
Self-sufficient Independent Component Analysis via KL Minimizing Flows

Song Liu
University of Bristol

Abstract

We study the problem of learning disentangled signals from data using non-linear Independent Component Analysis (ICA). Motivated by advances in self-supervised learning, we propose to learn self-sufficient signals: A recovered signal should be able to reconstruct a missing value of its own from all remaining components without relying on any other signals. We formulate this problem as the minimization of a conditional KL divergence. Compared to traditional maximum likelihood estimation, our algorithm is prior-free and likelihood-free, meaning that we do not need to impose any prior on the original signals or any observational model, which often restricts the model’s flexibility. To tackle the KL divergence minimization problem, we propose a sequential algorithm that reduces the KL divergence and learns an optimal de-mixing flow model at each iteration. This approach completely avoids the unstable adversarial training, a common issue in minimizing the KL divergence. Experiments on toy and real-world datasets show the effectiveness of our method.

1 INTRODUCTION

Learning disentangled features from the data has been extensively studied in recent years (Dinh et al., 2015; Chen et al., 2016; Hyvarinen et al., 2023; Wang et al., 2024). Assuming the observed signals are independent components transformed by an invertible mixing function, Independent Component Analysis (ICA) learns a de-mixing function that recovers the original independent signals. Linear ICA assumes the mixing function is linear (Amari et al., 1995; Bell and Sejnowski, 1995). However, extending linear ICA to a non-linear setting is non-trivial, as the independence assumption alone is not enough to identify the original signals.

Many methods have been proposed to enhance the identifiability by exploiting additional assumptions, such as autocorrelation (Hyvarinen and Morioka, 2017) or non-stationarity (Hyvarinen and Morioka, 2016) in the original signal, and conditional independence given an observed auxiliary variable (Hyvarinen et al., 2019; Khemakhem et al., 2020).

In this paper, we employ two of the latest ideas in representation learning and generative modelling to develop a novel ICA algorithm. Self-supervised learning learns a representation by masking partial data and trains a model to recover masked values with the remaining components (He et al., 2022; Shi et al., 2022; Tashiro et al., 2021). This idea inspires us to propose Self-sufficient Independent Component Analysis (SICA). We assume that the original signals are self-sufficient, i.e., we can reconstruct a missing value from the remaining components without relying on the other signals. We demonstrate that this sufficiency assumption naturally translates into a factorization condition for densities, which can be leveraged to train a de-mixing function.

Existing ICA methods train the de-mixing function using maximum likelihood estimation or maximising the ELBO. However, these methods require explicit prior and likelihood models. Overly complicated models make the likelihood function intractable, while simple ones restrict the model’s flexibility. The other branch of methods minimizes the mutual information between recovered signals. However, estimating the mutual information is not a trivial task, and existing methods often require adversarial training (Brakel and Bengio, 2017) of a neural mutual information estimator (Belghazi et al., 2018) together with the de-mixing function, which can be numerically unstable.

In recent years, ODE-based generative models have achieved remarkable success (Chen et al., 2018; Lipman et al., 2023; Liu et al., 2023; Yi et al., 2023). These methods train Ordinary Differential Equations (ODEs) to transport samples from a reference distribution to match a target distribution. Inspired by

this idea, we train ODEs as de-mixing functions (referred to as “de-mixing flows”). We learn an optimal de-mixing flow by enforcing the sufficiency mentioned above on the learned signals, resulting in a KL divergence minimization problem. To avoid adversarial training, we propose learning the de-mixing flow iteratively, thereby reducing the KL divergence at each iteration. Our approach does not impose any prior or likelihood model assumption, thus it is more flexible than likelihood-based approaches.

This paper is organized as follows: Section 2 clarifies the common independence assumptions used by classic ICAs, introduces our new SICA assumptions, and discusses the density factorization they imply. In Section 3, we detail under SICA assumptions, how to learn de-mixing functions using flow-based methods. Finally, in Section 4, we demonstrate that the proposed ICA algorithm achieves promising performance on both autoregressive and image datasets.

2 ICA AND SELF-SUFFICIENT ICA

Notations: $x, \mathbf{x}, \mathbf{X}$ are scalar, vector and matrix. $\mathbf{x}, \mathbf{x}, \mathbf{X}$ are random scalars, vectors and matrices. $p(\mathbf{x})$ is the probability of random variable $\mathbf{x}$ and $p(\mathbf{x} = x)$ is the probability density function of random variable $\mathbf{x}$ evaluated at x . $\mathbf{X}_{i,:}$ represents the i -th row of a matrix $\mathbf{X}$. $\mathbf{X}_{-i,:}$ represents all rows of $\mathbf{X}$ except the i -th row. $\tilde{\mathbf{X}}_{-i,:}$ represents all rows of a matrix $\mathbf{X}$ but the i -th row is replaced with a vector of missing values NaN.

2.1 Independent Component Analysis

The task of Independent Component Analysis (ICA) assumes that some original signal $\mathbf{s} \in \mathbb{R}^d$ is transformed through an invertible, possibly non-linear mixing function $\mathbf{f} : \mathbb{R}^d \rightarrow \mathbb{R}^d$:

$$\mathbf{x} = \mathbf{f}(\mathbf{s}), \quad (1)$$

where $\mathbf{s} = [s_1, \dots, s_d]$ is a vector of *independent components* and $\mathbf{x}$ is the observed signal. The task of ICA is to learn a de-mixing function $\mathbf{g} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that

$$\mathbf{z} := \mathbf{g}(\mathbf{x}). \quad (2)$$

The optimal $\mathbf{g}$ recovers the original signal, so $\mathbf{z} = \mathbf{s}$. If we restrict $\mathbf{g}$ in the family of bijective linear functions, we recover the linear ICA problem.

