Conformal prediction for time-series and flow-based generative models

Chen Xu

January 7, 2025

School of Industrial and Systems Engineering,
Georgia Institute of Technology

PhD Thesis Defense

Defense overview

- Part I: Conformal prediction for time-series

  → Quantify uncertainty of point estimators by building prediction intervals (Chapter 1) or regions (Chapter 2).

Defense overview

- Part I: Conformal prediction for time-series

  → Quantify uncertainty of point estimators by building prediction intervals (Chapter 1) or regions (Chapter 2).

- Part II: Flow-based generative models

  → Given samples from a high-dimensional distribution, model the distribution for sampling and likelihood estimation using MLE training (Chapter 3) or flow matching (Chapter 4).

Defense overview

- Part I: Conformal prediction for time-series

  → Quantify uncertainty of point estimators by building prediction intervals (Chapter 1) or regions (Chapter 2).
- Part II: Flow-based generative models

  → Given samples from a high-dimensional distribution, model the distribution for sampling and likelihood estimation using MLE training (Chapter 3) or flow matching (Chapter 4).
- Layout: problem motivation, mathematical setup, proposed method, results.

Outline

- Part I: Conformal prediction for time-series
- Part II: Flow-based generative models
- Future work: Connecting Parts I & II

Part I: Motivation

- Question
  - A black-box model $\hat{f}$ predicts the financial return $\hat{Y}$ to be 5%.
  - With high probability, where would the true return Y lie?
- One way is to build prediction regions around the predictions.

Part I: Motivation

- Question
  - A black-box model $\hat{f}$ predicts the financial return $\hat{Y}$ to be 5%.
  - With high probability, where would the true return Y lie?
- One way is to build prediction regions around the predictions.
- In special cases (e.g., linear regression with i.i.d. data), prediction regions have closed-form expressions.
- Yet, we want to generalize this to an arbitrary predictor $\hat{f}$, also when data are time series.

Part I: Setup

- Data: $(X_i, Y_i), X_i \in \mathbb{R}^d, Y_i \in \mathbb{R}^p$.
  X_i is feature vector and Y_i is (multi-variate) response.
  In time-series, X_i can be the history of Y_i.

Part I: Setup

- Data: $(X_i, Y_i), X_i \in \mathbb{R}^d, Y_i \in \mathbb{R}^p$.
  X_i is feature vector and Y_i is (multi-variate) response.
  In time-series, X_i can be the history of Y_i.
- Given a significance level $\alpha \in (0, 1)$ and a new feature X_t, produce a prediction region $C(X_t, \alpha)$:

$$\mathbb{P}(Y_t \in C(X_t, \alpha)) \geq 1 - \alpha. \tag{1}$$

Part I: Setup

- Data: $(X_i, Y_i), X_i \in \mathbb{R}^d, Y_i \in \mathbb{R}^p$.
  X_i is feature vector and Y_i is (multi-variate) response.
  In time-series, X_i can be the history of Y_i.
- Given a significance level $\alpha \in (0, 1)$ and a new feature X_t, produce a prediction region $C(X_t, \alpha)$:

$$\mathbb{P}(Y_t \in C(X_t, \alpha)) \geq 1 - \alpha. \tag{1}$$

- Example ($p = 1$): with 95% probability (i.e., $\alpha = 0.05$), true financial return (i.e., Y_t) is within the constructed interval (i.e., $C(X_t, \alpha)$).

Part I: Conformal prediction

- In a nutshell, conformal prediction first defines non-conformity scores $s(X, Y, \hat{f})$ (e.g., $s = \|Y - \hat{f}(X)\|_2$). Then,

$$C(X_t, \alpha) = \{y \in \mathbb{R}^p : Q_{\alpha/2}(\mathcal{S}) \leq s(X_t, y, \hat{f}) \leq Q_{1-\alpha/2}(\mathcal{S})\}. \quad (1)$$

Part I: Conformal prediction

- In a nutshell, conformal prediction first defines non-conformity scores $s(X, Y, \hat{f})$ (e.g., $s = \|Y - \hat{f}(X)\|_2$). Then,

$$C(X_t, \alpha) = \{y \in \mathbb{R}^p : Q_{\alpha/2}(\mathcal{S}) \leq s(X_t, y, \hat{f}) \leq Q_{1-\alpha/2}(\mathcal{S})\}. \quad (1)$$

- $\mathcal{S} = \{s(X_i, Y_i, \hat{f})\}$ is computed on a "hold-out" set $\{(X_i, Y_i)\}$ *different from the training set* of $\hat{f}$.
- Q_β for $\beta \in [0, 1]$ is the *empirical quantile* function.

Part I: Conformal prediction

- In a nutshell, conformal prediction first defines non-conformity scores $s(X, Y, \hat{f})$ (e.g., $s = \|Y - \hat{f}(X)\|_2$). Then,

$$C(X_t, \alpha) = \{y \in \mathbb{R}^p : Q_{\alpha/2}(\mathcal{S}) \leq s(X_t, y, \hat{f}) \leq Q_{1-\alpha/2}(\mathcal{S})\}. \quad (1)$$

  - $\mathcal{S} = \{s(X_i, Y_i, \hat{f})\}$ is computed on a "hold-out" set $\{(X_i, Y_i)\}$ *different from the training set* of $\hat{f}$.
  - Q_β for $\beta \in [0, 1]$ is the *empirical quantile* function.

- Thus, (1) includes all y with scores "conforming" to the past.
- The idea mainly originated in [Papadopoulos et al., 2007]. More comprehensive review can be found in [Angelopoulos and Bates, 2023].

Part I: Conformal prediction (cont.)

$$C(X_t, \alpha) = \{y \in \mathbb{R}^p : Q_{\alpha/2}(\mathcal{S}) \leq s(X_t, y, \hat{f}) \leq Q_{1-\alpha/2}(\mathcal{S})\}. \quad (1)$$

Prediction regions using (1) are

1. distribution-free in $Y|X$.
2. model-free in $\hat{f}$.
3. theoretically valid $1-\alpha$ coverage when $\{(X_i, Y_i)\}_{i\geq 1}$ are exchangeable (e.g., $(Z_1, Z_2) \stackrel{d}{=} (Z_2, Z_1)$).

Part I: Conformal prediction (cont.)

$$C(X_t, \alpha) = \{y \in \mathbb{R}^p : Q_{\alpha/2}(\mathcal{S}) \leq s(X_t, y, \hat{f}) \leq Q_{1-\alpha/2}(\mathcal{S})\}. \quad (1)$$

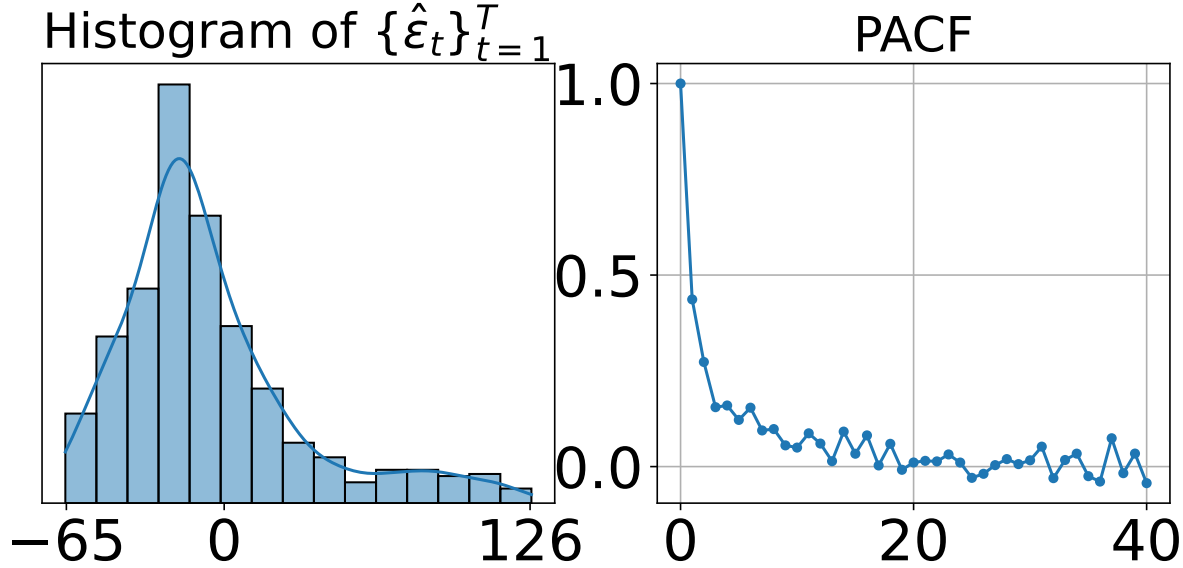
Prediction regions using (1) are

1. distribution-free in $Y|X$.
2. model-free in $\hat{f}$.
3. theoretically valid $1-\alpha$ coverage when $\{(X_i, Y_i)\}_{i\geq 1}$ are exchangeable (e.g., $(Z_1, Z_2) \stackrel{d}{=} (Z_2, Z_1)$).

Question: how to extend CP to time series, which are rarely exchangeable?

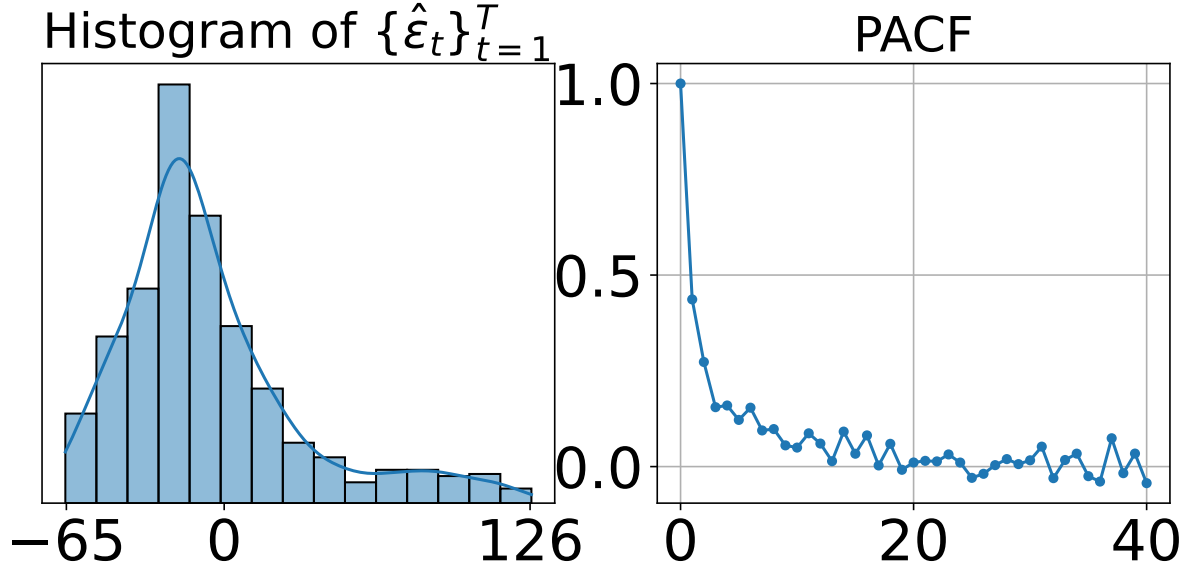
Part I: Proposed solution

- Exchangeability is rare for time series:



Part I: Proposed solution

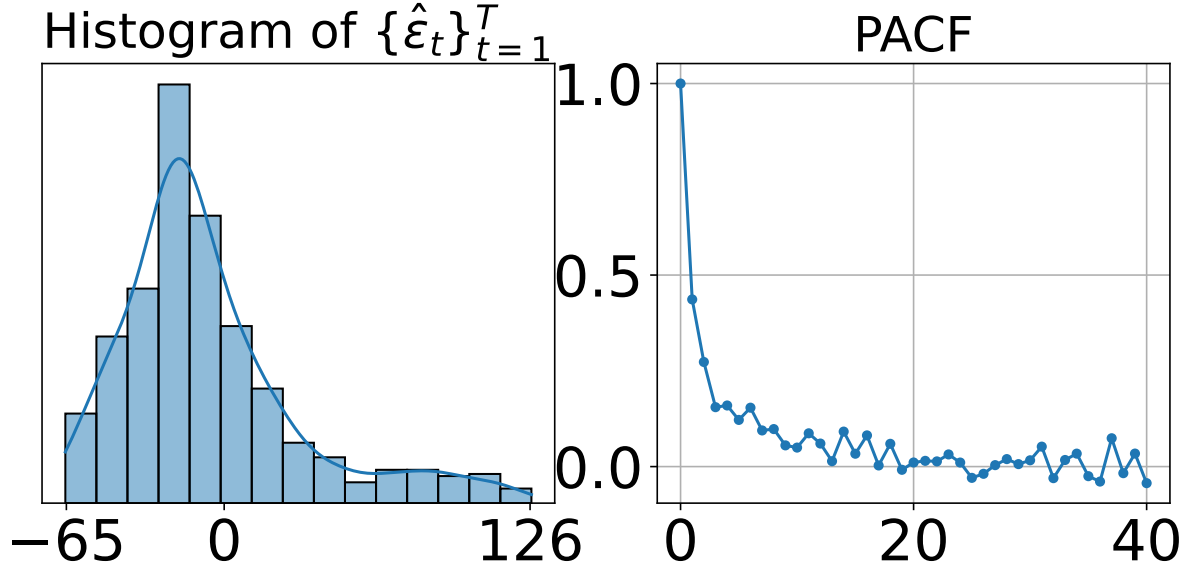
- Exchangeability is rare for time series:



- Our solution:
  1. Sequentially update $\mathcal{S}$ to be $\mathcal{S}_t$.
  2. Consider trainable conditional quantile models $\hat{\mathcal{Q}}$ (e.g., quantile random forest) besides the empirical quantile Q.

