighted average of the N different directions $X_i - x$ associated with the training points. The weights $\lambda(X,x,t)_i$ depend on the relative probability of each conditional path. However, when t tends to 1, these softmax weights concentrate on a single training point, effectively producing a nearly one-hot weight vector. In this regime, the velocity field is dominated by the nearest training sample, which results in trajectories converging to individual training samples. This mechanism can explain why the optimal closed-form velocity field memorizes, by design, the dataset. This observation suggests that memorization is closely related to the strong concentration of the weights $\lambda(X,x,t)_i$.

Intuitively, this memorization effect in closed-form flow matching can be interpreted as a form of high variance: the velocity field is sensitive to the presence of individual training points. Therefore, a way to mitigate this instability could be to average the velocity field to reduce its dependence on any single training

[4] The closed-form of the optimal velocity can be found in previous works such as [17,3,14,22,29], and can be generalized to other choices of continuous distribution p_0.

point. We propose to tackle this effect through bootstrap aggregating (recalled in the next section), which introduces randomness and averages the resulting velocity fields, leading to a reduction of the memorization.

2.2 Bagging (Bootstrap Aggregating) in a Nutshell

Bagging (bootstrap aggregating) was introduced by Breiman [6] for traditional machine learning settings such as regression or classification. Bagging is a common ensemble learning method [10] designed to improve predictive performance by combining models learned from randomly altered versions of the training dataset. The intuition is the following: if the training data we have come from independent sampling from an unknown distribution, then any resampling of our training set is a likely training set that we could have had.

The principle consists of learning several models from different resamples of the original dataset, called bootstrap samples. At a high level, averaging the predictions of multiple models learned on different bootstrap samples stabilizes the learning procedure and reduces variance without significantly increasing bias. This is especially the case for unstable learning algorithms, for which small perturbations of the training dataset can lead to significantly different models, thereby providing diversity among the models to be aggregated (which is known to be key in ensemble learning; see, *e.g.*, [20]).

More formally, we consider a training set $X = (X_i)_{i=1}^N$ of size N where the points are drawn independently from an unknown data distribution p_{data}, *i.e.*, $X_i \overset{iid}{\sim} p_{data}$. The bootstrap samples are obtained by resampling the dataset with replacement, implying that a bootstrap sample can contain repeated points and typically omits some observations. We denote by $\mathcal{B} = \mathrm{Cat}(([1, N])^N)$ the categorical uniform distribution of all possible bootstrap samples of size N on a set of size N (other sizes of bootstrap samples may be used). Then, we draw M bootstrap samples $\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(M)} \sim \mathcal{B}$, where each $\mathbf{b}^{(m)} \in ([1, N])^N$ is a vector of indices. Therefore, each bootstrap sample defines a resampled dataset $X_{\mathbf{b}^{(m)}} = (X_{\mathbf{b}_j^{(m)}})_{j=1}^N$. For each $X_{\mathbf{b}^{(m)}}$, we learn a model $\hat{h}_{X_{\mathbf{b}^{(m)}}}$. The final model is obtained by averaging the predictions of these M models:

$$\forall x \; ; \;\; \hat{h}^{Bag}(x) \;=\; \frac{1}{M} \sum_{m=1}^{M} \hat{h}_{X_{\mathbf{b}^{(m)}}}(x).$$

In other words, bagging approximates the expectation of the model over bootstrap resamples of the training dataset.

When $M \to \infty$, we obtain the "ideal bagging" that corresponds to the expectation over all possible bootstrap samples. This expectation is on a discrete set. Indeed, since a bootstrap sample of size N corresponds to N independent draws from the N training points, there are N^N possible bootstrap samples.

Hence, the ideal final model is

$$\forall x \; ; \quad h^{Bag}(x) \;=\; \mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[\hat{h}_{X_b}(x)\right] \\ \;=\; \frac{1}{N^N}\sum_{m=1}^{N^N}\hat{h}_{X_{\mathbf{b}^{(m)}}}(x), \tag{6}$$

where $\mathbf{b}^{(m)}$ denotes the m^{th} bootstrap sample when enumerating the N^N possible ones. As the order (within a bootstrap sample) is not important, the count of distinct bootstrap samples is the number of n-multicombinations (with replacement), *i.e.*, $\left(\binom{N}{N}\right) = \binom{2N-1}{N} = \frac{(2N-1)!}{N!(N-1)!}$ with grows more slowly but still grows exponentially fast. In practice, this combinatorial expectation is never computed exactly and is instead approximated by averaging over a finite number M of randomly sampled bootstrap datasets.

In this paper, we leverage the principle of bagging for flow matching to reduce memorization. We manage to combine the flow matching objective with the ideal bagging formulation, showing it can be implemented without sampling.

3 Bootstrap Aggregating for Closed-Form Empirical Flow

In this section, we introduce our contributions, showing how the bagging principle can be used in generative models. As mentioned in Section 2.1, we want to reduce the dependence of the closed-form on any training points to alleviate memorization. To do so, instead of averaging several learned models as done in classical bagging, we focus on bagging the closed-form solution of the flow-matching objective to introduce stochasticity in the velocity field. In Section 3.1, we start with the ideal bagging formulation, denoted by u^{BCFF}, for the velocity field of flow matching and then we detail its sampled approximation, denoted by u^{SBB}. We then show how the ideal bagging formulation can be combined with the one of flow matching to get a practical implementation that has bagging benefits without the cost of sampling.
Notational guide. See the table in supplementary material A that summarizes the main notations used in the paper.

3.1 "Ideal" Bagging for Closed-Form Flow

As mentioned in Section 2.2, ideal bagging can be formulated as an average over all possible bootstrap samples; see Equation (6). However, it is usually never even formulated because of its combinatorial cost. We propose *Bagging Closed-Form Flow* (**BCFF**) applying bootstrap aggregating to the closed-form flow (CFF) formulation. BCFF can be expressed as a discrete expectation, which can be written as the following sum:

$$\forall x,t \; ; \; u^{BCFF}(x,t) \;=\; \mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\star}_{X_{\mathbf{b}}}(x,t)\right] \\ \;=\; \frac{1}{N^N}\sum_{m=1}^{N^N}u^{\star}_{X_{\mathbf{b}^{(m)}}}(x,t),$$

where $\mathbf{b}^{(m)}$ denotes the m^{th} bootstrap sample when enumerating the N^N possible bootstrap samples, and $u^\star_{X_{\mathbf{b}^{(m)}}}(x,t)$ is the optimal velocity field (Equation (4)) associated to the resampled dataset $X_{\mathbf{b}^{(m)}} = (X_{\mathbf{b}_j^{(m)}})_{j=1}^N$.

