Conditional Generation of BFNs and its Advantages over Unconditional BFNs

Dec 24, 2025

Introduction

BFNs is a novel generative model class that iteratively refines the parameters of independent distributions using Bayesian inference, guided by noisy data samples and neural network outputs. It combines the strengths of:

- **Bayesian inference**: Optimal information integration for individual variables.
- **Neural networks**: Contextual understanding across variables.

Compared with Diffusion Models: BFNs operate on distribution parameters rather than noisy data, ensuring continuity even for discrete data. This avoids the need for a predefined forward process.

Key idea: Bayesian updates + neural networks for iterative refinement.

Introduction

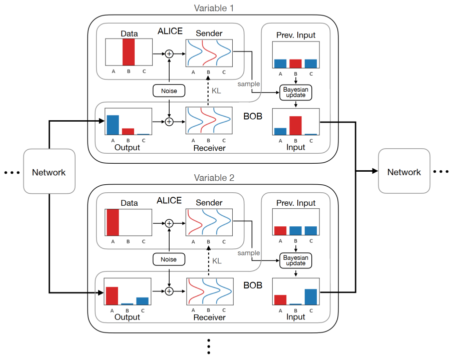


Figure: Overview of Bayesian Flow Networks (from original paper)

Introduction

The figure represents one step of the modelling process of a Bayesian Flow Network.

- The data is a ternary symbol sequence (e.g., 'B' and 'A').
- At each step, the network emits output distribution parameters based on previous input distribution.
- **Sender/Receiver distributions**: Continuous even for discrete data, created by adding noise.
- **Bayesian Update**: A sample from the sender is used to update input distribution parameters.
- **Loss function**: KL divergence from the receiver to the sender distribution (message from Alice to Bob).

Input and Sender Distributions

Given D-dimensional data $\mathbf{x} = (x^{(1)}, \ldots, x^{(D)}) \in \mathcal{X}^D$, let $c \in \mathcal{C}$ be the conditional variable. Let

$$\theta = (\theta^{(1)}, \ldots, \theta^{(D)})$$

be the parameters of a factorised **input distribution** $p_I(\cdot|\theta)$:

$$p_I(\mathbf{x}|\theta) = \prod_{d=1}^{D} p_I(x^{(d)}|\theta^{(d)}) \tag{1}$$

Let $p_S(\cdot|\mathbf{x};\alpha)$ be a similarly factorized **sender distribution** with $\mathbf{y} = (y^{(1)}, \ldots, y^{(D)}) \in \mathcal{Y}^D$:

$$p_S(\mathbf{y}|\mathbf{x};\alpha) = \prod_{d=1}^{D} p_S(y^{(d)}|x^{(d)};\alpha) \tag{2}$$

where $\alpha \in \mathbb{R}^+$ is an accuracy parameter.

Output Distributions $p_O(\cdot|\theta, c, t)$

Input parameters θ, processing time $t \rightarrow$ a neural network Ψ.

Output: $\Psi(\theta, c, t) = (\Psi^{(1)}(\theta, c, t), \ldots, \Psi^{(D)}(\theta, c, t))$ to parameterize an **output distribution**:

$$p_O(\mathbf{x}|\theta, c, t) = \prod_{d=1}^{D} p_O(x^{(d)}|\Psi^{(d)}(\theta, c, t)) \tag{3}$$

Receiver Distribution $p_R(\cdot|\theta, t, c, \alpha)$

Given sender distribution $p_S(\cdot|\mathbf{x}; \alpha)$ and output distribution $p_O(\cdot|\theta, c, t)$, the **receiver distribution** over $\mathcal{Y}^D$ is defined as:

$$p_R(\mathbf{y}|\theta, c; t, \alpha) = \mathbb{E}_{p_O(\mathbf{x}'|\theta, c; t)}[p_S(\mathbf{y}|\mathbf{x}'; \alpha)] \tag{4}$$

Bayesian Updates

Given parameters θ and sender sample $\mathbf{y}$ drawn with accuracy α, the **Bayesian update function** h computes θ':

$$\theta' \leftarrow h(\theta, \mathbf{y}, \alpha) \tag{5}$$

The **Bayesian update distribution** $p_U(\cdot|\theta, \mathbf{x}; \alpha)$ is defined by marginalizing out $\mathbf{y}$:

$$p_U(\theta'|\theta, \mathbf{x}; \alpha) = \mathbb{E}_{p_S(\mathbf{y}|\mathbf{x};\alpha)}[\delta(\theta' - h(\theta, \mathbf{y}, \alpha))] \tag{6}$$

where $\delta(\cdot - \mathbf{a})$ is the multivariate Dirac delta distribution.