Part I: Proposed solution

- Exchangeability is rare for time series:



- Our solution:
  1. Sequentially update $\mathcal{S}$ to be $\mathcal{S}_t$.
  2. Consider trainable conditional quantile models $\hat{\mathcal{Q}}$ (e.g., quantile random forest) besides the empirical quantile Q.

$$C(X_t, \alpha) = \{y \in \mathbb{R}^p : \hat{\mathcal{Q}}_{\alpha/2}(\mathcal{S}_t) \leq s(X_t, y, \hat{f}) \leq \hat{\mathcal{Q}}_{1-\alpha/2}(\mathcal{S}_t)\}. \quad (1)$$

Part I: Proposed solution (cont.)

- When $p = 1$ ($Y_i \in \mathbb{R}$) and $s(X_t, y, \hat{f}) = y - \hat{f}(X_t)$, the prediction residual, we have prediction intervals

$$C(X_t, \alpha) = [\hat{f}(X_t) + \hat{\mathcal{Q}}_{\alpha/2}(\mathcal{S}_t), \hat{f}(X_t) + \hat{\mathcal{Q}}_{1-\alpha/2}(\mathcal{S}_t)] \quad (1)$$

Part I: Proposed solution (cont.)

- When $p = 1$ ($Y_i \in \mathbb{R}$) and $s(X_t, y, \hat{f}) = y - \hat{f}(X_t)$, the prediction residual, we have prediction intervals

$$C(X_t, \alpha) = [\hat{f}(X_t) + \hat{\mathcal{Q}}_{\alpha/2}(\mathcal{S}_t), \hat{f}(X_t) + \hat{\mathcal{Q}}_{1-\alpha/2}(\mathcal{S}_t)] \quad (1)$$

- Theoretically,
  1. When $\hat{\mathcal{Q}}$ is the empirical quantile, we bound the coverage gap $|\mathbb{P}(Y_t \in C(X_t, \alpha)) - (1 - \alpha)|$.
     Details in following slides.

Part I: Proposed solution (cont.)

- When $p = 1$ ($Y_i \in \mathbb{R}$) and $s(X_t, y, \hat{f}) = y - \hat{f}(X_t)$, the prediction residual, we have prediction intervals

$$C(X_t, \alpha) = [\hat{f}(X_t) + \hat{\mathcal{Q}}_{\alpha/2}(\mathcal{S}_t), \hat{f}(X_t) + \hat{\mathcal{Q}}_{1-\alpha/2}(\mathcal{S}_t)] \quad (1)$$

- Theoretically,
  1. When $\hat{\mathcal{Q}}$ is the empirical quantile, we bound the coverage gap $|\mathbb{P}(Y_t \in C(X_t, \alpha)) - (1 - \alpha)|$.
     Details in following slides.
  2. When $\hat{\mathcal{Q}}$ is the quantile random forest [Meinshausen 2006], we show asymptotic conditional coverage $\mathbb{P}(Y_t \in C(X_t, \alpha)) \overset{T \to \infty}{\longrightarrow} 1 - \alpha$ under additional assumptions on the covariance between lagged non-conformity scores.

Part I: Theoretical guarantee

Suppose $Y_t = f(X_t) + \epsilon_t$:

$$s(X_t, y, \hat{f}) = y - \hat{f}(X_t) = \underbrace{f(X_t) - \hat{f}(X_t)}_{\text{prediction error}} + \underbrace{\epsilon_t}_{\text{“nature”}}$$

Part I: Theoretical guarantee

Suppose $Y_t = f(X_t) + \epsilon_t$:

$$s(X_t, y, \hat{f}) = y - \hat{f}(X_t) = \underbrace{f(X_t) - \hat{f}(X_t)}_{\text{prediction error}} + \underbrace{\epsilon_t}_{\text{“nature”}}$$

- Assumption 1 (Data regularity): Error process $\epsilon_1, \epsilon_2, \ldots$
  - stationary and strongly mixing
  - sum of mixing coefficients bounded by M
  - true CDF F is Lipschitz with constant $L > 0$
- Assumption 2 (Estimation quality): On T training data

$$\frac{1}{T}\sum_{t=1}^{T}(\hat{f}_t(X_t) - f(X_t))^2 \leq \delta_T^2,$$

Part I: Theoretical guarantee (cont.)

Given a training size T and $\alpha \in (0, 1)$,

$$|\mathbb{P}(Y_{T+1} \in C(X_{T+1}, \alpha) - (1 - \alpha)| \leq C((\log T/T)^{1/3} + \delta_T^{2/3})$$

- Factor $(\log T/T)^{1/3}$ comes from assuming α-mixing errors, different error assumptions (e.g., independent, stationary, etc.) yield different rates
- Coverage gap dependent on T and estimation quality of f by $\hat{f}$.

Part I: Assumption 1 can be extended

- Independent $\{\epsilon_t\}_{t\geq 1}$

$$\text{Rate} = (\log(16T)/T)^{1/2}.$$

- Stationary linear processes $\epsilon_t = \sum_{j=1}^{\infty} \delta_j z_{t-j}$.

$$\text{Rate} = \log T/\sqrt{T}$$

  Faster than strongly mixing errors, slower than independent errors.

- Joint density of $\{\epsilon_t\}_{t=1}^{T+1}$ satisfies a logarithmic Sobolev inequality

$$\text{Rate} = (\log(cT)/T)^{1/3}$$

Part I: Assumption 2: "Good" predictive algorithm

- Assumption 2 holds true for many classes of algorithms
- No-free-lunch theorem:
  assumption on f is necessary in order for us to approximate it well.

Part I: Assumption 2: "Good" predictive algorithm

- Assumption 2 holds true for many classes of algorithms
- No-free-lunch theorem:
  assumption on f is necessary in order for us to approximate it well.
- Examples
  - if f is sufficiently smooth,

    $$\delta_T = o(T^{-1/4})$$

    for neural networks sieve estimators (Chen and White, 1999).
  - If f is a sparse high-dimensional linear model,

    $$\delta_T = o(T^{-1/2})$$

    for Lasso and Dantzig selector (Bickel et al. 2009).

Part I: Prediction region when $p \geq 1$

- When $Y_t \in \mathbb{R}^p$ is a vector, we build prediction regions $C(X_t, \alpha) \subset \mathbb{R}^p$ subject to the same coverage guarantee.

Part I: Prediction region when $p \geq 1$

- When $Y_t \in \mathbb{R}^p$ is a vector, we build prediction regions $C(X_t, \alpha) \subset \mathbb{R}^p$ subject to the same coverage guarantee.
- The key lies in the design of scalar conformity scores:

$$s(X, Y, \hat{f}) = \hat{\epsilon}_t^T \hat{\Sigma}_\rho^{-1} \hat{\epsilon}_t, \tag{1}$$

where $\hat{\Sigma}_\rho$ is the low-rank approximation of sample covariance matrix $\hat{\Sigma}$ of residuals $\{\hat{\epsilon}_i\}$ in the hold-out set.

Part I: Prediction region when $p \geq 1$

- When $Y_t \in \mathbb{R}^p$ is a vector, we build prediction regions $C(X_t, \alpha) \subset \mathbb{R}^p$ subject to the same coverage guarantee.
- The key lies in the design of scalar conformity scores:

$$s(X, Y, \hat{f}) = \hat{\epsilon}_t^T \hat{\Sigma}_\rho^{-1} \hat{\epsilon}_t, \tag{1}$$

  where $\hat{\Sigma}_\rho$ is the low-rank approximation of sample covariance matrix $\hat{\Sigma}$ of residuals $\{\hat{\epsilon}_i\}$ in the hold-out set.
- This gives ellipsoids $\mathcal{B}(r, A) = \{x \in \mathbb{R}^p : x^T A x \leq r\}$.

Part I: Prediction region when $p \geq 1$

- When $Y_t \in \mathbb{R}^p$ is a vector, we build prediction regions $C(X_t, \alpha) \subset \mathbb{R}^p$ subject to the same coverage guarantee.
- The key lies in the design of scalar conformity scores:

$$s(X, Y, \hat{f}) = \hat{\epsilon}_t^T \hat{\Sigma}_\rho^{-1} \hat{\epsilon}_t, \tag{1}$$

  where $\hat{\Sigma}_\rho$ is the low-rank approximation of sample covariance matrix $\hat{\Sigma}$ of residuals $\{\hat{\epsilon}_i\}$ in the hold-out set.
- This gives ellipsoids $\mathcal{B}(r, A) = \{x \in \mathbb{R}^p : x^T A x \leq r\}$.
- Thus, the prediction region is the difference of ellipsoids:

$$C(X_t, \alpha) = \hat{f}(X_t) + \mathcal{B}(r_{\text{high}}, \hat{\Sigma}_\rho^{-1}) \backslash \mathcal{B}(r_{\text{low}}, \hat{\Sigma}_\rho^{-1}), \tag{2}$$

  where $r_{\text{high}} = \hat{\mathcal{Q}}_{1-\alpha/2}(\mathcal{S}_t)$, $r_{\text{low}} = \hat{\mathcal{Q}}_{\alpha/2}(\mathcal{S}_t)$.

Part I: Prediction region when $p \geq 1$

- When $Y_t \in \mathbb{R}^p$ is a vector, we build prediction regions $C(X_t, \alpha) \subset \mathbb{R}^p$ subject to the same coverage guarantee.
- The key lies in the design of scalar conformity scores:

$$s(X, Y, \hat{f}) = \hat{\epsilon}_t^T \hat{\Sigma}_\rho^{-1} \hat{\epsilon}_t, \tag{1}$$

  where $\hat{\Sigma}_\rho$ is the low-rank approximation of sample covariance matrix $\hat{\Sigma}$ of residuals $\{\hat{\epsilon}_i\}$ in the hold-out set.
- This gives ellipsoids $\mathcal{B}(r, A) = \{x \in \mathbb{R}^p : x^T A x \leq r\}$.
- Thus, the prediction region is the difference of ellipsoids:

$$C(X_t, \alpha) = \hat{f}(X_t) + \mathcal{B}(r_{\text{high}}, \hat{\Sigma}_\rho^{-1}) \backslash \mathcal{B}(r_{\text{low}}, \hat{\Sigma}_\rho^{-1}), \tag{2}$$

  where $r_{\text{high}} = \hat{\mathcal{Q}}_{1-\alpha/2}(\mathcal{S}_t)$, $r_{\text{low}} = \hat{\mathcal{Q}}_{\alpha/2}(\mathcal{S}_t)$.
- Theoretical guarantee closely resemble the former one for $p = 1$.

Part I: Performance metrics

We focus on:

1. Given valid ($\geq 1 - \alpha$) coverage on the test set, sizes of the prediction regions by each method.
2. Rolling coverage and sizes at particular t during test time (i.e., conditional coverage regarding time indices).

Part I: Empirical performances when $p = 1$

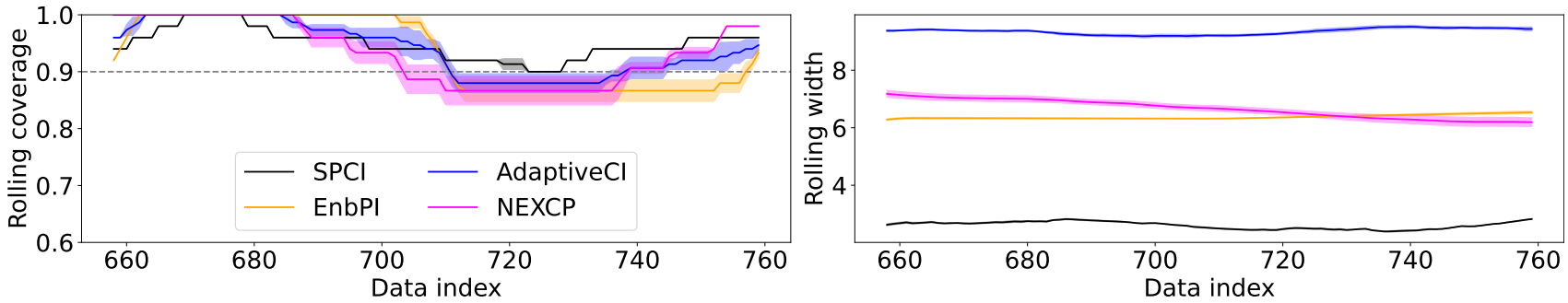
We improve over existing time-series conformal prediction methods and probabilitistic forecasting baselines.

Table: Performance of SPCI ($\hat{\mathcal{Q}}$ is quantile random forest) and EnbPI ($\hat{\mathcal{Q}}$ is empirical quantile) against existing approaches. Target coverage is 90%.