3.2 Sampling-Based Bagging (SBB)

To make the bootstrap tractable, the common approach is to not consider all the N^N possible cases but only a random subset of them. We thus introduce *Sampling-Based Bagging* (**SBB**) (short for SBBCFF), a sampling-based bagging method that averages over M random bootstrap samples drawn from the N^N possible ones. Formally, this results in a stochastic closed-form expressed as

$$\forall x,t\ ;\ u_M^{SBB}(x,t) \ = \ \frac{1}{M}\sum_{m=1}^{M} u^\star_{X_{\mathbf{b}^{(m)}}}(x,t),$$

where the M bootstrap samples are each made of independent resampling with replacement: $\forall m,\ \mathbf{b}^{(m)} \sim \mathcal{B}$, *i.e.*, $\forall m,j,\ \mathbf{b}_j^{(m)} \sim \mathrm{Cat}([1,N])$. Therefore, the solution is an average of standard flow matching closed-form solutions, each of which is a weighted average with softmax weights.

We denote by $\mathbf{B}$ the points corresponding to the bootstrap samples $\mathbf{b}$. More precisely, $\mathbf{B}_j^{(m)} := (X_{\mathbf{b}^{(m)}})_j = X_{\mathbf{b}_j^{(m)}}$ is one of the dataset points, with the possibility that $\mathbf{B}_{j_1}^{(m)} = \mathbf{B}_{j_2}^{(m)}$ even if $j_1 \neq j_2$. Then, we have

$$u_M^{SBB}(x,t) \ = \ \frac{1}{M}\sum_{m=1}^{M}\sum_{j=1}^{N} \lambda(\mathbf{B}^{(m)},x,t)_j \ u^{\mathrm{cond}}_{\mathbf{B}_j^{(m)}}(x,t). \tag{7}$$

Reminding that $\lambda(\mathbf{B}^{(m)},x,t)_j$ denotes the softmax weight (Equation (5)) of point $\mathbf{B}_j^{(m)} = X_{\mathbf{b}_j^{(m)}}$ in the bootstrap sample $\mathbf{B}^{(m)} = X_{\mathbf{b}^{(m)}}$, *i.e.*, we have

$$\begin{aligned}\lambda(\mathbf{B}^{(m)},x,t)_j \ &= \ \frac{p^{\mathrm{cond}}_{\mathbf{B}_j^{(m)}}(x|t)}{\sum_{k=1}^N p^{\mathrm{cond}}_{\mathbf{B}_k^{(m)}}(x|t)} \\ &= \ \mathrm{softmax}\left(\left(\frac{-\left\|x - t\,\mathbf{B}_{j'}^{(m)}\right\|^2}{2(1-t)^2}\right)_{j'=1}^{N}\right)_j .\end{aligned}$$

Equation (7) above cannot be easily simplified, as the softmax's exponential cannot commute with the sum over the M bootstrap samples. However, the formula highlights that, at every position (x,t), the stochastic bagging field $u^{SBB}(x,t)$ is a convex combination of the conditional velocity fields $u^{\mathrm{cond}}(x,t)$, the weight differing from the original weights $\lambda(X,x,t)$ of Equation (5).

In fact, while not easy to compute, even the ideal bagging form u^{BCFF} can also be viewed as a convex combination of conditional velocities. Thus, for all (x,t), there exists a vector of weights $\lambda^{Bag}(X,x,t)$ such that

$$u^{BCFF}(x,t) \;=\; u^{SBB}_{\infty}(x,t) \;=\; \sum_{i=1}^{N} \lambda^{Bag}(X,x,t)_i \; u^{\text{cond}}_{X_i}(x,t).$$

3.3 Implementing a fast ideal BCFF

Leveraging the insights on the generation phases of diffusion, CFM (and EFM) [3,2], we propose an efficient approach for BCFF, *i.e.*, bagging the closed-form flow. Indeed, the conditional flow matching objective has been shown to have very low stochasticity in the late generation process (after some value of t). The time of this "late" phase has been shown to grow with the dimensionality d of the space where dataset points live. Typically, when $t > 0.2$, the objective exhibits almost no stochasticity for small images like CIFAR-10.

In this section, we show how bagging impacts this late-generation regime and, more importantly, how ideal bagging can be expressed analytically, yielding a closed-form expression for bagging the existing closed-form flow (CFF). We then argue that, for early generation times (low values of t), bagging has almost no impact and that the closed-form itself can be used and even simplified.

Late phase of the generation process. The absence of stochasticity in the last phase of generation corresponds to a weight vector that is very close to a one-hot vector, where all but one value are nearly zero. Actually, in this regime, for some location (x,t), there exists an index i such that $p^{\text{cond}}_{X_i}(x|t) \gg p^{\text{cond}}_{X_j}(x|t), \forall j \neq i$ and thus, numerically, $\lambda(X,x,t)_i = 1$ and $u^{\star}_X(x,t) = u^{\text{cond}}_{X_i}(x,t)$. Since the normal density is a decreasing function of the distance (between x and tX_i), the index of interest corresponds to the nearest neighbor of $t^{-1}x$.

We now fix some values (x,t) and denote by S the dataset sorted by decreasing $\lambda(X,x,t)$. We also denote by s the corresponding permutation, *i.e.*, the point indices, so that $S_i = X_{s(i)}$. With this notation, S_1 is the closest point to $t^{-1}x$, and S_2 the second closest, and so on. Supposing that, in this phase, at (x,t) we have $p^{\text{cond}}_{S_1}(x|t) \gg p^{\text{cond}}_{S_2}(x|t)$, then we can split the expectation over all bootstrap samples into those that contain S_1 and those that do not. Formally, we have *(note we omit (x,t) for readability, but all u depend on it)*

$$\begin{aligned} u^{BCFF} \;&=\; \mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\star}_{X_{\mathbf{b}}}\right] \\ &=\; p^{in} \cdot \underbrace{\mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\star}_{X_{\mathbf{b}}} \middle| s(1) \in \mathbf{b}\right]}_{E_1} + (1-p^{in}) \cdot \underbrace{\mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\star}_{X_{\mathbf{b}}} \middle| s(1) \notin \mathbf{b}\right]}_{E_2}, \end{aligned}$$

where the probability of having any point of interest in a bootstrap sample is $p^{in} = 1-(1-\frac{1}{N})^N = 1-(\frac{N-1}{N})^N$. In the context of bagging, this value is usually approximated [11] as $p^{in} = 0.632$.