Most ICA algorithms look for a $\mathbf{g}$ such that all components in $\mathbf{z}$ are statistically independent. This is achieved mainly in two ways: maximum likelihood estimation or direct minimization of mutual information (Hyvarinen and Oja, 2000). For example, Nonlinear Independent Component Estimation (NICE) (Dinh

et al., 2015) finds $\mathbf{g}$ by maximizing the likelihood function constructed using *independent priors*. Least-squares ICA (LICA) (Suzuki and Sugiyama, 2011) finds $\mathbf{g}$ by minimizing the least-squares mutual information (or total correlation) between components of $\mathbf{z}$, i.e., $D[p(\mathbf{z}) \parallel \prod_{i=1}^d p(z_i)]$. LICA approximates the mutual information using density ratio estimation (Sugiyama et al., 2012). Non-linear Adversarial ICA (Brakel and Bengio, 2017) formulates the mutual information minimization problem as an adversarial training problem, and learns a non-linear transform $\mathbf{g}$.

2.2 Sufficiently Independent Component Analysis

The independence assumption in the classic ICA can be overly stringent. Consider the observed signal $\mathbf{x}$ generated using the following formula:

$$\begin{aligned} u &\sim \text{uniform}(0, 2\pi), s_2 := h_1(u) + \epsilon, s_1 := h_2(u) + \epsilon', \\ \mathbf{x} &= \begin{bmatrix} 1, & .5 \\ .5, & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \end{aligned} \quad (3)$$

where ϵ, ϵ' are some random independent noise. $\mathbf{x}$ is a linear mixture of s_1 and s_2 . ICA cannot recover s_1 and s_2 perfectly using $\mathbf{x}$ alone, as the independence assumption in ICA does not hold. s_1 and s_2 are not independent due to their associations with u . See Figure 1 for an example of such type of dataset.

To address this issue, Hyvarinen et al. proposes auxiliary variable ICA to generalise the independence assumption. In the above case, although s_1 and s_2 are not independent, they are conditionally independent given u . ICA with auxiliary variable assumes that $\mathbf{s} = [s_1, \dots, s_d]$ are conditionally independent given an observed auxiliary variable $\mathbf{u}$. This work has been further developed to handle structural temporal dependency in $\mathbf{u}$ (Halva et al., 2021). However, auxiliary variable ICA requires observing the common variable $\mathbf{u}$. In many cases, such auxiliary information is difficult to obtain.

In this paper, we study a specific setting of auxiliary-variable ICA. We assume that each component s_i comes with its own auxiliary variable, $\mathbf{u}_i$, that is *sufficient* in predicting s_i . By sufficiency, we mean that knowing information from any other pair $(s_j, \mathbf{u}_j), j \neq i$ will not improve the prediction of s_i . For example, if s_0 is one pixel on an image and $\mathbf{u}_0$ is the remaining pixels on the same image, then we assume that knowing the remaining pixels $\mathbf{u}_0$ is sufficient in predicting s_0 . Information from the other image $(s_1, \mathbf{u}_1)$ can not help us predict s_0 better.

Specifically, let pairs of signals and auxiliary variables

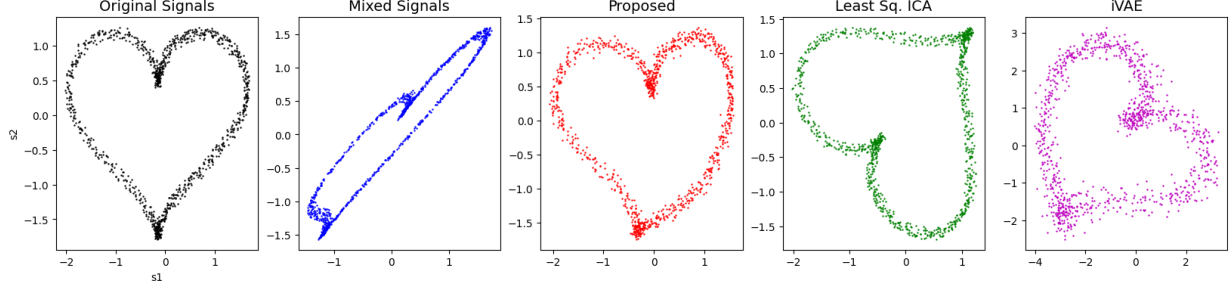


Figure 1: Two dependent signals mixed by a linear mixing function. In this case, both a linear ICA (LICA, Suzuki and Sugiyama (2011)) and a non-linear ICA (iVAE Khemakhem et al. (2020)) fail to de-mix the signals (the heart is still tilted), whereas the proposed method, SICA, successfully de-mixes the two signals.

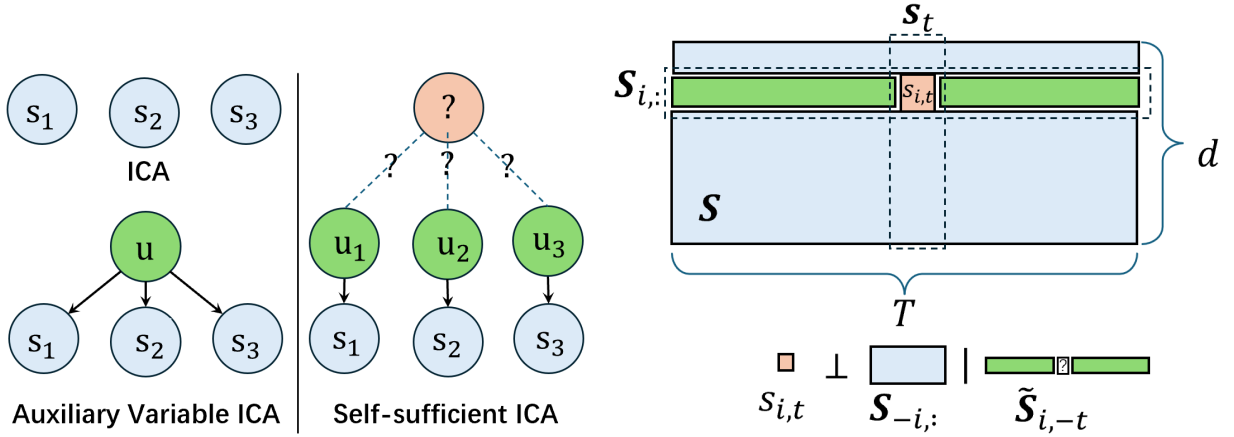


Figure 2: Left: Graphical models of ICA, Auxiliary Variable ICA and SICA. Right: SICA assumption on Sequence data.