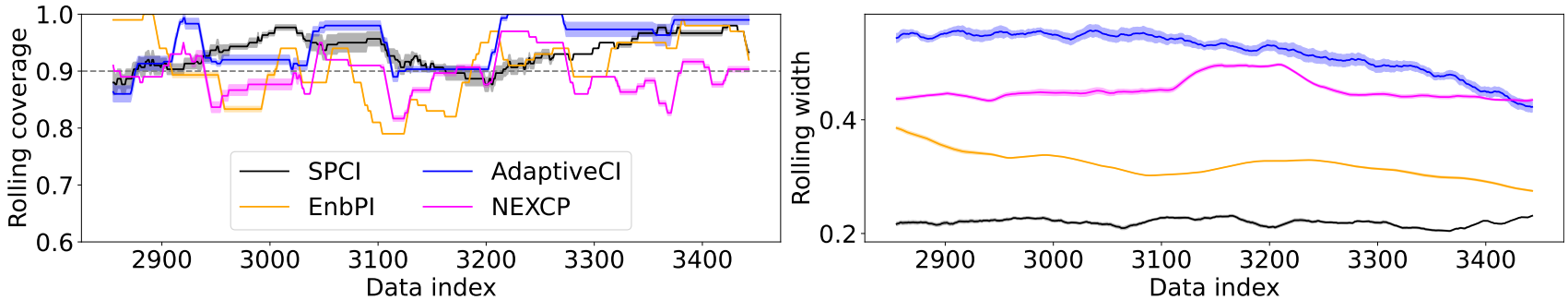
	Wind coverage	Wind width	Electric coverage	Electric width	Solar coverage	Solar width
SPCI	0.95 (1.50e-2)	2.65 (1.60e-2)	0.93 (4.79e-3)	0.22 (1.68e-3)	0.91 (1.12e-2)	47.61 (1.33e+0)
EnbPI	0.93 (6.20e-3)	6.38 (3.01e-2)	0.91 (6.84e-4)	0.32 (9.11e-4)	0.88 (4.25e-3)	48.95 (3.38e+0)
AdaptiveCI	0.95 (5.37e-3)	9.34 (3.56e-2)	0.95 (1.81e-3)	0.51 (7.25e-3)	0.96 (1.39e-2)	56.34 (1.15e+0)
NEX-CP	0.96 (8.21e-3)	6.68 (7.73e-2)	0.90 (2.05e-3)	0.45 (2.16e-3)	0.90 (7.73e-3)	102.80 (5.25e+0)
DeepAR	0.95 (5.32e-3)	6.86 (7.86e-3)	0.91 (3.45e-3)	0.62 (2.56e0-3)	0.92 (5.35e-3)	80.23 (4.94e+0)
TFT	0.92 (6.34e-2)	7.56 (5.34e-3)	0.95 (2.34e-2)	0.66 (2.34e-3)	0.93 (2.84e-3)	74.82 (4.23e+0)

Part I: Empirical performances when $p = 1$ (cont.)

We empirically see narrower rolling widths as well:



(a) Wind

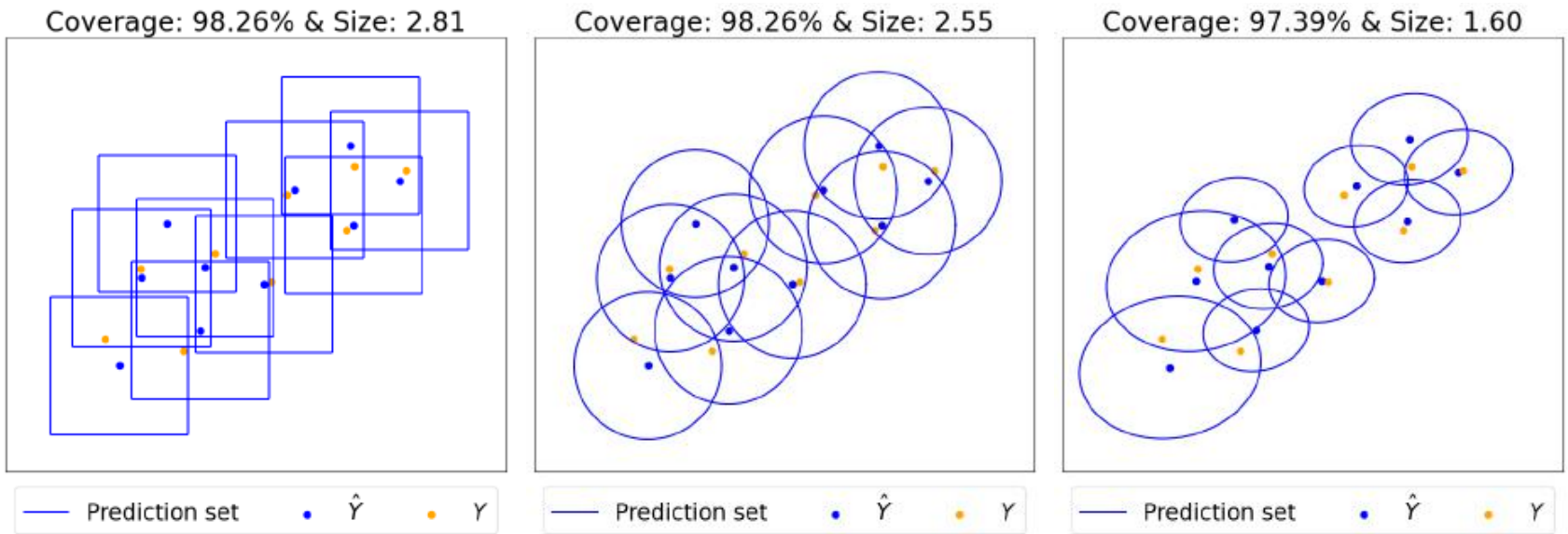


(b) Electric

Figure: Comparison of rolling coverage of CP methods on two datasets. Window size is 50 or 100.

Part I: Empirical performances when $p > 1$

Our approach on the right yield smaller prediction regions with adaptive sizes, without sacrificing coverage.



(a) (b) (c)

Figure: Comparison of ours (right) with Copula CP (left) and spherical CP (middle, $s(X_t, y, \hat{f}) = \|y - \hat{f}(X_t)\|_2$.)

Part I: Empirical performances when $p > 1$ (cont.)

On all three datasets, our approach yields significantly smaller prediction regions as p grows.

(a) Wind data

Method	$p = 2$ coverage	$p = 2$ size	$p = 4$ coverage	$p = 4$ size	$p = 8$ coverage	$p = 8$ size
MultiDimSPCI	0.97	**1.60**	0.96	**7.02**	0.96	**72.10**
CopulaCPTS (Sun & Yu, 2024)	0.98	2.55	0.97	10.23	0.97	252.67
Local ellipsoid (Messoudi et al., 2022)	0.96	3.51	0.97	13.07	0.98	1.09e+3
Copula (Messoudi et al., 2021)	0.98	2.81	0.98	10.32	0.97	1.60e+3
TFT (Lim et al., 2021)	0.94	10.61	0.75	159.39	0.94	2.91e+4
DeepAR (Salinas et al., 2020)	0.96	7.07	0.76	67.97	0.96	1.79e+5

(b) Solar data

Method	$p = 2$ coverage	$p = 2$ size	$p = 4$ coverage	$p = 4$ size	$p = 8$ coverage	$p = 8$ size
MultiDimSPCI	0.96	1.68	0.96	**2.89**	0.97	**4.97**
CopulaCPTS (Sun & Yu, 2024)	0.99	4.36	0.99	37.56	0.99	3.28e+3
Local ellipsoid (Messoudi et al., 2022)	0.97	**1.32**	0.97	3.20	0.97	43.07
Copula (Messoudi et al., 2021)	0.99	4.11	0.99	27.73	0.99	1.42e+3
TFT (Lim et al., 2021)	0.99	13.68	0.99	71.72	0.93	1.19e+3
DeepAR (Salinas et al., 2020)	0.97	10.76	0.98	157.09	0.74	31.82

(c) Traffic data

Method	$p = 2$ coverage	$p = 2$ size	$p = 4$ coverage	$p = 4$ size	$p = 8$ coverage	$p = 8$ size
MultiDimSPCI	0.96	**1.31**	0.96	**1.93**	0.96	**2.98**
CopulaCPTS (Sun & Yu, 2024)	0.95	1.70	0.94	3.15	0.95	14.10
Local ellipsoid (Messoudi et al., 2022)	0.95	1.36	0.94	2.08	0.95	4.13
Copula (Messoudi et al., 2021)	0.95	1.44	0.95	3.90	0.94	40.60
TFT (Lim et al., 2021)	0.89	9.07	0.93	87.92	0.88	9.69e+2
DeepAR (Salinas et al., 2020)	0.87	13.53	0.88	57.20	0.82	9.89e+3

Outline

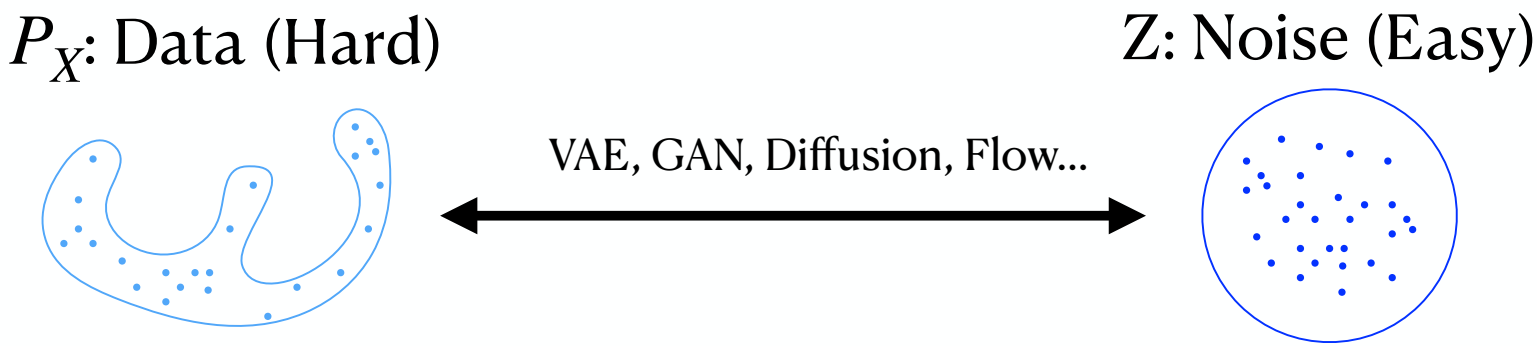
- Part I: Conformal prediction for time-series
- Part II: Flow-based generative models
- Future work: Connecting Parts I & II

Part II: Motivation

- Given d-dimensional samples $X_i \sim P_X$, we want to learn P_X by
  (1) sampling from P_X as in generative modeling.
  (2) estimate log likelihood $\log p_X(x), x \in \mathbb{R}^d$.

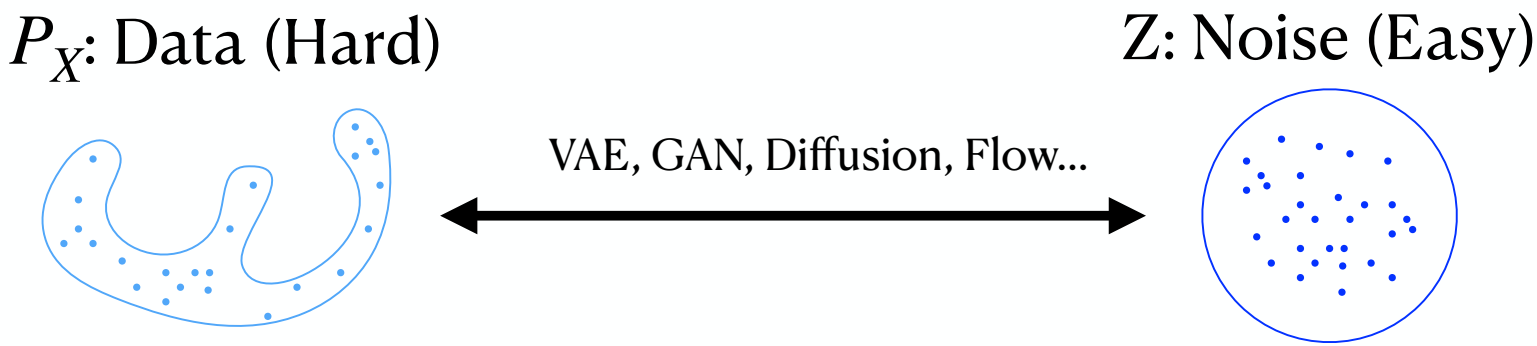
Part II: Motivation

- Given d-dimensional samples $X_i \sim P_X$, we want to learn P_X by
  (1) sampling from P_X as in generative modeling.
  (2) estimate log likelihood $\log p_X(x), x \in \mathbb{R}^d$.
- The problem is not new:



Part II: Motivation

- Given d-dimensional samples $X_i \sim P_X$, we want to learn P_X by (1) sampling from P_X as in generative modeling.
  (2) estimate log likelihood $\log p_X(x), x \in \mathbb{R}^d$.
- The problem is not new:



- The focus of flow models is to learn a *continuous and invertible* transformation (i.e., flow) between the "hard" (original P_X) and "easy" (standard Gaussian) distributions.

Part II: Setup

- Flow-based models can be represented as an *ordinary differential equation* (ODE) over a distribution pair (P, Q).