When the nearest neighbor to $t^{-1}x$ is in the bootstrap sample, *i.e.*, $s(1) \in \mathbf{b}$, we have $\hat{u}^{\star}_{X_{\mathbf{b}}} = u^{\text{cond}}_{X_{s(1)}}$ and the expectation E_1 simplifies. We have

$$\begin{aligned} u^{BCFF}(x,t) &= p^{in} \cdot \mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\text{cond}}_{X_{s(1)}} \middle| s(1) \in \mathbf{b}\right] + (1-p^{in}) \cdot \mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\star}_{X_{\mathbf{b}}} \middle| s(1) \notin \mathbf{b}\right] \\ &= p^{in} \cdot u^{\text{cond}}_{X_{s(1)}} + (1-p^{in}) \cdot \mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\star}_{X_{\mathbf{b}}} \middle| s(1) \notin \mathbf{b}\right]. \end{aligned}$$

The weight of the most influential point is thus

$$\lambda^{Bag,late}(X,x,t)_{s(1)} = p^{in}. \tag{8}$$

Following a similar reasoning, we decompose the rightmost expectation E_2 as

$$E_2 = p_2^{in} \cdot \mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\star}_{X_{\mathbf{b}}} \middle|^{s(1)\notin\mathbf{b}}_{s(2)\in\mathbf{b}}\right] + (1-p_2^{in}) \cdot \mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\star}_{X_{\mathbf{b}}} \middle|^{s(1)\notin\mathbf{b}}_{s(2)\notin\mathbf{b}}\right], \tag{9}$$

where $p_2^{in} \approx p^{in}$ (for large N) is the probability that S_2 is in a bootstrap sample, knowing that S_1 is not. More precisely, we exactly have $p_2^{in} = 1 - \left(\frac{N-2}{N-1}\right)^N$. Assuming $p^{\text{cond}}_{S_2} \gg p^{\text{cond}}_{S_3}$, the left part of Equation (9) becomes dominated by S_2 and the expectation $\mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\star}_{X_{\mathbf{b}}} \middle|^{s(1)\notin\mathbf{b}}_{s(2)\in\mathbf{b}}\right]$ simplifies to $u^{\text{cond}}_{X_s(2)}$, yielding

$$E_2 \;=\; p_2^{in} \cdot u^{\text{cond}}_{X_s(2)} + (1-p_2^{in}) \cdot \mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^{\star}_{X_{\mathbf{b}}} \middle|^{s(1)\notin\mathbf{b}}_{s(2)\notin\mathbf{b}}\right]$$

Then, the weight for the second most influential point becomes

$$\lambda^{Bag,late}(X,x,t)_{s(2)} \;=\; p_2^{in} \cdot (1-p^{in}) \;=\; \left(1 - \left(\frac{N-2}{N-1}\right)^N\right)\left(\frac{N-1}{N}\right)^N$$

Following the same reasoning recursively, in this late regime, we can rewrite

$$\begin{aligned} u^{BCFF}(x,t) &= \sum_{i=1}^{N} p_i^{in} \cdot \left(\prod_{j=1}^{i-1} 1 - p_j^{in}\right) \cdot u^{\text{cond}}_{X_{s(i)}}(x,t), \\ &\text{with} \;\; p_i^{in} = 1 - \left(\frac{N-i}{N-i+1}\right)^N. \end{aligned}$$

Thus, we obtain the effective weights of $s(i)$:

$$\begin{aligned} \lambda^{Bag,late}(X,x,t)_{s(i)} &= p_i^{in} \prod_{j=1}^{i-1}(1-p_j^{in}) \\ &= \left(1 - \left(\frac{N-1}{N}\right)^N\right) \cdot \prod_{j=1}^{i-1}\left(\frac{N-j}{N-j+1}\right)^N \qquad (10) \\ &= \left(1 - \left(\frac{N-1}{N}\right)^N\right) \cdot \left(\frac{N-i+1}{N}\right)^N \qquad (11) \end{aligned}$$

In practice, $p_1^{in} \approx 0.632$ and p_j^{in} values are increasing (with j), so the cumulated product decays extremely fast, putting non (almost-)zero weights to at most 4 or 5 points.

Early phase of the generation process. The early phase of the generation process can be decomposed into two regimes: *(i)* an initial regime where all training points contribute (almost) equally to the velocity field, followed by *(ii)* a cluster-based regime where the points belonging to a cluster contribute almost equally, while all other points have no more influence. We argue that bagging has very little impact in these regimes (compared to the late phase), unless the clusters contain only a few points.

Both regimes can be abstracted by considering a weight vector λ in which a total mass $1-\varepsilon \approx 1$ is concentrated on L entries. More precisely, we assume that L entries have weight $\frac{1-\varepsilon}{L}$, while the remaining $N-L$ entries share the remaining mass, each having weight $\frac{\varepsilon}{N-L}$.

Under these assumptions, as in the previous section, we break the expectation into two cases. Either the bootstrap sample contains at least one of the L influential points, or it contains none of them. In the first case, the least-influential points can be considered negligible compared to the influential point(s). In both cases, we use the symmetry argument (uniform weights) to simplify the expectations. The probability that a random bootstrap sample contains none of the L influential points is $p^{none} = \left(\frac{N-L}{N}\right)^N$. Under the above assumptions, using the sorting permutation s as before (in this case it puts the L influential points at the beginning), we can write *(omitting (x,t))*

$$u^{BCFF} \;=\; (1-p^{none}) \cdot \underbrace{\mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^\star_{X_\mathbf{b}} \middle|^{\exists k\le K}_{s(k)\in\mathbf{b}}\right]}_{E_3} + p^{none} \cdot \underbrace{\mathbb{E}_{\mathbf{b}\sim\mathcal{B}}\left[u^\star_{X_\mathbf{b}} \middle|^{\forall k\le K}_{s(k)\notin\mathbf{b}}\right]}_{E_4}. \quad (12)$$

Here, the first expectation E_3 is over bootstrap samples that include at least one influential point. For any such bootstrap sample, the actual softmax weight of the least-influential points $(\lambda(X_b,x,t)_{k'>K})$ remains small, as these points compete with at least one influential point. Using a symmetry argument (within the influential points and within the least-influential points), in a coarse approximation, we can consider that, on average, the relative weight will be mostly preserved and thus recover the original closed-form:

$$E_3 \;\approx\; \sum_{k=1}^{L} \frac{1-\varepsilon}{L} u^{\text{cond}}_{X_{s(k)}} + \sum_{k'=L+1}^{N} \frac{\varepsilon}{N-L} u^{\text{cond}}_{X_{s(k')}} = \sum_{i=1}^{N} \lambda_{s(i)} u^{\text{cond}}_{X_{s(i)}}. \quad (13)$$

The conditional expectation E_4 corresponds to bootstrap samples with only equally non-influential points. By symmetry, it simplifies to

$$E_4 \;=\; \sum_{k'=L+1}^{N} \frac{1}{N-L} u^{\text{cond}}_{X_{s(k')}}. \quad (14)$$

Equations (12), (13) and (14) give

$$u^{BCFF} \;\approx\; (1-p^{none}) \cdot \sum_{i=1}^{N} \lambda_{s(i)} u^{\text{cond}}_{X_{s(i)}} + p^{none} \cdot \sum_{k'=L+1}^{N} \frac{1}{N-L} u^{\text{cond}}_{X_{s(k')}}.$$

This shows that bagging removes, from the usual closed-form, a total mass p^{none} (mostly) from the influential point and redistributes it among the least-influential points. Since $p^{none} = \left(\frac{N-L}{N}\right)^N$, it vanishes very fast with L. For instance, with $L = 4$, it is below approximately 10^{-3} for any N.