be $(s_i, \mathbf{u}_i)$. We assume that

$$s_i \perp (\mathbf{s}_{-i}, \mathbf{u}_{-i}) | \mathbf{u}_i. \quad (4)$$

The independence assumption in equation 4 is equivalent to the following density factorization

$$p(\mathbf{s} | \mathbf{U}) = \prod_i p(s_i | \mathbf{u}_i), \quad (5)$$

where $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_d]^\top$. The proof involves applying the chain rule for densities; details can be found in Appendix A.1. One can see that this condition is stronger than the auxiliary variable ICA: The factorization in equation 5 implies the conditional independence of s_i and $\mathbf{s}_{-i}$ given $\mathbf{u}$, but not vice versa. Moreover, this assumption is weaker than the unconditional independence, as $(s_i, \mathbf{u}_i) \perp (\mathbf{s}_{-i}, \mathbf{u}_{-i})$ implies equation 4, but not vice versa.

We refer to this property as “self-sufficiency”, and call the algorithm that recovers these sufficiently independent signals as Self-sufficient Independent Component Analysis (SICA). We compare the ICA, Auxiliary Variable ICA, and SICA assumptions in the left illustration in Figure 2.

The next section shows how this assumption manifests in multi-dimensional sequence data.

2.3 Sequence SICA

Consider a random, multi-dimensional signal $\mathbf{s}_t$, indexed by $t \in [T]$. We define $\mathbf{S} := [\mathbf{s}_t]_{t \in [T]} \in \mathbb{R}^{d \times T}$, where $\mathbf{s}_t = [s_{1,t}, \dots, s_{d,t}]^\top$ is the t -th column of $\mathbf{S}$.

Let us define the auxiliary variable

$$\mathbf{u}_t = \tilde{\mathbf{S}}_{:, -t} := [\mathbf{s}_1 \dots \mathbf{s}_{t-1}, \text{NaN}, \mathbf{s}_{t+1}, \dots, \mathbf{s}_T].$$

$\tilde{\mathbf{S}}_{:, -t}$ is the same as $\mathbf{S}$ except its t -th column is replaced by a vector of missing values. Then SICA assumption described in equation 4 can be expressed as

$$s_{i,t} \perp \mathbf{S}_{-i,:} | \tilde{\mathbf{S}}_{i,-t}, t \in [T]. \quad (6)$$

It means at time t , given the remainder of i -th signal and the position of the missing value, the i -th signal is independent from any other signals at any time. See the right plot in Figure 2 for a visualization of equation 6.

Algorithm 1 Iterative KL minimization Algorithm

```

1: Input: A mixed multi-dimensional sequence  $\mathbf{X}$ 
2: Let  $\mathbf{Z}^{(0)} = \mathbf{X}$ 
3: for  $j = 1 \dots J$  do
4:    $\mathbf{l}^{(j)} := \arg \min_{\mathbf{l} \in \mathcal{L}} \sum_{t \in [T]} D_{\text{KL}}^{(j-1)}(\mathbf{l})$ 
5:    $\mathbf{Z}^{(j)} := \mathbf{l}^{(j)}(\mathbf{Z}^{(j-1)})$ 
6: end for
7: return  $\mathbf{Z}^{(J)}$ 
    
```

Equivalently, we can express the assumption as factorizations of density functions, as shown in equation 5

$$p(\mathbf{s}_t | \tilde{\mathbf{S}}_{:, -t}) = \prod_{i=1}^d p(s_{i,t} | \tilde{\mathbf{S}}_{i, -t}), t \in [T]. \quad (7)$$

Following the standard ICA setting, we assume that,

$$\mathbf{x}_t = \mathbf{f}(\mathbf{s}_t), t \in [T]$$

and we want to learn $\mathbf{g}$ such that

$$\mathbf{z}_t := \mathbf{g}(\mathbf{x}_t), t \in [T].$$

$\mathbf{g}$ is optimal iff $\mathbf{z}_t = \mathbf{s}_t, \forall t$.

One can easily extend t to multi-dimensional indices, which enables applications in image or video processing and spatial data analysis. In the rest of the paper, we focus on SICA applied to sequence data.

3 LEARNING DE-MIXING FLOWS

In this section, we propose numerical methods that learns de-mixing flows.

Let $\mathbf{Z} = [\mathbf{z}_t, \forall t]$. The SICA assumption in equation 7 suggests a learning criterion of minimizing the KL divergence

$$\mathbf{g}^* := \arg \min_{\mathbf{g}} \sum_{t \in [T]} D_{\text{KL}} \left[p(\mathbf{z}_t | \tilde{\mathbf{Z}}_{:, -t}) \left\| \prod_{i=1}^d p(z_{i,t} | \tilde{\mathbf{Z}}_{i, -t}) \right\| \right], \quad (8)$$

where $\tilde{\mathbf{Z}}_{:, -t}$ is similarly defined as $\tilde{\mathbf{S}}_{:, -t}$. This objective enforces the factorization of conditional distributions of the recovered signal $\mathbf{Z}$. One may adopt the adversarial training scheme, similar to (Brakel and Bengio, 2017), to minimize the divergence. However, adversarial optimization is known to be numerically challenging.