Bayesian Updates

Accuracy is additive: if $\alpha = \alpha_a + \alpha_b$, then

$$p_U(\theta''|\theta, \mathbf{x}; \alpha) = \mathbb{E}_{p_U(\theta'|\theta, \mathbf{x}; \alpha_a)}[p_U(\theta''|\theta', \mathbf{x}; \alpha_b)] \tag{7}$$

Probability of observing θ_n after n samples with accuracies $\alpha_1, \dots, \alpha_n$:

$$\int \cdots \int p_U(\theta_n|\theta_{n-1}, \mathbf{x}; \alpha_n) \dots p_U(\theta_1|\theta_0, \mathbf{x}; \alpha_1) = p_U\left(\theta_n \middle| \theta_0, \mathbf{x}; \sum_{i=1}^{n} \alpha_i\right) \tag{8}$$

Accuracy Schedule $\beta(t)$

For continuous time $t \in [0, 1]$, let $\alpha(t) > 0$ be the accuracy rate. Define **accuracy schedule** $\beta(t)$:

$$\beta(t) = \int_0^t \alpha(t')dt' \tag{9}$$

$\beta(t)$ is monotonically increasing, $\beta(0) = 0$, and:

$$\frac{d\beta(t)}{dt} = \alpha(t) \tag{10}$$

Bayesian Flow Distribution $p_F(\cdot|\mathbf{x}; t)$

The **Bayesian flow distribution** $p_F(\cdot|\mathbf{x}; t)$ is the marginal distribution over input parameters at time t:

$$p_F(\theta|\mathbf{x}; t) = p_U(\theta|\theta_0, \mathbf{x}; \beta(t)) \tag{11}$$

Loss Function $L(\mathbf{x}|c)$

Consider n sender samples at times $t_i = i/n$. Accuracy at step i:

$$\alpha_i = \beta(t_i) - \beta(t_{i-1}) \tag{12}$$

Input parameter sequence is recursively updated:

$$\theta_i = h(\theta_{i-1}, \mathbf{y}, \alpha_i) \tag{13}$$

Loss Function $L(\mathbf{x}|c)$

n-step discrete-time loss $L^n(\mathbf{x})$:

$$L^n(\mathbf{x}|c) = \mathbb{E}_{p(\theta_1,\ldots,\theta_{n-1})} \sum_{i=1}^{n} D_{KL}(p_S(\cdot|\mathbf{x};\alpha_i) \| p_R(\cdot|\theta_{i-1}, c; t_{i-1}, \alpha_i)) \quad (14)$$

where

$$p(\theta_1, \ldots, \theta_n) = \prod_{i=1}^{n} p_U(\theta_i|\theta_{i-1}, \mathbf{x}; \alpha_i) \quad (15)$$

Loss Function $L(\mathbf{x}|c)$

Reconstruction loss:

$$L^r(\mathbf{x}|c) = -\mathbb{E}_{p_F(\theta|\mathbf{x},1)} \ln p_O(\mathbf{x}|\theta, c; 1) \tag{16}$$

Total loss:

$$L(\mathbf{x}|c) = L^n(\mathbf{x}|c) + L^r(\mathbf{x}|c) \tag{17}$$

A Trick in Computing Discrete-Time Loss $L^n(\mathbf{x}|c)$

Equation (14) can be rewritten using expectations over $i \sim U\{1, n\}$ and $\theta \sim p_F$:

$$L^n(\mathbf{x}|c) = n\mathbb{E}_{i\sim U\{1,n\}, p_F(\theta|\mathbf{x};t_{i-1})} D_{KL}(p_S(\cdot|\mathbf{x};\alpha_i) \| p_R(\cdot|\theta, c; t_{i-1}, \alpha_i)) \quad (18)$$

This allows **Monte-Carlo sampling** without computing the full n-step sum.