Therefore, in the low t regime where several points share most of the probability mass, the bagging correction is negligible. We thus expect bagging to mainly affect the late phase of the generation process, where the weight vector λ becomes highly concentrated.

Efficient BCFF using Nearest Neighbors. Based on the previous conclusions, we propose the following fast implementation for BCFF. Choosing a threshold τ, typically 0.2, for $t \leq \tau$, we use CFF, the normal closed-form. For $t > \tau$, to compute the flow at position (x, t), we query the dataset to get the K (typically 5) closest neighbors of $t^{-1}x$. We then use a weighted combination of their conditional velocity fields according to $\lambda^{Bag,late}$ from Equation (11).

4 Experiments

Our experiments aim at studying the behavior and the impact of applying and approximating bootstrap aggregating for closed-form flow methods. We are particularly interested in the memorization aspects. Results are condensed into information dense plots, more details being provided in supplementary material C

Protocol and Datasets. We expect bootstrap to be less impactful when the dataset is very densely sampled. We thus consider different base datasets and subsample them to produce training sets of varying sizes N. **2D-MOONS** corresponds to the two moons dataset, **MNIST** [21] is the usual 28×28 images of digits (fashion articles for **FMNIST**), **CIFAR-10** [19] corresponds to 32×32 images. We compare the baseline closed-form (CFF) with the proposed bagging approaches: the one based on sampling (SBB) and the one based on the nearest neighbors ordering (BCFF). For SBB, we vary the number of bootstrap samples M (each containing N points) used to approximate the expectation. The nearest neighbor bagging that we proposed to implement BCFF is designed for the late/final stage of the generation process. Therefore, we study the impact of the threshold τ on the time t, after which we use the proposed closed-form. Before this threshold, we apply the standard closed-form formulation.

4.1 Evaluation Metrics

Our analysis relies on four metrics designed to capture different facets of the impact of bagging.

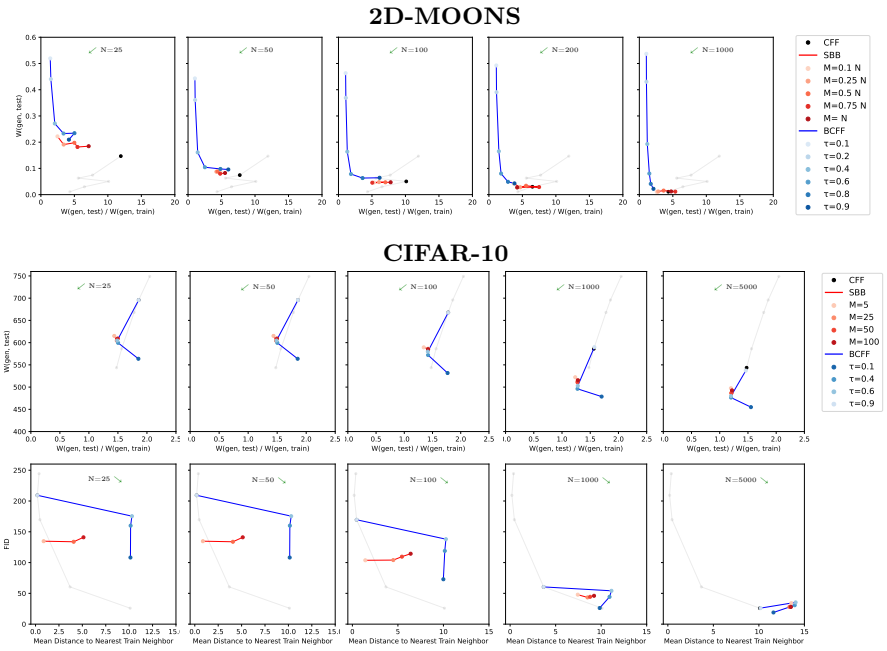

Fig. 2: Generation quality as a function of memorization. For each dataset and each training size N, the first row shows $W(gen, test)$ and the memorization ratio ${}^{W(gen,test)}/_{W(gen,train)}$: the best model being closer to the the lower left corner. For CIFAR-10, the second row shows the FID and the nearest training example distance, so the best model is at the bottom right. The green arrow (↙ or ↘) indicates the corner of interest. The metrics for the baseline CFF are in black, for SBB in red with varying bootstrap samples counts M, and the fast BCFF in blue (see runtimes in supplementary material C.2). To provide a visual anchor, we report in gray the metrics for CFF with all the N.

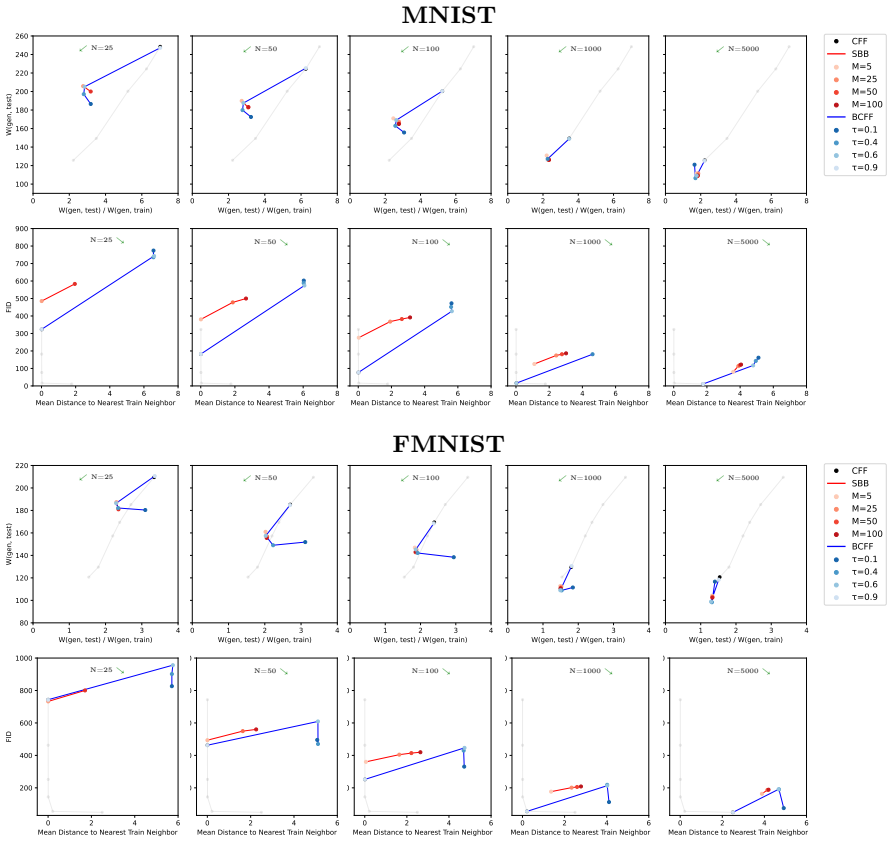

Fig. 3: Generation quality as a function of memorization. For each dataset and each training size N, the first row shows $W(gen, test)$ and the memorization ratio $W(gen,test)/W(gen,train)$: the best model being closer to the the lower left corner. The second row shows the FID and the nearest training example distance: so the best model is at the bottom right. The green arrow (↙ or ↘) indicates the corner of interest. The metrics for the baseline CFF are in black, for SBB in red with varying bootstrap samples counts M, and the fast BCFF in blue (see runtimes in supplementary material C.2). To provide a visual anchor, we report in gray the metrics for CFF with all the N.