Instead of directly minimize equation 8, we propose an iterative approach to refine $\mathbf{z}_t$ gradually. At iteration j , we learn a refinement function $\mathbf{l}^{(j)}$ by minimizing

the following surrogate objective,

$$\mathbf{l}^{(j)} := \arg \min_{\mathbf{l} \in \mathcal{L}} \sum_{t \in [T]} D_{\text{KL}}^{(j-1)}(\mathbf{l})$$

$$D_{\text{KL}}^{(j-1)}(\mathbf{l}) := D_{\text{KL}} \left[p(\mathbf{z}_t | \tilde{\mathbf{Z}}_{:, -t}^{(j-1)}) \left\| \prod_{i=1}^d p(\mathbf{z}_{i,t}^{(j-1)} | \tilde{\mathbf{Z}}_{i, -t}^{(j-1)}) \right\| \right], \quad (9)$$

where we define

$$\mathbf{z}_t := \mathbf{l}(\mathbf{z}_t^{(j-1)}; \tilde{\mathbf{Z}}_{:, -t}^{(j-1)}). \quad (10)$$

The refinement function $\mathbf{l}$ is invertible with respect to the first argument. Notice that equation 9 is similar to equation 8, but only $\mathbf{z}_t$ depends on the refinement function $\mathbf{l}$. Other random variables are carried over from the $j-1$ -th iteration.

Isolating the random variables that depend on $\mathbf{l}$ is a crucial step toward deriving a tractable minimization algorithm of the KL divergence in equation 8, as we will see in the next section. The detailed algorithm is provided in Algorithm 1.

Finally, after the algorithm terminates, we obtain $\mathbf{g} := \bigcirc_{j=1}^J \mathbf{l}^{(j)}$, i.e., the de-mixing function $\mathbf{g}$ that is the composite of all previously learned refinements.

We prove that Algorithm 1 reduces the KL divergence over iterations:

Theorem 3.1. *If $\mathbf{l}$ is invertible as defined above, the KL divergence $D_{\text{KL}}^{(j)}(\mathbf{l}^{(j)})$ at the end of each iteration is non-increasing, i.e.,*

$$D_{\text{KL}}^{(j)}(\mathbf{l}^{(j)}) \leq D_{\text{KL}}^{(j-1)}(\mathbf{l}^{(j-1)}), \forall j = 1 \dots J.$$

The proof is included in the Appendix A.2.

In this paper, we assume that we only observe one sample at each t . To help us build the de-mixing flow learning algorithm, we make the following assumption.

Assumption 3.2.

$$p(\mathbf{z}_t | \tilde{\mathbf{Z}}_{:, -t}^{(j-1)}) = p(\mathbf{z}_{t'} | \tilde{\mathbf{Z}}_{:, -t'}^{(j-1)}),$$

$$p(\mathbf{z}_{i,t}^{(j-1)} | \tilde{\mathbf{Z}}_{i, -t}^{(j-1)}) = p(\mathbf{z}_{i,t'}^{(j-1)} | \tilde{\mathbf{Z}}_{i, -t'}^{(j-1)}), \forall t, t' \in [T].$$

Assumption 3.2 assumes that the above conditional distributions are identical over the entire time period. This stationary condition paves the way for using samples at different time points to learn a de-mixing flow as we will see in the next sections.

3.1 Minimizing KL using Wasserstein Gradient Flow

As the solution described in this section applies to all j , we simplify notations $\mathbf{Z}^{(j-1)}$, $\mathbf{z}_t^{(j-1)}$ and $\mathbf{Z}_{:,t}^{(j-1)}$ as $\mathbf{Z}$, $\mathbf{z}_t$ and $\mathbf{Z}_{:,t}$ by omitting the superscripts.

To optimize equation 9, we need to choose the family $\mathcal{L}$. Inspired by the success of flow-based generative models, we construct the refinement function $\mathbf{l}$ using an ODE. Let $\mathbf{l}(\mathbf{y}; \tilde{\mathbf{Z}}_{:, -t})$ be the solution of an ODE using $\mathbf{y}$ as the initial value. The ODE evolves according to the following differential equation

$$\frac{d}{d\tau} \mathbf{y}(\tau) = \mathbf{v}(\mathbf{y}(\tau), \tilde{\mathbf{Z}}_{:, -t}, \tau). \quad (11)$$

Then, finding $\mathbf{l}$ translates into the problem of finding the vector field $\mathbf{v}$, and we show the vector field that minimises $D_{\text{KL}}(\mathbf{l})$ is a Wasserstein Gradient Flow (WGF) (Ambrosio et al., 2008).

Suppose $\mathbf{y}(\tau)$ is the solution of an ODE at time τ , with a random variable $\mathbf{y}(0)$ as its initial value. Let ρ_τ be the density of $\mathbf{y}(\tau)$, a curve of probability densities. If ρ_τ evolves along the steepest descent direction of $D_{\text{KL}}[\rho_\tau \|\mu]$, ρ_τ is a Wasserstein-2 gradient flow of the functional objective $D_{\text{KL}}[\rho_\tau \|\mu]$. The velocity field of such a gradient flow takes a closed form:

$$\mathbf{v}^*(\mathbf{y}) = -\nabla_{\mathbf{y}} \log \frac{\rho_\tau(\mathbf{y})}{\mu(\mathbf{y})}. \quad (12)$$

The proof of this result could be found in, e.g., Theorem 1.4.1, (Chewi, 2025).

Let $\mathbf{z}_t(\tau)$ be the solution to an ODE at time τ . Recall the KL divergence we are minimising in equation 9 is

$$D_{\text{KL}} \left[\underbrace{p(\mathbf{z}_t(\tau) | \tilde{\mathbf{Z}}_{:, -t})}_{\rho_\tau} \parallel \prod_{i=1}^d p(\mathbf{z}_{i,t} | \tilde{\mathbf{Z}}_{i,-t}) \right].$$

According to equation 12, the WGF vector field minimizes our KL divergence is

$$\mathbf{v}^*(\mathbf{y}; \tilde{\mathbf{Z}}_{:, -t}) := -\nabla_{\mathbf{y}} \log \frac{\rho_\tau(\mathbf{y} | \tilde{\mathbf{Z}}_{:, -t})}{\prod_i p_i(y_i | \tilde{\mathbf{Z}}_{i,-t})}, \quad (13)$$

where

$$\begin{aligned} \rho_\tau(\mathbf{y} | \tilde{\mathbf{Z}}_{:, -t}) &:= p(\mathbf{z}_t(\tau) = \mathbf{y} | \tilde{\mathbf{Z}}_{:, -t} = \tilde{\mathbf{Z}}_{:, -t}), \\ p_i(y_i | \tilde{\mathbf{Z}}_{i,-t}) &:= p(z_{i,t} = y_i | \tilde{\mathbf{Z}}_{i,-t} = \tilde{\mathbf{Z}}_{i,-t}). \end{aligned}$$

and the vector field $\mathbf{v}^*$

$$\mathbf{v}^* : \mathbb{R}^d \otimes \tilde{\mathbb{R}}^{T \times d} \rightarrow \mathbb{R}^d,$$

where $\tilde{\mathbb{R}} := \mathbb{R} \cup \{\text{NaN}\}$. Although one may use any density ratio estimation techniques (Sugiyama et al., 2012) to approximate the density ratio in equation 13, the ratio in equation 13 is a conditional density ratio, and estimating the conditional ratio requires conditional samples from the numerator and the denominator distributions, which are not available in our setting.