MNIST Sample Generation: Unconditional vs. Conditional BFN

Figure: Samples in original paper (10,000 steps)

Conditional BFN (100 steps)

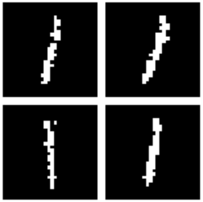


Figure: 1

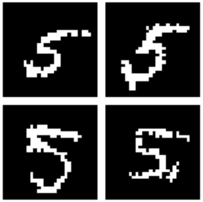


Figure: 5

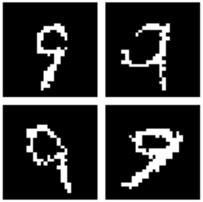
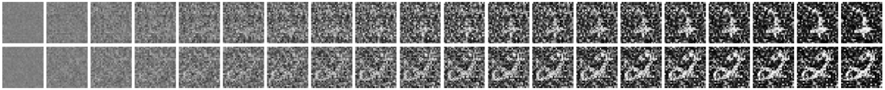
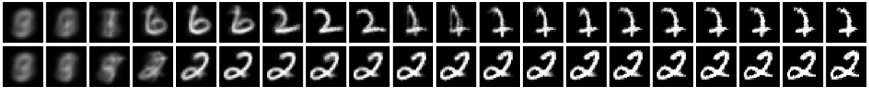


Figure: 9

MNIST Input and output distributions



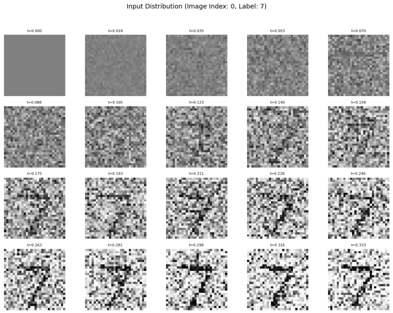
(a) Input Distribution



(b) Output Distribution

Figure: Input and output distributions from original paper's unconditional BFN for a test image

MNIST Input and output distributions



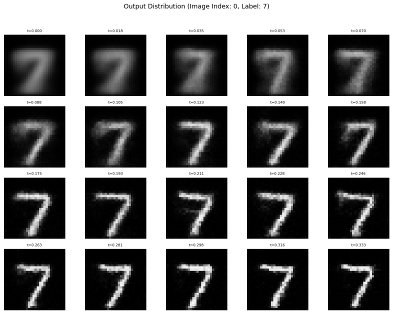


Figure: Input and output distributions from our conditionally trained BFN for a test image with a fixed label

Conditional Generative Models: Setup

Spaces and Measures:

- Data space $(\mathcal{X}, \mathcal{F}_X)$, condition space $(\mathcal{C}, \mathcal{F}_C)$
- Joint measure $\mu_{X,C}$ with marginal μ_C and conditional measures $\mu_{X|C=c}$

Model Families:

- Unconditional model family: $\mathcal{P}$
- Conditional model family:

$$\tilde{\mathcal{P}} = \{\nu_{X|C} : \forall c, \nu_{X|C}(\cdot|c) \in \mathcal{P}\}$$

Divergence: $D(\cdot\|\cdot)$

- Convex in first argument
- Non-negative and integrable

Conditional Generative Models: Theorem

Assumptions:

1. *Conditional distributions easier to approximate:*

$$\inf_{\nu_X} D(\mu_{X|C=c} \| \nu_X) \leq \inf_{\nu_X} D(\mu_X \| \nu_X), \quad \text{a.e. } \mu_C$$

2. *Independent conditional choice:* $\forall c \in \mathcal{C}$, $\arg\min_{\nu_X \in \mathcal{P}} D(\mu_{X|C=c} \| \nu_X)$ exists.
3. *Measurability & integrability:*

$$\int_{\mathcal{C}} D(\mu_{X|C=c} \| \nu_{X|C=c}) \, \mu_C(dc) < \infty$$

Theorem:

$$\inf_{\nu_{X|C} \in \tilde{\mathcal{P}}} \int_{\mathcal{C}} D(\mu_{X|C=c} \| \nu_{X|C=c}) \, \mu_C(dc) \;\leq\; \inf_{\nu_X \in \mathcal{P}} D(\mu_X \| \nu_X)$$

Intuition: splitting a complex distribution into simpler conditionals reduces the average divergence, making conditional models typically better.

References

- Graves, A., Srivastava, R. K., Atkinson, T., and Gomez, F. (2023). *Bayesian flow networks*.

Thank You!