Measures of generalization. We consider two measures of generalization. The first is the Wasserstein distance, denoted by $W(gen, test)$, between generated samples and some held-out test data. A good model should produce a low value of $W(gen, test)$. As the Wasserstein distance can be noisy for small samples, we typically use $10K$ samples for each of these, as we want to compare distributions. The second generalization measure, available only when an embedding space exists (so not for 2D-MOONS) is the FID (Frechet Inception Distance) between generated and test data. Often used in generative models, FID corresponds to an approximate Wasserstein distance computed in an embedding space obtained through supervised training.

Measures of memorization. We also consider two measures of memorization. A good generative model should avoid memorization, which corresponds to having a large distance to the nearest training example and a low memorization ratio. The first measure is the average (over generated points) of the distances between the point and its closest training point, as used in [2]. This metric is actually a measure of non-memorization: higher values correspond to lower memorization. We introduce a second memorization measure, *the memorization ratio*, defined as the ratio between $W(gen, test)$ and $W(gen, train)$, where $W(gen, train)$ is the Wasserstein distance between generated samples and the train set. Since the training may be smaller than the $10K$ samples used for the generation and the test set, this Wasserstein distance mechanically increases. Thus, a higher ratio indicates either a larger distance to the test set and/or a smaller distance to the training set, both corresponding to higher memorization. This metric, like the $W(gen, test)$ and the FID but unlike the nearest training sample distance, is a lower-is-better metric. However, avoiding memorization alone does not guarantee quality. This limitation applies to both memorization metrics.

Results and Analysis. In Figures 2 and 3, we report the four measures in a compact form. We propose two types of plots, both with a vertical axis measuring generation quality and the horizontal axis measuring memorization. The first type of graph shows $W(gen, test)$ and the memorization ratio $W(gen,test)/W(gen,train)$: the best model being closer to the lower left corner. The second type reports the FID and the nearest training example distance, so the best model is at the bottom right. Note that, to improve readability, the plotted values correspond to the average over multiple runs, each run using a different training set. Plots including variances are reported in supplementary material C.1.

Behavior on 2D-MOONS. In Figure 2, we observe, as expected, since the dataset is rather simple, no improvement in generalization (distance to the test) is observed when using bagging. However, bagging effectively reduces memorization. We observe that the closed-form BCFF method is more effective at reducing memorization than the sampling-based SBB method. Moreover, applying BCFF too early (before $\tau = 0.5$) in the generation is highly detrimental. This is consistent with the fact that the BCFF formula relies on assumptions that are only valid in the late generation phase.

Behavior on MNIST and FMNIST. In Figure 3, the first set of metric ($W(gen, test)$ and $\frac{W(gen,test)}{W(gen,train)}$) is very consistent across all values of N: using bagging is beneficial for both measures, meaning it improves generalization and reduces memorization. We also observe that BCFF, despite being computationally heavy, is slightly more beneficial than SBB. In addition, BCFF can be applied earlier in the generation process (*i.e.* for earlier t, for instance after $\tau = 0.4$), which is consistent with the observations in [2] that generation phases depend on the dimensionality of the space.

The second set of metrics (FID and nearest training point distance) paints a slightly different picture. We observe that bagging tends to degrade the FID, with SBB performing better than BCFF in terms of FID. BCFF still achieves the best non-memorization behavior but at the cost of worse FID. Interestingly, BCFF does not appear to benefit significantly from using $\tau = 0.4$.

Behavior on CIFAR-10. In Figure 2, the first set of metrics reinforces the conclusions observed previously: BCFF provides the best trade-off and can be applied as early as $\tau = 0.4$. Probably due to the increased complexity of the data, SBB with the number of samples tested, still worsens the memorization ratio $W(gen,test)/W(gen,train)$.

The second set of metrics paints a much more favorable picture for bagging. Both approaches improve both FID and memorization. SBB appears to require even more samples (than 100), although the improvements seem to lessen as the number increases. BCFF shows the best behavior in terms of memorization while achieving an FID very close to that of SBB (which is much more expensive).

5 Conclusion

In this work, we propose a bagging-based approach to mitigate memorization in closed-form flow matching models. Rather than training multiple models on bootstrap resamples of the dataset, which would be computationally expensive, we introduce a formulation that applies the bagging principle directly at the closed-form level by averaging velocity fields associated with resampled datasets. We instantiate this idea through two practical approaches: a sampling-based bootstrap (SBB) and a nearest-neighbor-based one (BCFF). Our experiments on datasets of increasing complexity show that bagging can reduce memorization while maintaining or improving generation quality.

We believe that several directions for future work naturally emerge from our contribution. From a practical perspective, while we have shown that bagging reduces memorization in the closed form, we did not study its impact in the case of flow matching (a model trained to fit this closed form). From a theoretical perspective, analyzing the relationship between bootstrap aggregating, variance reduction, and memorization in generative models could provide new insights into generalization mechanisms of modern generative models. In addition, while our experiments empirically show that BCFF improves generalization, as illustrated by the reduced Wasserstein distance between generated and test data, a theoretical

understanding of this behavior remains an open question. In particular, it would be interesting to study whether the probability path induced by BCFF is closer to the true data distribution than the empirical path induced by the training dataset.