However, we can see that

$$\mathbf{v}^*(\mathbf{y}; \tilde{\mathbf{Z}}_{:, -t}) = -\nabla_{\mathbf{y}} \log \frac{\rho_\tau(\mathbf{y}, \tilde{\mathbf{Z}}_{:, -t})}{\prod_i p_i(y_i, \tilde{\mathbf{Z}}_{i,-t})}, \quad (14)$$

where both the numerator and denominator are joint densities defined similarly as the above. This is because the gradient is only taken with respect to $\mathbf{y}$, not the conditional variables. Hence, the gradient of the log conditional ratio is equivalent to the gradient of the log joint ratio. The joint ratio is much simpler to estimate, as it only requires joint samples that can be readily constructed from $\mathbf{Z}$.

Let $(\mathbf{z}'_{i,t}, \mathbf{Z}'_{i,-t})$ be an identical but independent copy of $(z_{i,t}, \mathbf{Z}_{i,-t})$. According to Assumption 3.2, the pairs are identically distributed across all t . Thus, we can construct sets of samples from both the numerator and the denominator, i.e., $\{(\mathbf{z}_t, \tilde{\mathbf{Z}}_{:, -t})\}_{t=1}^T \sim \rho_\tau$ and

$$\{(\mathbf{z}'_{1,t}, \tilde{\mathbf{Z}}'_{1,-t})\}_{t=1}^T \sim p_1, \dots, \{(\mathbf{z}'_{d,t}, \tilde{\mathbf{Z}}'_{d,-t})\}_{t=1}^T \sim p_d.$$

We can use these sets to estimate the joint ratio in equation 14 with any density ratio estimation method.

In practice, we do not have access to independent copies of $(z_{i,t}, \mathbf{Z}_{i,-t})$, so we create $\{(\mathbf{z}'_{i,t}, \mathbf{Z}'_{i,-t})\}_{t=1}^T$ by permuting samples in $\{(\mathbf{z}_{i,t}, \mathbf{Z}_{i,-t})\}_{t=1}^T$.

After obtaining $\mathbf{v}^*$, we solve the ODE in equation 11 using the Euler method. In experiments, we observe that even one step of the Euler update can already work effectively.

3.2 Minimising KL using Rectified Flow

On the other hand, if we do not require the trajectory of the density ρ_τ to take the steepest descent of the KL divergence, then any ODE that transports samples from the initial distribution $p(\mathbf{z}_t | \tilde{\mathbf{Z}}_{:, -t})$ to the target $\prod_i p(\mathbf{z}_{i,t} | \tilde{\mathbf{Z}}_{i,-t})$ would be optimal as it reduces the KL divergence to zero. This relaxation opens the door to using other types of distribution matching flows, such as Rectified Flow (Liu et al., 2023).

Let $\mathbf{y}(0)$ and $\mathbf{y}(1)$ be two random variables. The rectified flow trains an ODE that transports samples from the reference $p(\mathbf{y}(0))$ to the target $p(\mathbf{y}(1))$, and the vector field is trained by minimizing the following least squares objective:

$$\begin{aligned} \mathbf{v}^* &:= \arg \min_{\mathbf{v}} \int_0^1 \mathbb{E}[\|\mathbf{y}(1) - \mathbf{y}(0) - \mathbf{v}(\mathbf{y}(\tau), \tau)\|^2] d\tau, \\ \mathbf{y}(\tau) &= \tau \mathbf{y}(1) + (1-\tau) \mathbf{y}(0). \end{aligned} \quad (15)$$

One can prove that solving an ODE using $\mathbf{v}^*$ as the vector field and samples of $\mathbf{y}(0)$ as initial points, will

transport samples from $p(\mathbf{y}(0))$ to the target distribution $p(\mathbf{y}(1))$ (see Theorem 3.3 in Liu et al. (2023)).

Recall that our task is to transport samples from the conditional distribution $p(\mathbf{z}_t|\tilde{\mathbf{Z}}_{:, -t})$ to the target $\prod_i p(z_{i,t}|\tilde{\mathbf{Z}}_{i, -t})$. To achieve this, we propose the following variant of the rectified flow objective. Define

$$\begin{aligned}\mathbf{y}(0) &= [\mathbf{z}_t, \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}], \\ \mathbf{y}(1) &= [\mathbf{z}'_t, \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}], \\ \mathbf{y}(\tau) &= [\mathbf{z}_t(\tau), \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}].\end{aligned}$$

where

$$(\mathbf{z}'_t, \tilde{\mathbf{Z}}'_{:, -t}) := \left(\begin{bmatrix} z'_{1,t} \\ \vdots \\ z'_{d,t} \end{bmatrix}, \begin{bmatrix} \tilde{\mathbf{Z}}'_{1, -t} \\ \vdots \\ \tilde{\mathbf{Z}}'_{d, -t} \end{bmatrix} \right).$$

Note that $\mathbf{y}(\tau)$ looks slightly differently from the one in equation 15, since

$$\begin{aligned}\tau \tilde{\mathbf{Z}}_{:, -t} + (1 - \tau) \tilde{\mathbf{Z}}_{:, -t} &= \tilde{\mathbf{Z}}_{:, -t} \\ \tau \tilde{\mathbf{Z}}'_{:, -t} + (1 - \tau) \tilde{\mathbf{Z}}'_{:, -t} &= \tilde{\mathbf{Z}}'_{:, -t}.\end{aligned}$$