Part II: Setup

- Flow-based models can be represented as an *ordinary differential equation* (ODE) over a distribution pair (P, Q).
- Consider the random process $\{x(t), t \in [0, 1]\}$ where $x(0) \sim P$ and $x(1) \sim Q$. The process is governed by the ODE:

$$dx(t)/dt = f(x(t), t), \tag{1}$$

$$\rightarrow x(t) = x(0) + \int_0^t f(x(s), s)ds. \tag{2}$$

Part II: Setup

- Flow-based models can be represented as an *ordinary differential equation* (ODE) over a distribution pair (P, Q).
- Consider the random process $\{x(t), t \in [0, 1]\}$ where $x(0) \sim P$ and $x(1) \sim Q$. The process is governed by the ODE:

$$dx(t)/dt = f(x(t), t), \tag{1}$$

$$\rightarrow x(t) = x(0) + \int_0^t f(x(s), s)ds. \tag{2}$$

- The essential objective of flow models is to parametrize and learn f, either implicitly (normalizing flow) or explicitly (flow matching)

Part II: Setup

- Flow-based models can be represented as an *ordinary differential equation* (ODE) over a distribution pair (P, Q).
- Consider the random process $\{x(t), t \in [0, 1]\}$ where $x(0) \sim P$ and $x(1) \sim Q$. The process is governed by the ODE:

$$dx(t)/dt = f(x(t), t), \tag{1}$$

$$\rightarrow x(t) = x(0) + \int_0^t f(x(s), s)ds. \tag{2}$$

- The essential objective of flow models is to parametrize and learn f, either implicitly (normalizing flow) or explicitly (flow matching)
- Our contributions lie in proposing iterative algorithms that improve both ways of training flow models.

Part II: Setup

ODE formulation with $x(0) \sim P, x(1) \sim Q$:

$$dx(t)/dt = f(x(t), t), \tag{1}$$

$$\rightarrow x(t) = x(0) + \int_0^t f(x(s), s)ds. \tag{2}$$

Once we have such an f:

Part II: Setup

ODE formulation with $x(0) \sim P, x(1) \sim Q$:

$$dx(t)/dt = f(x(t), t), \tag{1}$$

$$\rightarrow x(t) = x(0) + \int_0^t f(x(s), s)ds. \tag{2}$$

Once we have such an f:

1. (Forward $\rightarrow$, starting at $x(0) \sim P$) Perform likelihood estimation of $\log p(x(0))$ with change of variables: integrate f over $t \in [0, 1]$ to collect $\int_0^1 \nabla \cdot f$ and $\log q(\hat{x}(1))$.

Part II: Setup

ODE formulation with $x(0) \sim P, x(1) \sim Q$:

$$dx(t)/dt = f(x(t), t), \tag{1}$$

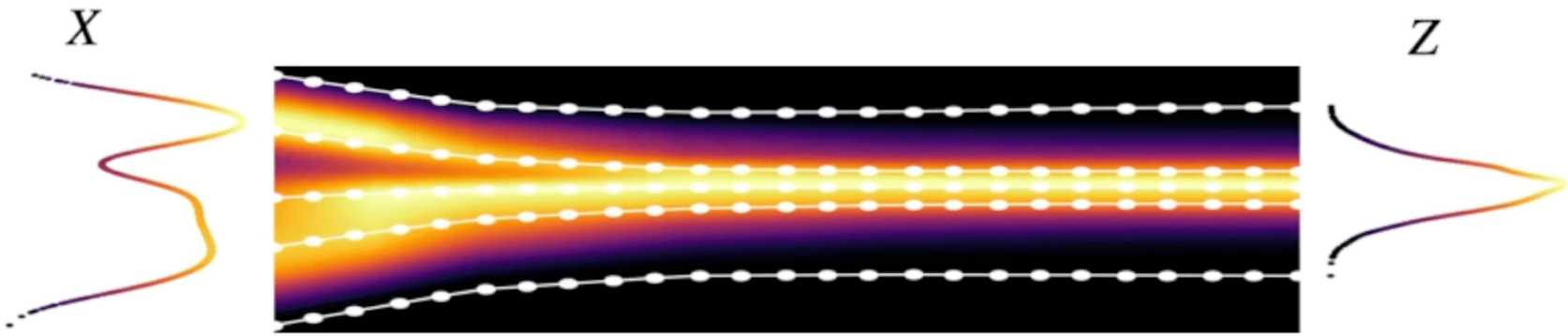
$$\rightarrow x(t) = x(0) + \int_0^t f(x(s), s)ds. \tag{2}$$

Once we have such an f:

1. (Forward $\rightarrow$, starting at $x(0) \sim P$) Perform likelihood estimation of $\log p(x(0))$ with change of variables: integrate f over $t \in [0, 1]$ to collect $\int_0^1 \nabla \cdot f$ and $\log q(\hat{x}(1))$.
2. (Backward $\leftarrow$, starting at $x(1) \sim Q$) Perform generation of new samples $\hat{x}(0)$ from P upon integrating f over $t \in [1, 0]$.

Part II: Illustration

- $X(0) = X \sim P, X(1) = Z \sim \mathcal{N}(0, I)$.
- ($\rightarrow$) Forward process for likelihood estimation.
- ($\leftarrow$) Reverse process for sampling from P.



Part II: Normalizing flow

- Let $P = P_X$ (data distribution). When $Q = Z, Z \sim \mathcal{N}(0, I)$, this is called a *normalizing* flow [Grathwohl et al., 2019].
- Goal: Parametrize f by a model $f(\cdot; \theta)$ and learn θ.

Part II: Normalizing flow

- Let $P = P_X$ (data distribution). When $Q = Z, Z \sim \mathcal{N}(0, I)$, this is called a *normalizing* flow [Grathwohl et al., 2019].
- Goal: Parametrize f by a model $f(\cdot; \theta)$ and learn θ.
- Define the solution map $T_\theta(x) = x + \int_0^1 f(x(s), s; \theta) ds, x(0) = x$.
  Let $T_\# P_X$ be the pushforward measure: $(T_\# P_X)(\cdot) = P_X(T^{-1}(\cdot))$.

Part II: Normalizing flow

- Let $P = P_X$ (data distribution). When $Q = Z, Z \sim \mathcal{N}(0, I)$, this is called a *normalizing* flow [Grathwohl et al., 2019].
- Goal: Parametrize f by a model $f(\cdot; \theta)$ and learn θ.
- Define the solution map $T_\theta(x) = x + \int_0^1 f(x(s), s; \theta) ds, x(0) = x$.
  Let $T_\# P_X$ be the pushforward measure: $(T_\# P_X)(\cdot) = P_X(T^{-1}(\cdot))$.
- Training objective:

$$\min_\theta \mathrm{KL}((T_\theta)_\# P_X \| Z). \tag{1}$$

(1) has a closed-form expression in θ, which is trained with finite-sample approximation.

Part II: Normalizing flow

- Let $P = P_X$ (data distribution). When $Q = Z, Z \sim \mathcal{N}(0, I)$, this is called a *normalizing* flow [Grathwohl et al., 2019].
- Goal: Parametrize f by a model $f(\cdot; \theta)$ and learn θ.
- Define the solution map $T_\theta(x) = x + \int_0^1 f(x(s), s; \theta) ds, x(0) = x$.
  Let $T_\# P_X$ be the pushforward measure: $(T_\# P_X)(\cdot) = P_X(T^{-1}(\cdot))$.
- Training objective:

$$\min_\theta \mathrm{KL}((T_\theta)_\# P_X \| Z). \tag{1}$$

  (1) has a closed-form expression in θ, which is trained with finite-sample approximation.
- Training is implicit because a priori the properties of f and $f(\cdot; \theta^*)$ are unknown.

Part II: Proposed solution

- We are inspired by the JKO scheme [Jordan et al., 1998] to build a block-wise training scheme: $\theta \to \{\theta_k\}$.

Part II: Proposed solution

- We are inspired by the JKO scheme [Jordan et al., 1998] to build a block-wise training scheme: $\theta \to \{\theta_k\}$.
- Starting from $P_0 = P_X$, with step size $h > 0$, the JKO scheme at the $k-$th step is

$$P_k^* = \arg\min_{P \in \mathcal{P}} \mathsf{KL}(P \| Z) + \frac{1}{2h} W_2^2(P_{k-1}, P). \tag{1}$$

Part II: Proposed solution

- We are inspired by the JKO scheme [Jordan et al., 1998] to build a block-wise training scheme: $\theta \to \{\theta_k\}$.
- Starting from $P_0 = P_X$, with step size $h > 0$, the JKO scheme at the $k-$th step is

$$P_k^* = \arg\min_{P \in \mathcal{P}} \mathsf{KL}(P \| Z) + \frac{1}{2h} W_2^2(P_{k-1}, P). \tag{1}$$

- We instead solve for θ_k^* of the k-th transport map T_{θ_k}:

$$\theta_k^* = \arg\min_{\theta} \mathsf{KL}((T_\theta)_\# P_{k-1} \| Z) + \frac{1}{2h} \mathbb{E}_{x \sim P_{k-1}} \|x - T_\theta(x)\|^2. \tag{2}$$

Part II: Proposed solution

- We are inspired by the JKO scheme [Jordan et al., 1998] to build a block-wise training scheme: $\theta \to \{\theta_k\}$.
- Starting from $P_0 = P_X$, with step size $h > 0$, the JKO scheme at the $k-$th step is

$$P_k^* = \arg\min_{P \in \mathcal{P}} \mathsf{KL}(P \| Z) + \frac{1}{2h} W_2^2(P_{k-1}, P). \tag{1}$$

- We instead solve for θ_k^* of the k-th transport map T_{θ_k}:

$$\theta_k^* = \arg\min_{\theta} \mathsf{KL}((T_\theta)_\# P_{k-1} \| Z) + \frac{1}{2h} \mathbb{E}_{x \sim P_{k-1}} \|x - T_\theta(x)\|^2. \tag{2}$$

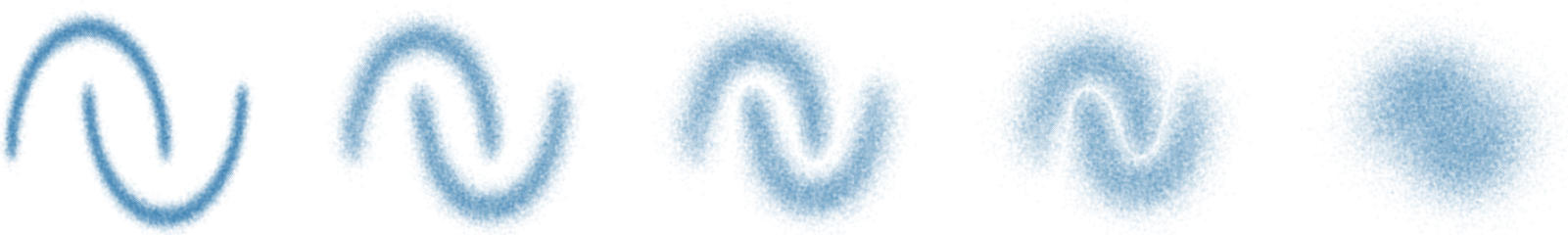
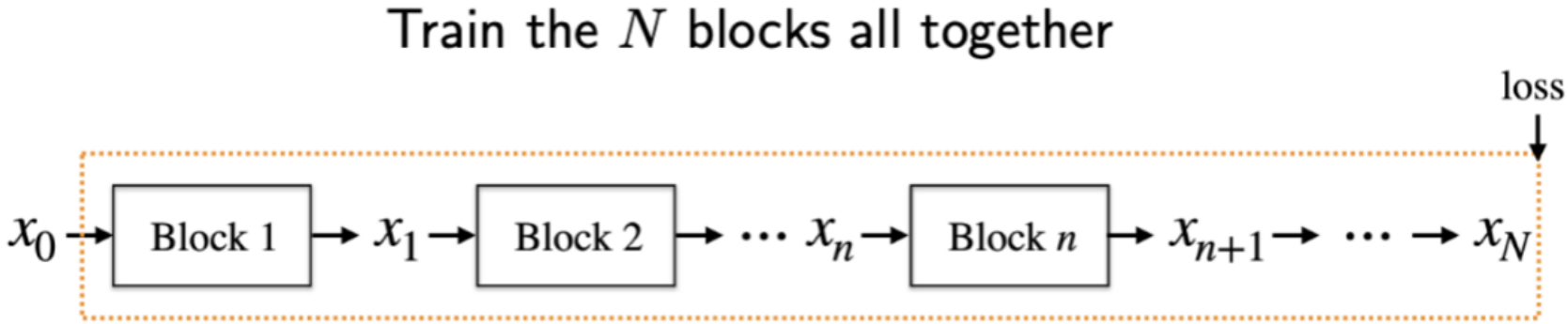


Figure: P_0 (leftmost) to P_4 (rightmost) after 4 JKO blocks.

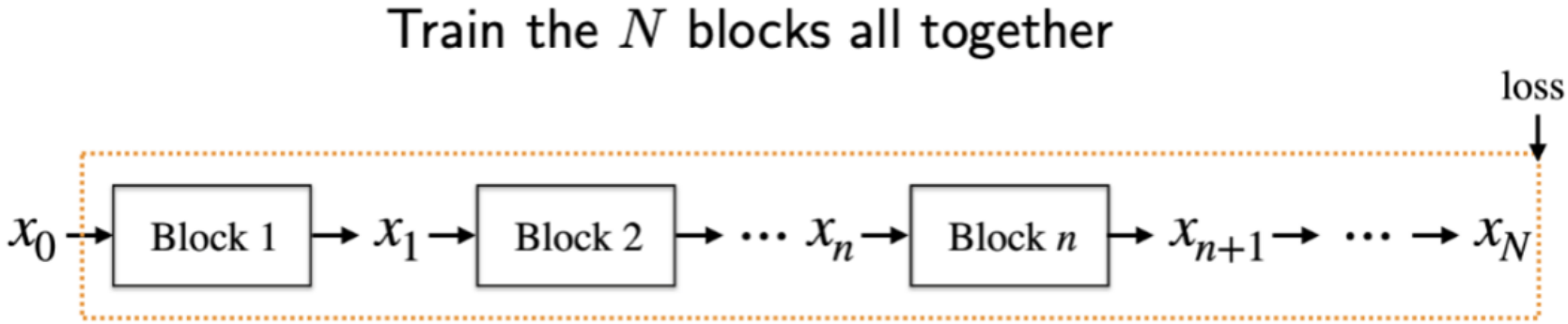
Part II: Proposed solution (cont.)