Acknowledgments. This work has been partly funded by public grants from the French National Research Agency (ANR), namely the DATeS project (ANR-25-CE23-6035), the LSD project (ANR-25-CE23-6481) and the Famous project (ANR-23-CE23-0019).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Albergo, M.S., Vanden-Eijnden, E.: Building normalizing flows with stochastic interpolants. ICLR (2023)
2. Bertrand, Q., Gagneux, A., Massias, M., Emonet, R.: On the closed-form of flow matching: Generalization does not arise from target stochasticity. In: NeurIPS (2025)
3. Biroli, G., Bonnaire, T., de Bortoli, V., Mézard, M.: Dynamical regimes of diffusion models. Nature Communications **15**(1), 9957 (2024)
4. Bonnaire, T., Urfin, R., Biroli, G., Mezard, M.: Why diffusion models don't memorize: The role of implicit dynamical regularization in training. In: NeurIPS (2025)
5. Borsos, Z., Marinier, R., Vincent, D., Kharitonov, E., Pietquin, O., Sharifi, M., Roblek, D., Teboul, O., Grangier, D., Tagliasacchi, M., et al.: Audiolm: a language modeling approach to audio generation. IEEE/ACM transactions on audio, speech, and language processing **31**, 2523–2533 (2023)
6. Breiman, L.: Bagging predictors. Machine learning **24**(2), 123–140 (1996)
7. Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramer, F., Balle, B., Ippolito, D., Wallace, E.: Extracting training data from diffusion models. In: USENIX Security Symposium (USENIX Security 23) (2023)
8. Dar, S.U.H., Ghanaat, A., Kahmann, J., Ayx, I., Papavassiliu, T., Schoenberg, S.O., Engelhardt, S.: Investigating data memorization in 3d latent diffusion models for medical image synthesis. In: International Conference on Medical Image Computing and Computer-Assisted Intervention (2023)
9. Dieleman, S., Sartran, L., Roshannai, A., Savinov, N., Ganin, Y., Richemond, P.H., Doucet, A., Strudel, R., Dyer, C., Durkan, C., et al.: Continuous diffusion for categorical data. arXiv preprint arXiv:2211.15089 (2022)
10. Dietterich, T.G.: Ensemble methods in machine learning. In: International workshop on multiple classifier systems (2000)
11. Efron, B., Tibshirani, R.: Improvements on cross-validation: the 632+ bootstrap method. Journal of the American Statistical Association **92**(438), 548–560 (1997)
12. Gagneux, A., Martin, S., Emonet, R., Bertrand, Q., Massias, M.: A visual dive into conditional flow matching. In: ICLR Blogpost (2025)
13. Gao, R., Hoogeboom, E., Heek, J., de Bortoli, V., Murphy, K., Salimans, T.: Diffusion meets flow matching: Two sides of the same coin. ICLR Blogpost (2025)
14. Gao, W., Li, M.: How do flow matching models memorize and generalize in sample data subspaces? arXiv preprint arXiv:2410.23594 (2024)
15. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. NeurIPS (2020)

16. Kadkhodaie, Z., Guth, F., Simoncelli, E.P., Mallat, S.: Generalization in diffusion models arises from geometry-adaptive harmonic representations. ICLR (2024)
17. Kamb, M., Ganguli, S.: An analytic theory of creativity in convolutional diffusion models. ICML (2025)
18. Khanna, S., Kharbanda, S., Li, S., Varma, H., Wang, E., Birnbaum, S., Luo, Z., Miraoui, Y., Palrecha, A., Ermon, S., et al.: Mercury: Ultra-fast language models based on diffusion. arXiv preprint arXiv:2506.17298 (2025)
19. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009), Toronto, ON, Canada
20. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine learning **51**(2), 181–207 (2003)
21. LeCun, Y., Cortes, C., Burges, C.: THE MNIST DATASET of handwritten digits (1998), `http://yann.lecun.com/exdb/mnist/`
22. Li, S., Chen, S., Li, Q.: A good score does not lead to a good generative model. arXiv preprint arXiv:2401.04856 (2024)
23. Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. ICLR (2023)
24. Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R.T., Lopez-Paz, D., Ben-Hamu, H., Gat, I.: Flow matching guide and code. arXiv preprint arXiv:2412.06264 (2024)
25. Liu, X., Gong, C., Liu, Q.: Flow straight and fast: Learning to generate and transfer data with rectified flow. ICLR (2023)
26. Lukoianov, A., Yuan, C., Solomon, J., Sitzmann, V.: Locality in image diffusion models emerges from data statistics. In: NeurIPS (2025)
27. Niedoba, M., Zwartsenberg, B., Murphy, K., Wood, F.: Towards a mechanistic explanation of diffusion model generalization. ICML (2025)
28. Ning, M., Sangineto, E., Porrello, A., Calderara, S., Cucchiara, R.: Input perturbation reduces exposure bias in diffusion models. ICML (2023)
29. Scarvelis, C., de Ocáriz, H.S.B., Solomon, J.: Closed-form diffusion models. TMLR (2025)
30. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: ICML (2015)
31. Somepalli, G., Singla, V., Goldblum, M., Geiping, J., Goldstein, T.: Diffusion art or digital forgery? investigating data replication in diffusion models. In: CVPR (2023)
32. Somepalli, G., Singla, V., Goldblum, M., Geiping, J., Goldstein, T.: Understanding and mitigating copying in diffusion models. NeurIPS (2023)
33. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. ICLR (2021)
34. Stability AI: `https://stability.ai/stablediffusion` (2023)
35. Vastola, J.J.: Generalization through variance: how noise shapes inductive biases in diffusion models. ICLR (2025)
36. Villegas, R., Babaeizadeh, M., Kindermans, P.J., Moraldo, H., Zhang, H., Saffar, M.T., Castro, S., Kunze, J., Erhan, D.: Phenaki: Variable length video generation from open domain textual descriptions. In: ICLR (2022)

Supplementary Material for
Mitigating Memorization in Closed-Form Flow Matching with Bootstrap Aggregating

A Notations

Table 1: Summary of the main notations used throughout the paper.

Notation	Meaning
$m \in [1, M]$	Index in a stochastic approximation (of an expectation)
$i \in [1, N]$	Index in the training dataset
$j \in [1, N]$	Index in a bootstrap sample
$X = (X_i)_{i=1}^N \in (\mathbb{R}^d)^N$	Training dataset X is a sequence/set of N points
$b = (b_j)_{j=1}^N \in ([1, N])^N$	A bootstrap sample is a list of N indices (with replacement)
$\mathcal{B} = \mathrm{Cat}(([1, N])^N)$	Categorical uniform distribution of all possible bootstrap samples (of size N on a set of size N)
$X_b = (X_{b_j})_{j=1}^N \in (\mathbb{R}^d)^N$	Bootstrap sample taken from X
$\lambda(X, x, t) = (\lambda(X, x, t)_i)_{i=1}^N$	Weight vector (for points of X) in the CFM closed-form at (x, t)
$s(X, x, t)$	Permutation sorting points of X by decreasing $\lambda(X, x, t)$ values
$X_{s(X,x,t)_1}$	Dataset point with the highest λ (closest) *w.r.t.* position (x, t)
$X_{s(X,x,t)}$	Dataset resorted by decreasing λ (closest) *w.r.t.* position (x, t)
p_{data}	Unknown target data distribution
p_0	Source distribution, typically a unit gaussian $\mathcal{N}(0, I_d)$
$p^\star(x\|t)$	Probability path corresponding to the unknown p_{data}
$\hat{p}_{data} = \frac{1}{N}\sum_{i=1}^N \delta_{X_i}$	Training data distribution
$\hat{p}^\star(x\|t)$	Probability path corresponding to the training dataset
$p_X^\star(x\|t)$	Probability path corresponding to any dataset X
$u_X^\star(x, t)$	Optimal closed-form velocity field for any dataset X
$p_{X_i}^{\mathrm{cond}}(x\|t)$	Conditional probability path conditioned on X_i
$u_{X_i}^{\mathrm{cond}}(x, t)$	Conditional velocity field conditioned on X_i

B Extended Discussions on Bagging and Flow Matching

B.1 Varying the Bootstrap Sample Size

In most bagging applications, bootstrap samples are drawn with the same size as the original training set.