Rewriting the equation 15 using the newly defined $\mathbf{y}(0)$, $\mathbf{y}(1)$ and $\mathbf{y}(\tau)$, we obtain

$$\mathbf{v}^* := \arg \min_{\mathbf{v}} \int_0^1 \mathbb{E}[\|\mathbf{z}'_t - \mathbf{z}_t - \mathbf{v}(\mathbf{z}_t(\tau), \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}, \tau)\|^2] d\tau \quad (16)$$

where the vector field

$$\mathbf{v} : \mathbb{R}^d \otimes \tilde{\mathbb{R}}^{T \times d} \otimes \tilde{\mathbb{R}}^{T \times d} \otimes \mathbb{R} \rightarrow \mathbb{R}^d.$$

Finally, our refinement function is

$$\mathbf{l}(\mathbf{z}_t; \tilde{\mathbf{Z}}_{:, -t}) = \mathbf{y}(0) + \int_0^1 \mathbf{v}^*(\mathbf{y}(\tau), \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}, \tau) d\tau.$$

with $\mathbf{y}(0)$ set to $\mathbf{z}_t$. Similar to the previous section, we approximate it using the Euler method. Now, we prove that the above refinement function is optimal.

Theorem 3.3. $\mathbf{l}(\cdot, \tilde{\mathbf{Z}}_{:, -t})$ transports $\mathbf{z}_t$ to the target distribution, i.e.,

$$p(\mathbf{l}(\mathbf{z}_t, \tilde{\mathbf{Z}}_{:, -t})|\tilde{\mathbf{Z}}_{:, -t}) = \prod_i p(z_{i,t}|\tilde{\mathbf{Z}}_{i, -t}).$$

The proof of this theorem can be found at Section A.3.

Theorem 3.3 states that $\mathbf{l}(\cdot, \tilde{\mathbf{Z}}_{:, -t})$ transports samples from the initial distribution $p(\mathbf{z}_t|\tilde{\mathbf{Z}}_{:, -t})$ to the target $\prod_i p(z_{i,t}|\tilde{\mathbf{Z}}_{i, -t})$, and as a result, reducing the KL divergence in equation 9 to zero.

We provide the concrete SICA algorithm with WGF and Rectified Flow as de-mixing flow choices in Algorithm 2.

4 EXPERIMENTS

We now evaluate the empirical performance of SICA on both artificial and real-world datasets. For both flow variants (WGF and RF), the vector field $\mathbf{v}$ is parameterized using a three-layer one-dimensional convolutional neural network (CNN) (`torch.nn.Conv1d`) with 16 hidden channels, to process sequence inputs. We consider several linear and non-linear ICA variants: FastICA (Hyvarinen, 1999), LICA (Suzuki and Sugiyama, 2011), Permutation-Contrastive Learning (PCL) (Hyvarinen and Morioka, 2017) and iVAE (Khemakhem et al., 2020). See the supplementary material for more implementation details and source code to reproduce experimental results.

4.1 Autoregressive Signals

In this experiment, the true signals are generated independently using an autoregressive model, and are created using the following formula:

$$\begin{aligned}s_{i,t} &= \sum_{k=1}^7 \beta_k s_{i,t-k} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.1^2), \\ \beta &= [4, 3, 2, 1, -0.5, 0.3, -0.2] \times 0.1.\end{aligned}$$

$s_{i,1} \dots s_{i,7}$ are sampled from the standard normal distribution, and the mixing function $\mathbf{f}$ is constructed as an iterative update on the true signal. At iteration j , we mix the signal with an update

$$\begin{aligned}\mathbf{x}^{(j)} &= \mathbf{x}^{(j-1)} + \mathbf{h}(\mathbf{W}\mathbf{x}^{(j-1)}), \\ \mathbf{W} &\in \mathbb{R}^{d \times d}, \mathbf{h} : \mathbb{R}^d \rightarrow \mathbb{R}^d.\end{aligned}$$

We initialize $\mathbf{x}^{(0)}$ as $\mathbf{s}$. When $\mathbf{h}$ is the identity function, the mixed signal $\mathbf{x}$ is a linear function of $\mathbf{s}$, given by $\mathbf{x} = (\mathbf{I} + \mathbf{W})^J \mathbf{s}$, where J denotes the total number of mixing updates. In this section, we consider the setting where the number of signals is $d = 2$ and the sequence length is $T = 1024$. $\mathbf{W}$ is specified with diagonal entries equal to 1 and off-diagonal entries equal to 0.7. We examine two scenarios: (i) $\mathbf{h}$ as the identity function, which will be referred to as the *linear* setting, and (ii) $\mathbf{h}$ as the GELU activation, which will be referred to as the *nonlinear* setting.

First, we perform 20 non-linear mixing updates and use SICA with RF to recover the original signal. We then illustrate the recovered signal in Figure 3(a), where we can see that the observed signals (blue) are almost visually indistinguishable due to the mixing function. However, the signal recovered by SICA (red) accurately traces the true signal (black).