- Existing end-to-end training

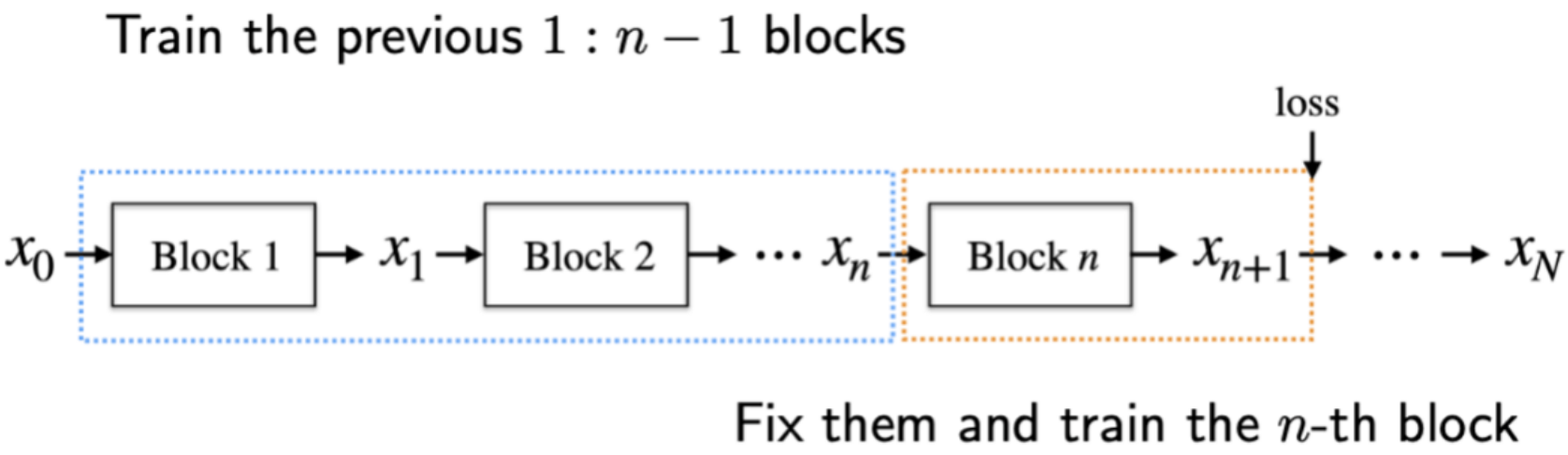


Part II: Proposed solution (cont.)

- Existing end-to-end training



- Our progressive training



- Benefits: More flexible and save computation.

Part II: Flow Matching

- Recall the focus of flow models is to learn the ODE velocity field $dx(t)/dt = f(x(t), t)$ where $x(0) \sim P$ and $x(1) \sim Q$.

Part II: Flow Matching

- Recall the focus of flow models is to learn the ODE velocity field $dx(t)/dt = f(x(t), t)$ where $x(0) \sim P$ and $x(1) \sim Q$.
- Normalizing flow uses the KL divergence objective. This requires repeated function evaluations of $f(\cdot; \theta)$ to get $(T_\theta)_\# P$ and is expensive for large $f(\cdot; \theta)$ in high dimensions.

Part II: Flow Matching

- Recall the focus of flow models is to learn the ODE velocity field $dx(t)/dt = f(x(t), t)$ where $x(0) \sim P$ and $x(1) \sim Q$.
- Normalizing flow uses the KL divergence objective. This requires repeated function evaluations of $f(\cdot; \theta)$ to get $(T_\theta)_\# P$ and is expensive for large $f(\cdot; \theta)$ in high dimensions.
- Hence, recent efforts try to find scalable alternatives to learn f.

Part II: Flow Matching

Flow matching [Lipman el at., 2023, Albergo and Vanden-Eijnden 2023] explicitly specifies and learns $f(x(t), t) = dx(t)/dt$:

Part II: Flow Matching

Flow matching [Lipman el at., 2023, Albergo and Vanden-Eijnden 2023] explicitly specifies and learns $f(x(t), t) = dx(t)/dt$:

1. Design $x(t)$ and $dx(t)/dt$, where $x(0) \sim P$ and $x(1) \sim Q$.
   A simple example is the linear interpolation:
   $x(t) = x(0) + t(x(1) - x(0)), dx(t)/dt = x(1) - x(0)$.

Part II: Flow Matching

Flow matching [Lipman el at., 2023, Albergo and Vanden-Eijnden 2023] explicitly specifies and learns $f(x(t), t) = dx(t)/dt$:

1. Design $x(t)$ and $dx(t)/dt$, where $x(0) \sim P$ and $x(1) \sim Q$.
   A simple example is the linear interpolation:
   $x(t) = x(0) + t(x(1) - x(0)), dx(t)/dt = x(1) - x(0)$.
2. Train $f(x(t), t; \theta)$ to learn $dx(t)/dt$ via

$$\min_{\theta} \mathbb{E}_{t, x(t)|x(0), x(1)} \|dx(t)/dt - f(x(t), t; \theta)\|_2^2. \tag{1}$$

   In practice, minimization of (1) is via finite-sample approximation upon sampling $t \in U[0, 1]$ and independent batches of $x(0)$ and $x(1)$ from training data.

Part II: Proposed solution

- Motivated by the former JKO scheme, we want to match iteratively and locally P_{k-1} and P_k by a smaller model $f(\cdot;\theta_k)$, where $P_0 = P$ and $P_K = Q$.
- Q: How to design (P_{k-1}, P_k)?

Part II: Proposed solution

- Motivated by the former JKO scheme, we want to match iteratively and locally P_{k-1} and P_k by a smaller model $f(\cdot;\theta_k)$, where $P_0 = P$ and $P_K = Q$.
- Q: How to design (P_{k-1}, P_k)?
- For generative modeling ($Q = \mathcal{N}(0, I_d)$), it is natural to consider the Ornstein–Uhlenbeck (OU) process starting at $X_0 \sim P$ for $t = 0$:

$$dX_t = -X_t dt + \sqrt{2} dW_t \tag{1}$$

$$\rightarrow X_t = \exp(-t) X_0 + \sqrt{1 - \exp(-2t)} Z, Z \sim \mathcal{N}(0, I_d) \tag{2}$$

Part II: Proposed solution

- Motivated by the former JKO scheme, we want to match iteratively and locally P_{k-1} and P_k by a smaller model $f(\cdot;\theta_k)$, where $P_0 = P$ and $P_K = Q$.
- Q: How to design (P_{k-1}, P_k)?
- For generative modeling ($Q = \mathcal{N}(0, I_d)$), it is natural to consider the Ornstein–Uhlenbeck (OU) process starting at $X_0 \sim P$ for $t = 0$:

$$dX_t = -X_t dt + \sqrt{2} dW_t \tag{1}$$

$$\rightarrow X_t = \exp(-t) X_0 + \sqrt{1 - \exp(-2t)} Z, Z \sim \mathcal{N}(0, I_d) \tag{2}$$

- Denote (2) as $P_t^* = (\mathrm{OU})_0^t(P)$ where $X_t \sim P_t^*$ and $X_0 \sim P$.

Part II: Proposed solution

- Motivated by the former JKO scheme, we want to match iteratively and locally P_{k-1} and P_k by a smaller model $f(\cdot;\theta_k)$, where $P_0 = P$ and $P_K = Q$.
- Q: How to design (P_{k-1}, P_k)?
- For generative modeling ($Q = \mathcal{N}(0, I_d)$), it is natural to consider the Ornstein–Uhlenbeck (OU) process starting at $X_0 \sim P$ for $t = 0$:

$$dX_t = -X_t dt + \sqrt{2} dW_t \tag{1}$$

$$\rightarrow X_t = \exp(-t) X_0 + \sqrt{1 - \exp(-2t)} Z, Z \sim \mathcal{N}(0, I_d) \tag{2}$$

- Denote (2) as $P_t^* = (\mathrm{OU})_0^t(P)$ where $X_t \sim P_t^*$ and $X_0 \sim P$.
- We can thus apply flow matching to the pair (P_t^*, P) and obtain $P_t = (T_\theta)_\# P$ as the pushforward distribution afterwards.

Part II: Proposed solution (cont.)

Let $P_0 = P_X$. Given a sequence of local stepsizes $\{\gamma_k\}$, our proposed *local* flow matching (LFM) proceeds as follows:

Part II: Proposed solution (cont.)

Let $P_0 = P_X$. Given a sequence of local stepsizes $\{\gamma_k\}$, our proposed *local* flow matching (LFM) proceeds as follows:

For the k-th block:

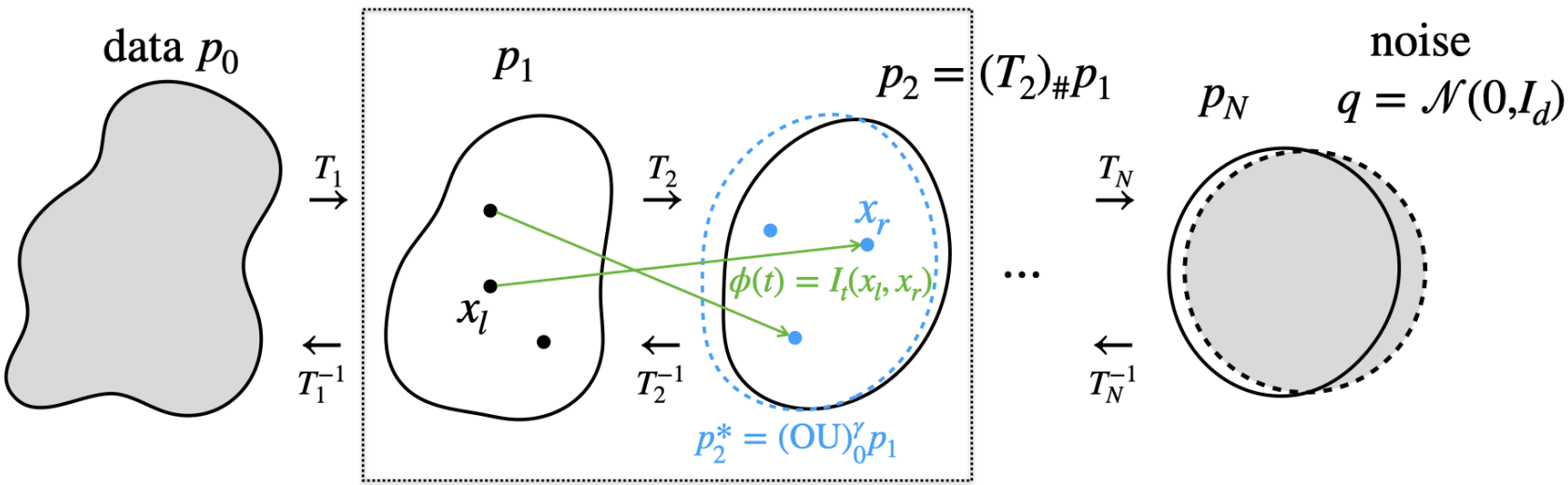
1. Obtain the pair (P_k^*, P_{k-1}) for $P_k^* = (\mathsf{OU})_0^{\gamma_k} P_{k-1}$.
2. Train $f(\cdot; \theta_k)$ using flow matching on (P_k^*, P_{k-1}).
3. Push-forward to get $P_k = (T_k)_{\#} P_{k-1}$ for $T_k = T_{\theta_k}$.

Part II: Proposed solution (cont.)

Let $P_0 = P_X$. Given a sequence of local stepsizes $\{\gamma_k\}$, our proposed *local* flow matching (LFM) proceeds as follows:

For the k-th block:

1. Obtain the pair (P_k^*, P_{k-1}) for $P_k^* = (\mathrm{OU})_0^{\gamma_k} P_{k-1}$.
2. Train $f(\cdot; \theta_k)$ using flow matching on (P_k^*, P_{k-1}).
3. Push-forward to get $P_k = (T_k)_{\#} P_{k-1}$ for $T_k = T_{\theta_k}$.



Part II: Theoretical Guarantee

- Let $P^*_{n+1} = (\mathrm{OU})_0^\gamma P_n$ (OU with uniform step sizes).
  Recall $P_{n+1} = (T_{n+1})_\# P_n$ (pushforward distribution).