While at first sight varying this number can be useful, two realizations are important. First, increasing the bootstrap size too much brings the bootstrap samples very close to the original sample. In extreme cases, each point appears several times in all bootstrap samples, resulting in a velocity field very close to the one computed from the original training set, thus eliminating the variance effect of bagging. Second, decreasing the bootstrap size too much increases the variance. In this situation, each bootstrap sample contains too little information

about the data, which can lead to unstable velocity fields. Although averaging still reduces variance, the individual velocity field may become too different to provide a meaningful improvement.

In practice, it is thus important to use a bootstrap sample size able to provide a reasonable trade-off between diversity of the samples and stability of the individual velocity fields (in practice N is reasonable).

B.2 Bagging Trained Conditional Flow Matching Models

As commonly done in bagging, it is possible to apply bagging at the level of trained CFM models. The idea consists of drawing M bootstrap samples, then fully learning a CFM model on each of these samples (CFM or EFM), and finally averaging their velocity field prediction during the generation phase. While this approach could provide some gains, it is computationally expensive both at training and generation time, typically requiring approximately M times the cost of a single CFM. Moreover, such models incorporate different sources of implicit regularization, such as the inductive biases or stochastic optimization through mini-batches. This suggests that the usefulness of bagging at the level of trained models is limited compared to what we propose in our contribution (*i.e.*, compared to bagging at the closed-form level).

B.3 Is Minibatch CFM/EFM Already Doing Bagging?

Most models are trained with stochastic versions of gradient descent algorithms. As such, there is already some sampling when computing, *e.g.*, the target in EFM. While one might argue that minibatch EFM already implements bagging, it is not the case, especially at high t values. Indeed, the unbiased way (proposed in [2]) of approximating the closed-form of flow matching requires always including the "current" points in the minibatch (used to approximate the closed-form).

For high t values, in EFM the closest point will almost always be in the minibatch and thus will receive a weight of around 1. This behavior differs significantly from our bagging solution, which assigns a weight of approximately 0.632 to the closest point. As a consequence, the bagging flow differs from minibatch EFM, reducing memorization beyond minibatching.

B.4 Bagging the Probability Path

In CFM, $\hat{p}^*$ is the expectation of $p_{X_i}^{cond}$ values. It is technically not fitted/estimated as such, so it does not require bagging. Also, bagging this formula does not change anything, as the formula is linear and thus the expectations commute.

$$
\begin{aligned}
\hat{p}_M^{SBB}(x|t) &= \frac{1}{M}\sum_{m=1}^{M} \hat{p}_{\mathbf{B}^{(m)}}(x|t) \\
&= \frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{j=1}^{N} p^{\text{cond}}_{\mathbf{B}_j^{(m)}}(x|t) \\
&= \frac{1}{N}\sum_{j=1}^{N}\frac{1}{M}\sum_{m=1}^{M} p^{\text{cond}}_{\mathbf{B}_j^{(m)}}(x|t) \\
&= \frac{1}{N}\sum_{j=1}^{N}\frac{1}{N}\sum_{i=1}^{N} p^{\text{cond}}_{X_i}(x|t) \\
&= \frac{1}{N}\sum_{i=1}^{N} p^{\text{cond}}_{X_i}(x|t) \\
&= \hat{p}(x|t)
\end{aligned}
$$

B.5 Probability Path Corresponding to u^{BCFF}

The velocity field induced by bagging u^{BCFF} (or its sampling-based version u^{SBB}) differs from the closed-form solution of the flow matching problem. The continuity equation (which is a key point to obtaining the closed-form expression of the flow in CFM) also applies to the bagged velocity fields. Thus, there exists a probability path p^{BCFF} associated with the velocity field u^{BCFF}. We keep as future work the possible derivation of an expression for p^{BCFF}. While empirically shown with experiments that show reduced Wasserstein between test and generated data, the possible proof that p^{BCFF} is closer to the real unknown $p^\star$ (closer than the empirical one $\hat{p}^\star$ is) also constitutes future work.

C Additional Experimental Details

C.1 Extended results

Figures 4 and 5 show variances on each dataset, with SBB tending to be more stable as M increases.

C.2 Runtimes

Figure 6 shows the generation time of 10000 samples. SBB is significantly slower than CFF, but its approximation with BCFF runs on the contrary faster than CFF, in particular with τ fixed at 0.2.

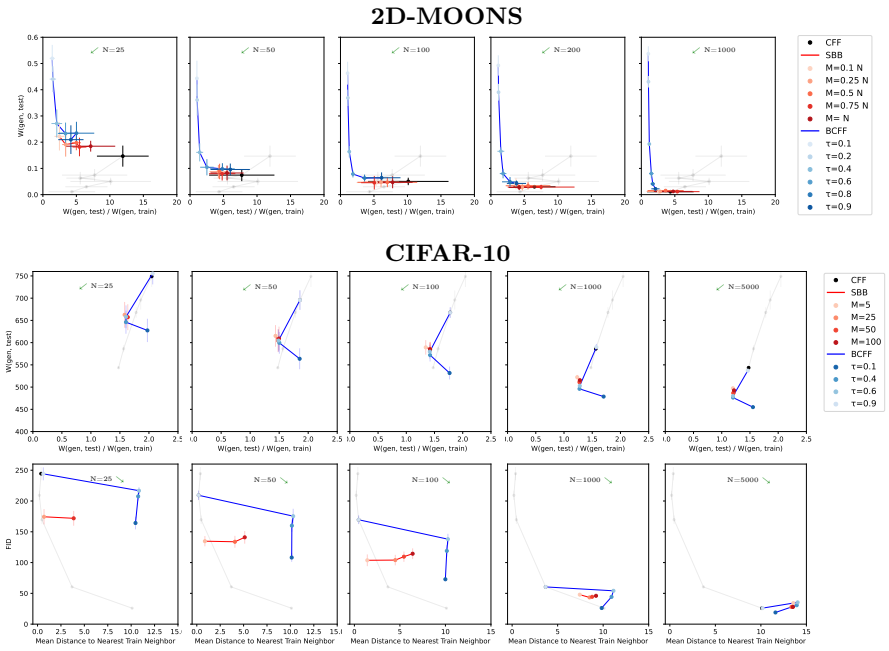

Fig. 4: Generation quality as a function of memorization. For each dataset and each training size N, the first line of plots shows $W(gen, test)$ and the memorization ratio $W(gen, test)/W(gen, train)$: the best model being closer to the the lower left corner. For CIFAR, the second line shows the FID (we have no embedding for FMNIST and so we report no FID) and the nearest training example distance: so the best model is at the bottom right. The green arrow (↙ or ↘) indicates the corner of interest. The metrics for the baseline CFF are in black, for SBB in red with varying bootstrap samples counts M and the fast BCFF in blue. To provide a visual anchor, we report in gray the metrics for CFF with all the N.