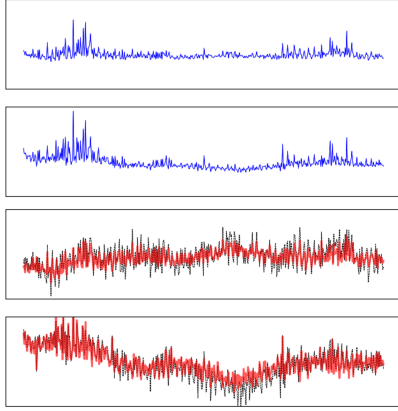
We further measure the performance using mean correlation coefficients (MCC) between the recovered signals and the ground truth. We plot MCC over various

Algorithm 2 Iterative KL minimization Algorithm with WGF or RF

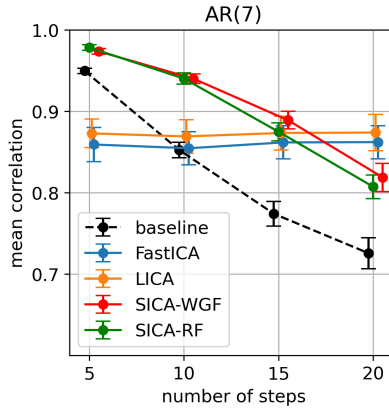
```

1: Input: A mixed multi-dimensional sequence  $\mathbf{X}$ , Choice of Flow: {WGF, RF}.
2: Let  $\mathbf{Z}^{(0)} = \mathbf{X}$ 
3: for  $j = 1 \dots J$  do
4:   Construct dataset  $\mathcal{D} := \{(\mathbf{z}_t, \tilde{\mathbf{Z}}_{:, -t}^{(j-1)})\}_{t=1}^T$ .
5:   for  $i = 1 \dots d$  do //for each signal
6:      $\mathcal{D}'_i = \{(z_{i,t'}, \tilde{\mathbf{Z}}_{i, -t'}^{(j-1)})\}_{t' \in T'}$ ,  $T'$  is a random permutation of  $\{1 \dots T\}$ .
7:   end for
8:   if Choice of Flow is WGF then
9:     Estimate  $\mathbf{v}^*$  in equation 14 using  $\mathcal{D}$  and  $\mathcal{D}'_1, \dots, \mathcal{D}'_d$ .
10:    For all  $t \in [T]$ ,  $\mathbf{z}_t^{(j)} = \mathbf{z}_t^{(j-1)} + \eta \mathbf{v}^*(\mathbf{z}_t^{(j-1)}; \tilde{\mathbf{Z}}_{:, -t}^{(j-1)})$  // one step of Euler update
11:  else
12:    Estimate  $\mathbf{v}^*$  in equation 16 using  $\mathcal{D}$  and  $\mathcal{D}'_1, \dots, \mathcal{D}'_d$ .
13:    For all  $t \in [T]$ ,  $\mathbf{z}_t^{(j)} = \text{ODESolve}(\text{init} = \mathbf{z}_t^{(j-1)}, \text{vector field} = \mathbf{v}^*(\cdot; \tilde{\mathbf{Z}}_{:, -t}^{(j-1)}, \tilde{\mathbf{Z}}_{:, -t}^{(j-1)}), \text{time} = 1)$ .
14:  end if
15: end for
16: return  $\mathbf{Z}^{(J)}$ 

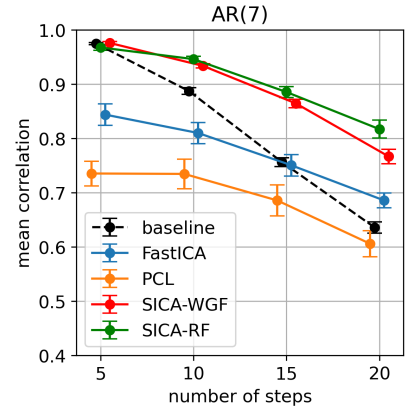
```



(a) Mixed Signal (blue), Recovered Signal (red) and True Signal (black)



(b) Linear Mixing $\mathbf{f}$ on AR (7)



(c) Non-linear Mixing $\mathbf{f}$ on AR (7)

Figure 3: AR (7) dataset. Left: Illustrative Example. Middle and Right, MCC over various mixing steps. The higher the better. MCC is measured over 20 independent runs, and error bars represent the standard error.

numbers of mixing steps (J). The larger J is, the more mixed the signals are.

Figure 3(b) shows that in the linear setting, as the signals become more entangled, the MCC of SICA methods reduces. In contrast, the linear ICAs (FastICA, LICA) remain steady. This is not surprising as linear ICAs are designed to handle linear mixing functions. However, in three out of four cases, $J = 5, 10, 15$, SICA methods still outperforms FastICA, LICA and the baseline (MCC of the observed signal $\mathbf{x}$, marked as “baseline” on the plot), which shows that although our non-linear flows are not as robust as linear models, their performance is still strong in non-extreme cases.

We now demonstrate the advantage of our method using the non-linear setting. In this setting, we also

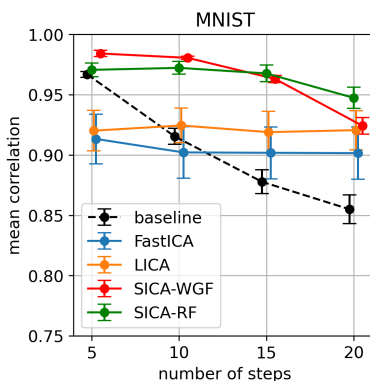
compare SICA with non-linear ICA variants (PCL and iVAE). Figure 3(c) shows that MCCs of all methods, including FastICA, reduce as the number of mixing steps increases. However, in this setting, SICA methods maintain a significant lead in MCC compared to FastICA, PCL, and the baseline. Note that iVAE cannot achieve an MCC above 0.4 for all mixing steps, so it is not plotted for better visualization of the other methods.

4.2 MNIST Images

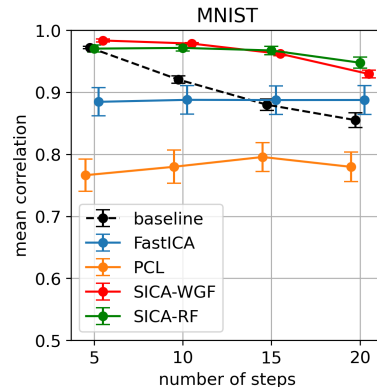
In this set of experiments, we evaluate the performance of ICA methods on the task of de-mixing overlaid images. We randomly select samples from the MNIST dataset, and flatten them as signals $\mathbf{s} \in \mathbb{R}^{d \times (32 \times 32)}$.