Part II: Theoretical Guarantee

- Let $P^*_{n+1} = (\mathrm{OU})_0^\gamma P_n$ (OU with uniform step sizes).
  Recall $P_{n+1} = (T_{n+1})_\# P_n$ (pushforward distribution).
- Forward (data → noise, training) and reverse (noise → data, generation) processes:

$$\begin{aligned} &\text{(forward)} \quad P = P_0 \xrightarrow{T_1} P_1 \xrightarrow{T_2} \cdots \xrightarrow{T_N} P_N \approx Q, \\ &\text{(reverse)} \quad P \approx Q_0 \xleftarrow{T_1^{-1}} Q_1 \xleftarrow{T_2^{-1}} \cdots \xleftarrow{T_N^{-1}} Q_N = Q, \end{aligned} \tag{1}$$

Part II: Theoretical Guarantee

- Let $P^*_{n+1} = (\mathrm{OU})_0^\gamma P_n$ (OU with uniform step sizes).
  Recall $P_{n+1} = (T_{n+1})_\# P_n$ (pushforward distribution).
- Forward (data → noise, training) and reverse (noise → data, generation) processes:

$$\begin{aligned} &\text{(forward)} \quad P = P_0 \xrightarrow{T_1} P_1 \xrightarrow{T_2} \cdots \xrightarrow{T_N} P_N \approx Q, \\ &\text{(reverse)} \quad P \approx Q_0 \xleftarrow{T_1^{-1}} Q_1 \xleftarrow{T_2^{-1}} \cdots \xleftarrow{T_N^{-1}} Q_N = Q, \end{aligned} \tag{1}$$

- Let $\varepsilon^2 = \int_0^1 \int_{\mathbb{R}^d} \|f - \hat{f}\|^2 \rho_t(x) dx dt$ be the FM learning error (uniform for all n).
- We analyze the closeness $\chi^2(P_N \| Q)$ and $\chi^2(P \| Q_0)$ in ε.

Part II: Theoreical Guarantee (cont.)

- Under regularity assumptions on P, we can prove

$$\text{(forward)} \quad \chi^2(P_N \| Q) \leq \underbrace{\exp(-2\gamma N)\chi^2(P_0\|Q)}_{(1)} + \underbrace{C_\gamma \varepsilon^{1/2}}_{(2)} \quad (1)$$

Part II: Theoreical Guarantee (cont.)

- Under regularity assumptions on P, we can prove

$$\text{(forward)} \quad \chi^2(P_N \| Q) \leq \underbrace{\exp(-2\gamma N)\chi^2(P_0\|Q)}_{(1)} + \underbrace{C_\gamma \varepsilon^{1/2}}_{(2)} \quad (1)$$

(1) utilizes the contraction property of OU processes and (2) analyzes closeness $\chi^2(P_N\|Q) \approx \chi^2(P_N^*\|Q)$ in ε.

Part II: Theoreical Guarantee (cont.)

- Under regularity assumptions on P, we can prove

$$\text{(forward)} \quad \chi^2(P_N \| Q) \le \underbrace{\exp(-2\gamma N)\chi^2(P_0\|Q)}_{(1)} + \underbrace{C_\gamma \varepsilon^{1/2}}_{(2)} \quad (1)$$

  (1) utilizes the contraction property of OU processes and (2) analyzes closeness $\chi^2(P_N\|Q) \approx \chi^2(P_N^*\|Q)$ in ε.

- By invertibility of T_k and the data processing inequality for f-divergences, for $N \sim \log(1/\varepsilon)$, we have

$$\text{(reverse)} \quad \chi^2(P\|Q_0) = \chi^2(P_N\|Q) \le 2C_\gamma \varepsilon^{1/2}. \quad (2)$$

Part II: Theoreical Guarantee (cont.)

- Under regularity assumptions on P, we can prove

$$\text{(forward)} \quad \chi^2(P_N \| Q) \leq \underbrace{\exp(-2\gamma N)\chi^2(P_0\|Q)}_{(1)} + \underbrace{C_\gamma \varepsilon^{1/2}}_{(2)} \quad (1)$$

  (1) utilizes the contraction property of OU processes and (2) analyzes closeness $\chi^2(P_N\|Q) \approx \chi^2(P_N^*\|Q)$ in ε.

- By invertibility of T_k and the data processing inequality for f-divergences, for $N \sim \log(1/\varepsilon)$, we have

$$\text{(reverse)} \quad \chi^2(P\|Q_0) = \chi^2(P_N\|Q) \leq 2C_\gamma \varepsilon^{1/2}. \quad (2)$$

- This means small learning error ε guarantees small χ^2 generation error, which further implies small KL-divergence between P and Q_0 because $\mathrm{KL}(P\|Q_0) \leq \chi^2(P\|Q_0)$.

Part II: Performance metrics

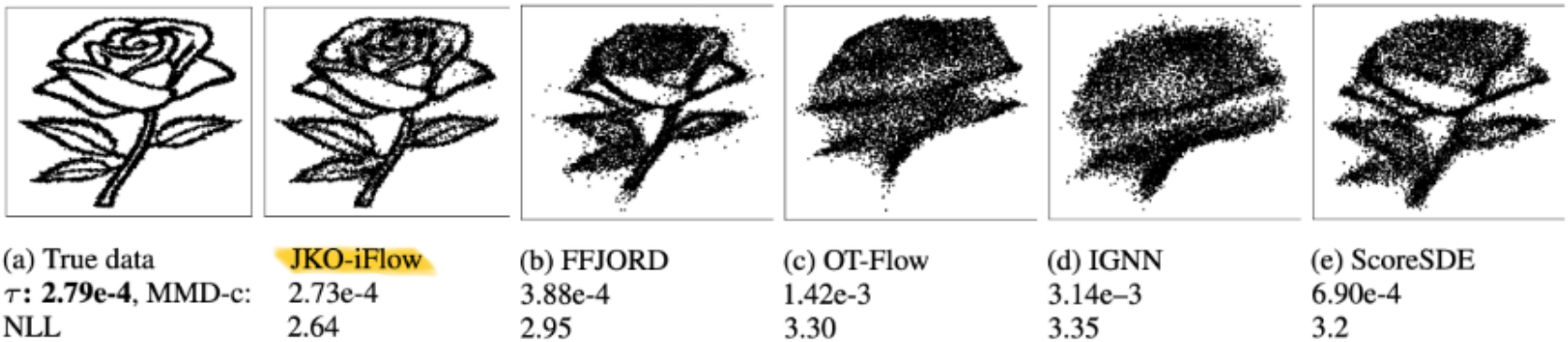
We focus on:

1. Generative performance: qualitative visual quality and quantitative metrics (e.g., FID for images and success rates for robotic manipulation).
2. Likelihood estimation in negative log-likelihood (NLL).

Takeaway: Our iterative schemes converge faster than non-iterative baselines.

Part II: Empirical Performances (Normalizing flow)

- Metrics are maximum mean discrepancy (MMD) and NLL.



(a) True data
τ: **2.79e-4**, MMD-c:
NLL

JKO-iFlow
2.73e-4
2.64

(b) FFJORD
3.88e-4
2.95

(c) OT-Flow
1.42e-3
3.30

(d) IGNN
3.14e–3
3.35

(e) ScoreSDE
6.90e-4
3.2

(f) Fractal tree
τ: **3.12e-4**, MMD-c: 2.17e-4
NLL 2.20

(g) Olympic rings
τ: **3.16e-4**, MMD-c: 2.36e-4
NLL 1.66

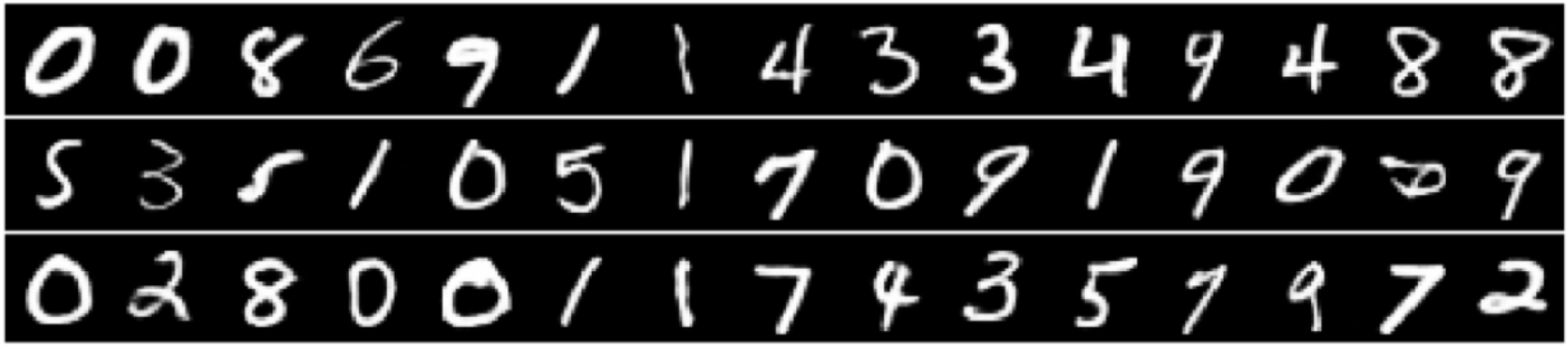
(h) Checkerboard
τ: **3.09e-4**, MMD-c: 2.70e-4
NLL 3.59

Part II: Empirical Performances (Normalizing flow)

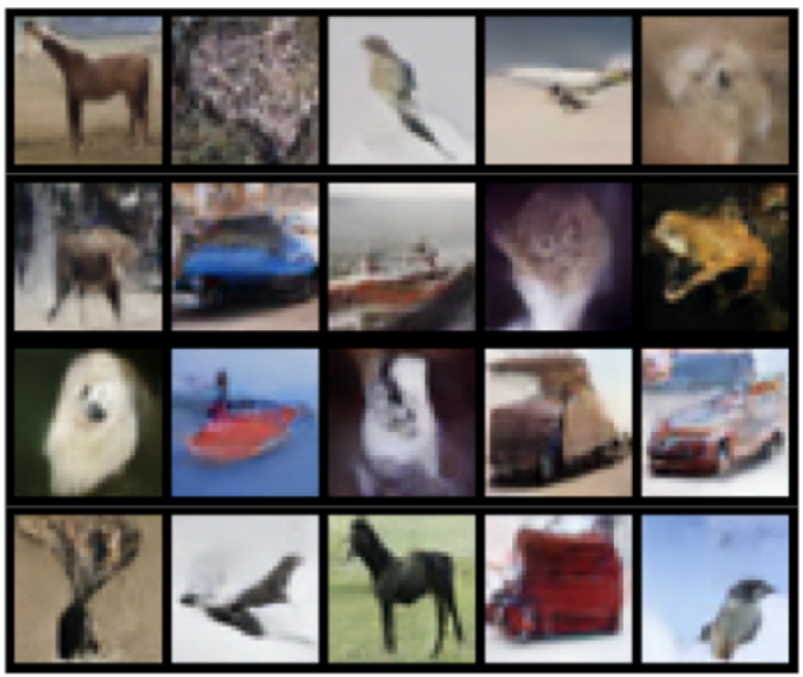
Data Set	Model	# Param	Training				Testing		
			Time (h)	# Batches	Time/Batches (s)	Batch size	MMD-m	MMD-1	NLL
							τ: **1.73e-4**	τ: **2.90e-4**	
	JKO-iFlow	76K, L=4	0.07	0.76K	3.51e-1	10000	9.86e-5	2.40e-4	-0.12
	OT-Flow	76K	0.36	7.58K	1.71e-1	10000	7.58e-4	5.35e-4	0.32
POWER	FFJORD	76K, L=4	0.67	7.58K	3.18e-1	10000	9.89e-4	1.16e-3	0.63
$d = 6$	IGNN	304K, L=16	0.29	7.58K	1.38e-1	10000	1.93e-3	1.59e-3	0.95
	IResNet	304K, L=16	0.41	7.58K	1.95e-1	10000	3.92e-3	2.43e-2	3.37
	ScoreSDE	76K	0.06	7.58K	2.85e-2	10000	9.12e-4	6.08e-3	3.41
	ScoreSDE	76K	0.60	75.80K	2.85e-2	10000	7.12e-4	5.04e-3	3.33
	JKO-iFlow	57K, L=3	0.05	0.76K	2.63e-1	10000	3.86e-4	7.20e-4	-0.06
							τ: **1.85e-4**	τ: **2.73e-4**	
	JKO-iFlow	76K, L=4	0.07	0.76K	3.32e-1	5000	1.52e-4	5.00e-4	-7.65
	OT-Flow	76K	0.23	7.60K	1.09e-1	5000	1.99e-4	5.16e-4	-6.04
GAS	FFJORD	76K, L=4	0.65	7.60K	3.08e-1	5000	1.87e-3	3.28e-3	-2.65
$d = 8$	IGNN	304K, L=16	0.34	7.60K	1.61e-1	5000	6.74e-3	1.43e-2	-1.65
	IResNet	304K, L=16	0.46	7.60K	2.18e-1	5000	3.20e-3	2.73e-2	-1.17
	ScoreSDE	76K	0.03	7.60K	1.42e-2	5000	1.05e-3	8.36e-4	-3.69
	ScoreSDE	76K	0.30	76.00K	1.42e-2	5000	2.23e-4	3.38e-4	-5.58
	JKO-iFlow	95K, L=5	0.09	0.76K	4.15e-1	5000	1.51e-4	3.77e-4	-7.80
							τ: **2.46e-4**	τ: **3.75e-4**	
	JKO-iFlow	112K, L=4	0.03	0.34K	3.61e-1	2000	9.66e-4	3.79e-4	12.55
	OT-Flow	112K	0.21	3.39K	2.23e-1	2000	6.58e-4	3.79e-4	11.44
MINIBOONE	FFJORD	112K, L=4	0.28	3.39K	2.97e-1	2000	3.51e-3	4.12e-4	23.77
$d = 43$	IGNN	448K, L=16	0.63	3.39K	6.69e-1	2000	1.21e-2	4.01e-4	26.45
	IResNet	448K, L=16	0.71	3.39K	7.54e-1	2000	2.13e-3	4.16e-4	22.36
	ScoreSDE	112K	0.01	3.39K	6.37e-3	2000	5.86e-1	4.33e-4	27.38
	ScoreSDE	112K	0.10	33.90K	6.37e-3	2000	4.17e-3	3.87e-4	20.70
							τ: **1.38e-4**	τ: **1.01e-4**	
	JKO-iFlow	396K, L=4	0.05	1.03K	1.85e-1	1000	2.24e-4	1.91e-4	-153.82
	OT-Flow	396K	0.62	10.29K	2.17e-1	1000	5.43e-1	6.49e-1	-104.62
BSDS300	FFJORD	396K, L=4	0.54	10.29K	1.89e-1	1000	5.60e-1	6.76e-1	-37.80
$d = 63$	IGNN	990K, L=10	1.71	10.29K	5.98e-1	1000	5.64e-1	6.86e-1	-37.68
	IResNet	990K, L=10	2.05	10.29K	7.17e-1	1000	5.50e-1	5.50e-1	-33.11
	ScoreSDE	396K	0.01	10.29K	3.50e-3	1000	5.61e-1	6.60e-1	-7.55
	ScoreSDE	396K	0.10	102.90K	3.50e-3	1000	5.61e-1	6.62e-1	-7.31
	JKO-iFlow	396K, L=4	0.08	1.03K	2.76e-1	5000	1.41e-4	8.83e-5	-156.68