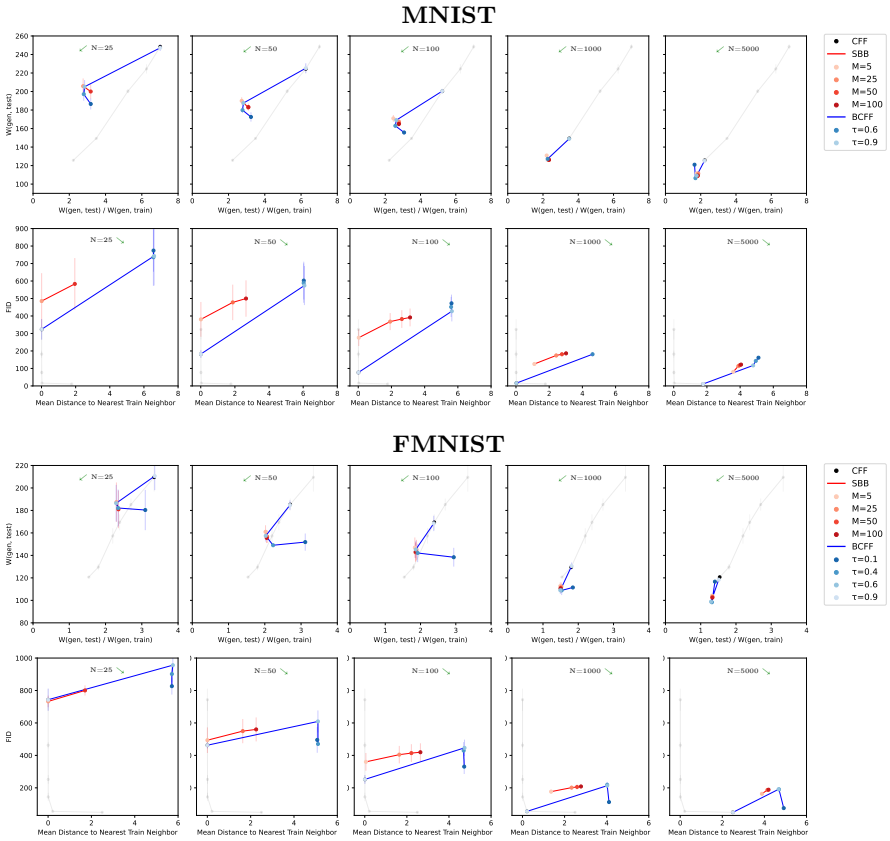

Fig. 5: Generation quality as a function of memorization. For each dataset and each training size N, the first line of plots shows $W(gen, test)$ and the memorization ratio $W(gen, test)/W(gen, train)$: the best model being closer to the the lower left corner. The second line shows the FID (we have no embedding for FMNIST and so we report no FID) and the nearest training example distance: so the best model is at the bottom right. The green arrow (↙ or ↘) indicates the corner of interest. The metrics for the baseline CFF are in black, for SBB in red with varying bootstrap samples counts M and the fast BCFF in blue. To provide a visual anchor, we report in gray the metrics for CFF with all the N.

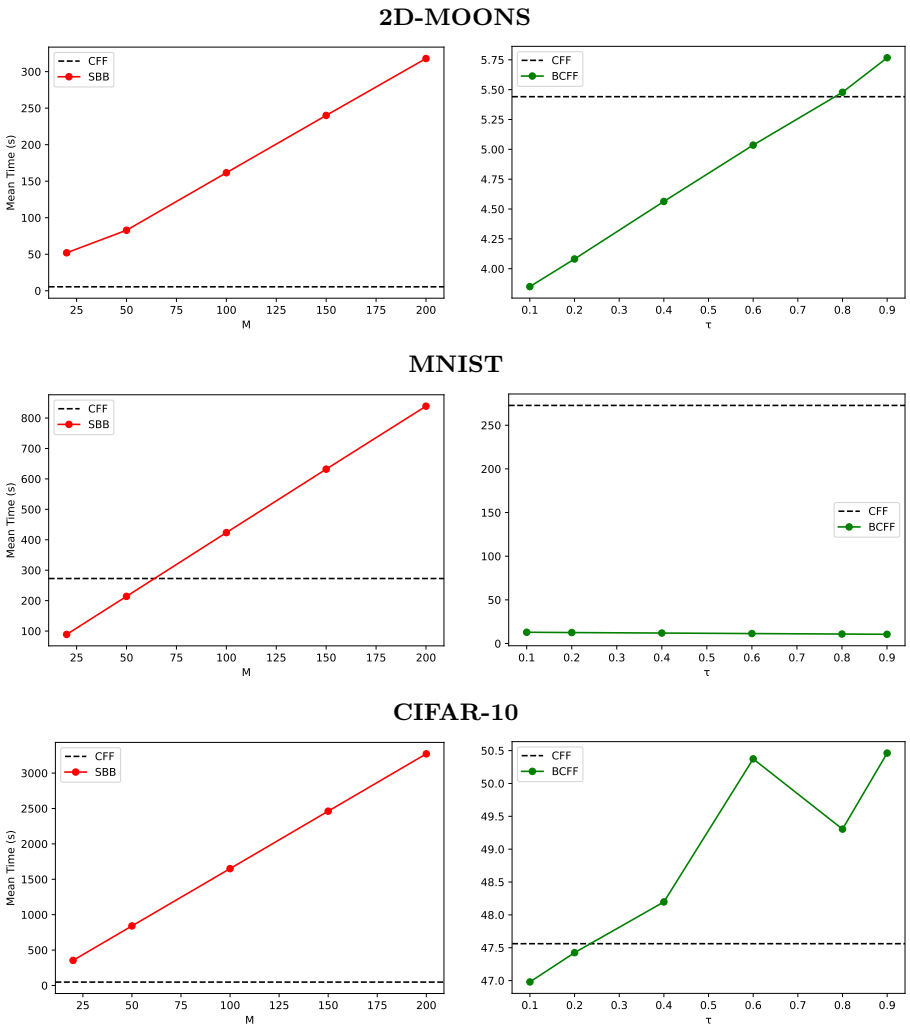

Fig. 6: Generation time as a function of the number of bootstrap samples M used in SBB, or the threshold τ at which we switch from CFF to BCFF.