Figure: Quantitative metrics (MMD and NLL)

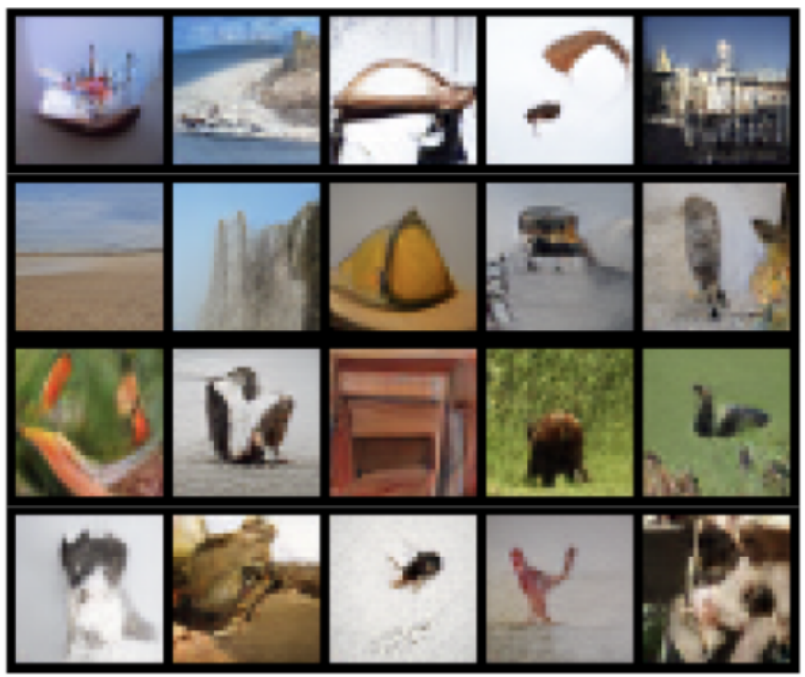
Part II: Empirical Performances (Normalizing flow)



(a) Generated MNIST digits. FID: 7.95.

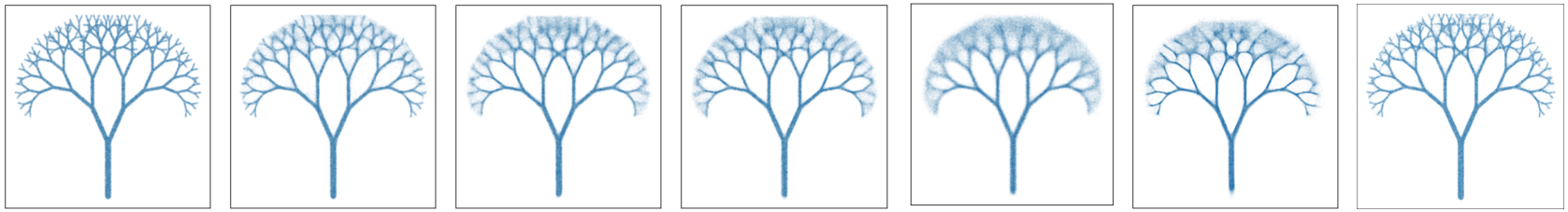


(b) Generated CIFAR10 images. FID: 29.10.

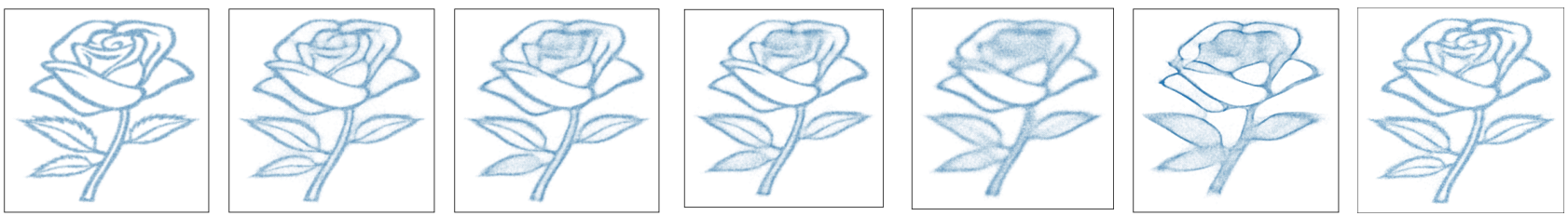


(c) Generated Imagenet-32 images. FID: 20.10.

Part II: Empirical Performances (Flow Matching)



(a)	(b)	(c)	(d)	(e)	(f)	(g)
True data from p	LFM NLL=<u>2.24</u>	FM NLL=2.35	InterFlow NLL=2.36	ScoreSDE NLL=2.51	Roundtrip NLL=2.45	JKO-iFlow NLL=**2.11**



(h)	(i)	(j)	(k)	(l)	(m)	(n)
True data from p	LFM NLL=<u>2.60</u>	FM NLL=2.64	InterFlow NLL=2.64	ScoreSDE NLL=2.72	Roundtrip NLL=2.86	JKO-iFlow NLL=**2.53**

Figure: The lowest NLL is in **bold** and the second lowest NLL is <u>underlined</u>.

Part II: Empirical Performances (Flow Matching)

Table: Test NLL (lower is better) on tabular data, where d denotes the data dimension. The lowest NLL is indicated in **bold**, and the second-lowest NLL is underlined.

	LFM	InterFlow	JKO-iFlow	AdaCat	ScoreSDE	Roundtrip	OT-Flow	nMDMA	CPF	BNAF	FFJORD
POWER (d=6)	-0.67	-0.57	-0.40	-0.56	-0.47	4.33	-0.30	**-1.78**	-0.52	-0.61	-0.46
GAS (d=8)	**-12.43**	-12.35	-9.43	-11.27	-11.65	1.62	-9.20	-8.43	-10.36	-12.06	-8.59
MINIBOONE (d=43)	9.95	10.42	10.55	14.14	10.45	18.63	10.55	18.60	10.58	**8.95**	10.43
BSDS300 (d=63)	**-157.80**	-156.22	-157.75	-	-143.31	27.29	-154.20	-	-154.99	-157.36	-157.40

Part II: Empirical Performances (Flow Matching)

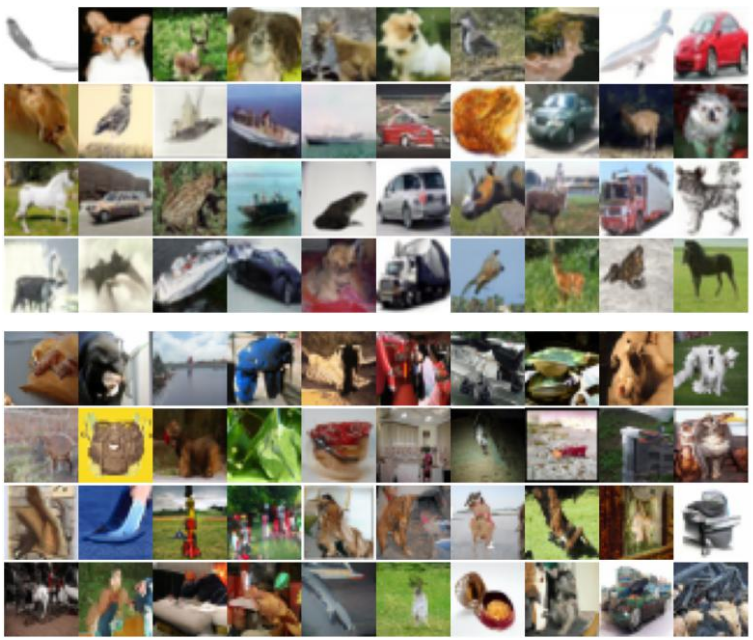
Table: FID (lower is better) comparison. LFM converges faster than FM baselines.

	CIFAR-10			Imagenet-32			Flowers-128		
	FID	Batch size	# of batches	FID	Batch size	# of batches	FID	Batch size	# of batches
LFM (Trig interpolant)	**8.45**	**200**	$\mathbf{5 \times 10^4}$	**7.00**	**256**	$\mathbf{2 \times 10^5}$	**59.7**	40	4×10^4
InterFlow	10.27*	400	5×10^5	8.49*	512	6×10^5	65.9	40	4×10^4
LFM (OT interpolant)	**8.55**	200	5×10^4	**7.20**	256	2×10^5	**55.7**	40	4×10^4
FM	12.30	200	5×10^4	7.51	256	2×10^5	70.8	40	4×10^4

Part II: Empirical Performances (Flow Matching)

Table: FID (lower is better) comparison. LFM converges faster than FM baselines.

	CIFAR-10			Imagenet-32			Flowers-128		
	FID	Batch size	# of batches	FID	Batch size	# of batches	FID	Batch size	# of batches
LFM (Trig interpolant)	**8.45**	**200**	$\mathbf{5 \times 10^4}$	**7.00**	**256**	$\mathbf{2 \times 10^5}$	**59.7**	40	4×10^4
InterFlow	10.27*	400	5×10^5	8.49*	512	6×10^5	65.9	40	4×10^4
LFM (OT interpolant)	**8.55**	200	5×10^4	**7.20**	256	2×10^5	**55.7**	40	4×10^4
FM	12.30	200	5×10^4	7.51	256	2×10^5	70.8	40	4×10^4



Part II: Empirical Performances (Flow Matching)

LFM facilitates model distillation for fast generation of 128×128 images. NFE $=$ number of fucntion evaluations of $f(\cdot; \theta)$.

Table: FID of LFM vs. InterFlow [Albergo and Vanden-Eijnden, 2023].

	Pre-distillation	Distilled @ 4 NFEs	Distilled @ 2 NFEs
LFM	59.7	**71.0**	**75.2**
InterFlow	59.7	80.0	82.4

Part II: Empirical Performances (Flow Matching)

LFM facilitates model distillation for fast generation of 128×128 images. NFE $=$ number of fucntion evaluations of $f(\cdot; \theta)$.

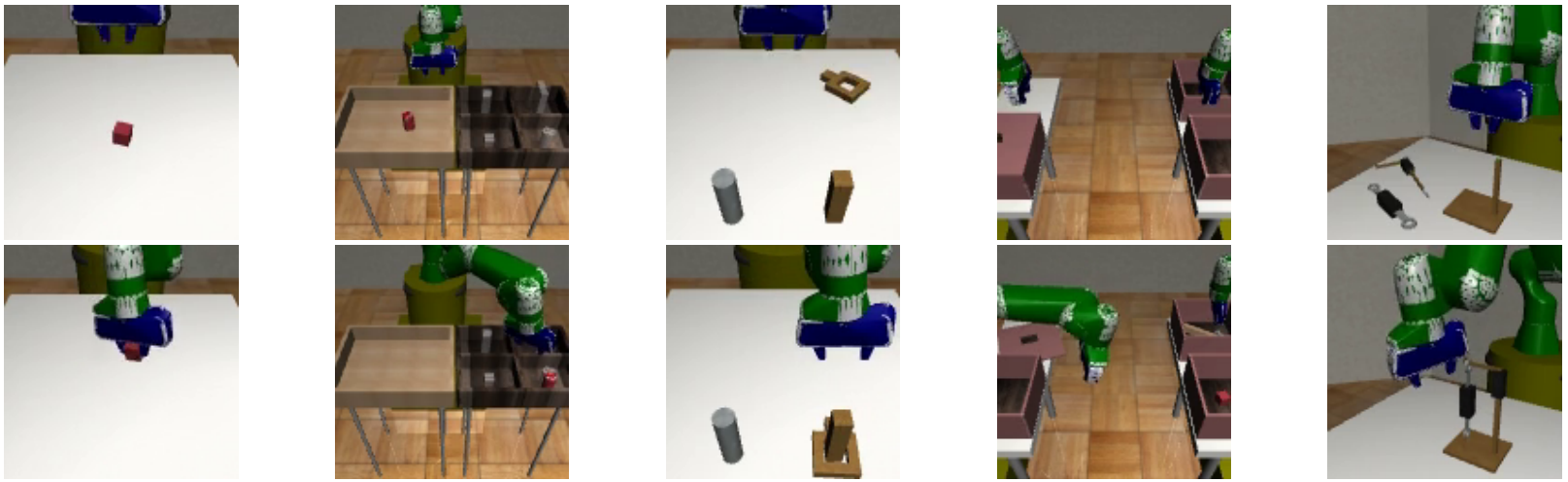
Table: FID of LFM vs. InterFlow [Albergo and Vanden-Eijnden, 2023].

	Pre-distillation	Distilled @ 4 NFEs	Distilled @ 2 NFEs
LFM	59.7	**71.0**	**75.2**
InterFlow	59.7	80.0	82.4

Table: FID comparison of LFM vs. FM [Lipman et al., 2023] and K-Rectified Flow [Liu et al., 2023].

	LFM	1-Rectified Flow (FM)	2-Rectified Flow	3-Rectified Flow
Pre-distillation	55.7	55.7	62.3	65.4
Distilled @ 4 NFEs	**76.9**	84.6	82.8	82.7

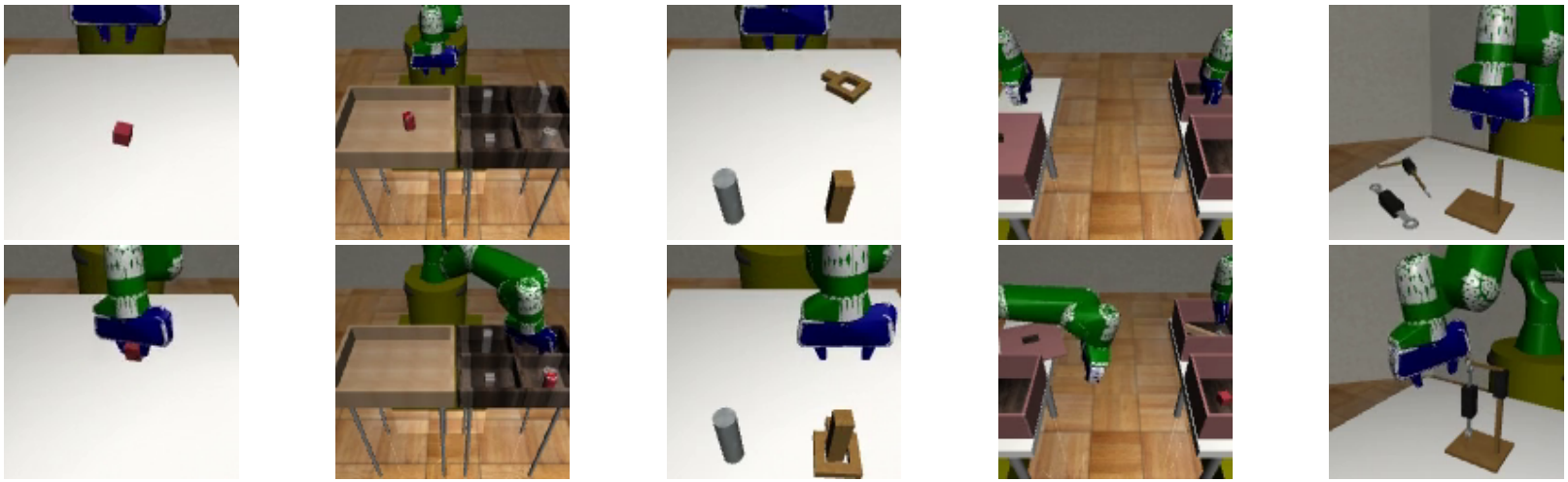
Part II: Empirical Performances (Flow Matching)



(a) Lift **(b)** Can **(c)** Square **(d)** Transport **(e)** Toolhang

Figure: Initial (top row) and final success (bottom row) per task.

Part II: Empirical Performances (Flow Matching)



(a) Lift **(b)** Can **(c)** Square **(d)** Transport **(e)** Toolhang

Figure: Initial (top row) and final success (bottom row) per task.

Table: Success rate comparison for robotic manipulation. Both methods are evaluated over 100 rollouts and success rates are reported at different epochs in the format of (Success rate, epochs).

	Lift	Can	Square	Transport	Toolhang
FM	(1.00, 200)	(0.94, 200) (0.98, 500)	(**0.88**, 200) (**0.94**, 750)	(0.60, 200) (0.81, 1500)	(0.52, 200)
LFM	(1.00, 200)	(**0.97**, 200) (**0.99**, 500)	(0.87, 200) (0.93, 750)	(**0.75**, 200) (**0.88**, 1500)	(**0.53**, 200)

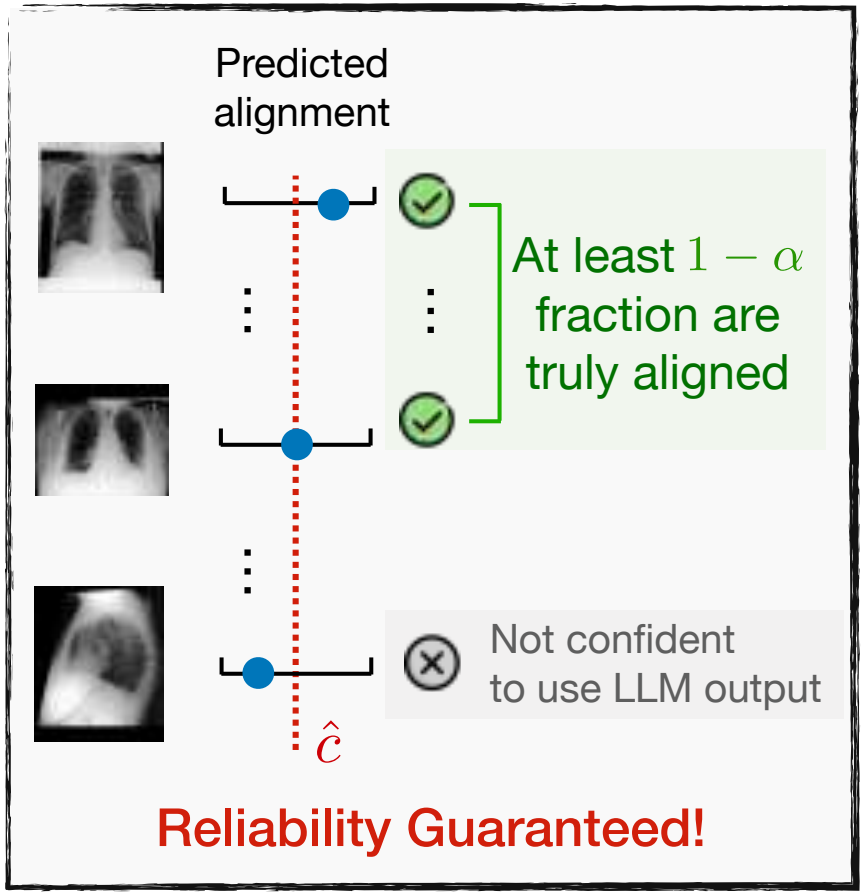
Outline

- Part I: Conformal prediction for time-series
- Part II: Flow-based generative models
- Future work: Connecting Parts I & II

Safe and reliable generative models via CP

Q: How to ensure generative models give reliable or human-preference-aligned outputs?

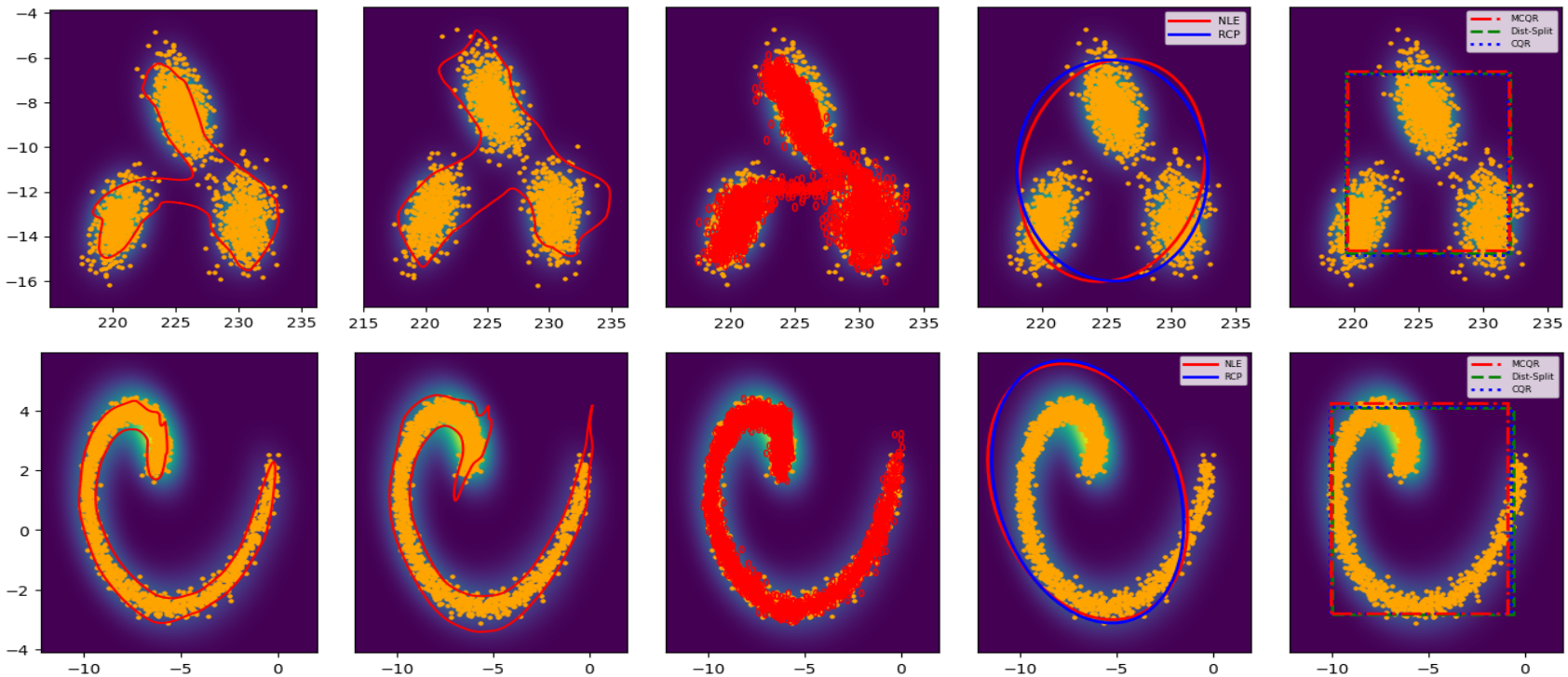
(a) For image generative model [Horwitz & Hoshen, 2022]



(b) For foundation models [Gui et al., 2024]

General-shape prediction regions

Q: How to generate such regions beyond pre-scribed shapes (e.g., ellipsoids or hypercubes)?



(a) CONTRA (b) ResCONTRA (c) PCP (d) Ellipses (e) Rectangles

(a) CONTRA (using NFlow with NLL as the non-conformity score) [anonymous submission].

Conclusion

- Part I: Extended conformal prediction for time series to develop distribution-free uncertainty quantification methods for point predictors.

Conclusion

- Part I: Extended conformal prediction for time series to develop distribution-free uncertainty quantification methods for point predictors.
  (→) The proposed methods yield significantly smaller prediction regions than baseline methods without losing empirical coverage.

Conclusion

- Part I: Extended conformal prediction for time series to develop distribution-free uncertainty quantification methods for point predictors.
  (→) The proposed methods yield significantly smaller prediction regions than baseline methods without losing empirical coverage.
- Part II: Proposed iterative training approaches for flow-based generative models, suitable for implicit MLE training or explicit FM training.

Conclusion

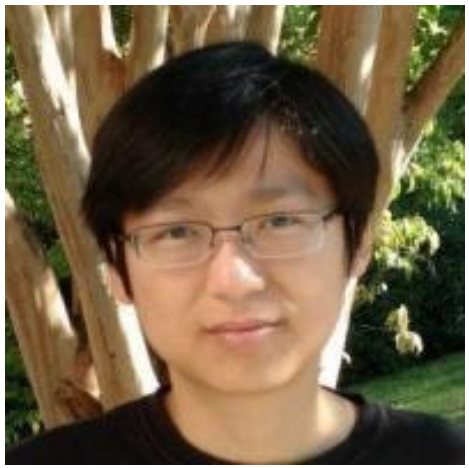
- Part I: Extended conformal prediction for time series to develop distribution-free uncertainty quantification methods for point predictors.

  (→) The proposed methods yield significantly smaller prediction regions than baseline methods without losing empirical coverage.

- Part II: Proposed iterative training approaches for flow-based generative models, suitable for implicit MLE training or explicit FM training.

  (→) Our models allow efficient generation of novel samples and accurate density estimation in NLL.

Acknowledgment (Alphabetical by last name)



(a) Dr. Xiuyuan Cheng

(b) Dr. Johannes Milz

(c) Dr. Arkadi Nemirovski

(d) Dr. Feng Qiu

(e) Dr. Yao Xie

Gratitude to my family

(a) My parents and grandma

(b) My wife

Publications (covered in defense)

In chronological order:

1. Chen Xu, Xiuyuan Cheng, and Yao Xie. Local Flow Matching generative models. *Preprint*. 2024.
2. Chen Xu, Hanyang Jiang, and Yao Xie. Conformal prediction for multi-dimensional time series by ellipsoidal sets. *ICML 2024 (spotlight)*.
3. Chen Xu, Xiuyuan Cheng, and Yao Xie. Normalizing flow neural networks by JKO scheme. *NeurIPS 2023 (spotlight)*.
4. Chen Xu and Yao Xie. Sequential predictive conformal inference for time series. *ICML 2023*.
5. Chen Xu and Yao Xie. Conformal prediction for time series. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2023.
6. Chen Xu and Yao Xie. Conformal prediction interval for dynamic time series. *ICML 2021 (oral)*.