(a) Mixed Signal (1st row), True Signal (2nd row), Signal Recovered by SICA-RF (3rd row) and by FastICA (4th row)



(b) Linear Mixing f on MNIST



(c) Non-linear Mixing f on MNIST

Figure 4: MNIST dataset. Left: Illustrative Example. Middle and Right: MCC over various mixing steps. MCC are measured over 20 independent runs and error bars represent the standard error.

Then apply the linear and non-linear mixing steps described in the previous section to obtain mixed signals $\mathbf{x}$.

First, we randomly select three images ($d = 3$), apply 20 steps of non-linear mixing. It can be seen that all observed channels are visually identical after mixing. We then use SICA to disentangle these images. It can be seen that SICA correctly disentangles three images as 6, 9, and 5. The separations given by FastICA are not as clear.

One interesting observation is that the first recovered channel by FastICA has a white digit 9, but the third recovered channel has a darker digit 5. This reveals a limitation of linear ICA in image separation: most algorithms recover the sources only up to an arbitrary dimension-wise transformation. Such transformations are often difficult to correct. For example, without prior knowledge of whether the ground truth consists of white digits on a dark background or dark digits on a white background, it becomes impossible to resolve this ambiguity, resulting in suboptimal recovery. However, our proposed flow-based methods do not appear to suffer from this issue, and studying the identifiability of our flow-based models could be an interesting future direction.

Finally, we evaluate the performance of SICA using MCC over multiple runs in Figures 4(b) and 4(c). In the linear setting, the results show that, across all mixing steps, SICA methods not only stay above the baseline but also achieve superior performance compared to all linear ICA methods. In the non-linear setting (Figure 4(c)), we compare the SICA with two non-linear

ICA variants (PCL and iVAE). Note that iVAE again cannot achieve MCC above 0.55 for all mixing steps; thus, it is not plotted for better visualization of other methods. It can be seen that SICA again maintains the lead among all methods across all mixing steps.

5 CONCLUSIONS

In this paper, we propose a novel ICA criterion, SICA, based on the sufficiency assumption: the identified signals should be self-sufficient, and other recovered signals do not help the reconstruction of missing data of that signal. We show how to leverage the factorization implied by this assumption, and design a KL divergence objective to disentangle signals. We propose using de-mixing flows to minimize the KL divergence iteratively and provide justification for their optimality. Experiments conducted on both synthetic and real-world datasets yield promising results.

References

- Amari, S.-i., Cichocki, A., and Yang, H. (1995). A New Learning Algorithm for Blind Signal Separation. In *Advances in Neural Information Processing Systems*, volume 8.
- Ambrosio, L., Gigli, N., and Savaré, G. (2008). *Gradient Flows, In Metric Spaces and in the Space of Probability Measures*. Birkhäuser Basel, Basel.
- Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. (2018). Mutual Information Neural Estimation. In *Proceedings of the 35th International Conference on Ma-*

- chine Learning, volume 80 of *Proceedings of Machine Learning Research*, pages 531–540. PMLR.
- Bell, A. J. and Sejnowski, T. J. (1995). An Information-Maximization Approach to Blind Separation and Blind Deconvolution. *Neural Computation*, 7(6):1129–1159.
- Brakel, P. and Bengio, Y. (2017). Learning Independent Features with Adversarial Nets for Non-linear ICA. arXiv:1710.05050 [stat].
- Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.
- Chen, X., Duan, Y., Houthoofd, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.
- Chewi, S. (2025). Log Concave Sampling.
- Dinh, L., Krueger, D., and Bengio, Y. (2015). NICE: Non-linear Independent Components Estimation. arXiv:1410.8516 [cs].
- Halva, H., Le Corff, S., Lehericy, L., So, J., Zhu, Y., Gassiat, E., and Hyvarinen, A. (2021). Disentangling Identifiable Features from Noisy Data with Structured Nonlinear ICA. In *Advances in Neural Information Processing Systems*, volume 34, pages 1624–1633. Curran Associates, Inc.
- He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked Autoencoders Are Scalable Vision Learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009.
- Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634.
- Hyvarinen, A., Khemakhem, I., and Morioka, H. (2023). Nonlinear independent component analysis for principled disentanglement in unsupervised deep learning. *Patterns*, 4(10). Publisher: Elsevier.
- Hyvarinen, A. and Morioka, H. (2016). Unsupervised Feature Extraction by Time-Contrastive Learning and Nonlinear ICA. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.
- Hyvarinen, A. and Morioka, H. (2017). Nonlinear ICA of Temporally Dependent Stationary Sources. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 460–469. PMLR.
- Hyvarinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430.
- Hyvarinen, A., Sasaki, H., and Turner, R. (2019). Non-linear ica using auxiliary variables and generalized contrastive learning. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 859–868. PMLR.
- Khemakhem, I., Kingma, D., Monti, R., and Hyvarinen, A. (2020). Variational autoencoders and non-linear ica: A unifying framework. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2207–2217. PMLR.
- Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. (2023). Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*.
- Liu, X., Gong, C., and qiang liu (2023). Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*.
- Shi, Y., Siddharth, N., Torr, P. H., and Kosiorek, A. R. (2022). Adversarial masking for self-supervised learning. In *International Conference on Machine Learning*.
- Sugiyama, M., Suzuki, T., and Kanamori, T. (2012). *Density Ratio Estimation in Machine Learning*. Cambridge University Press.
- Suzuki, T. and Sugiyama, M. (2011). Least-Squares Independent Component Analysis. *Neural Computation*, 23(1):284–301.
- Tashiro, Y., Song, J., Song, Y., and Ermon, S. (2021). CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. In *Advances in Neural Information Processing Systems*, volume 34, pages 24804–24816. Curran Associates, Inc.
- Wang, X., Chen, H., Tang, S., Wu, Z., and Zhu, W. (2024). Disentangled representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9677–9696.
- Yi, M., Zhu, Z., and Liu, S. (2023). MonoFlow: Rethinking divergence GANs via the perspective of Wasserstein gradient flows. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 39984–40000. PMLR.

Self-sufficient ICA via KL Minimising Flows

Supplementary Materials

A MISSING PROOFS

A.1 Proof of Equation 5

Proof. By the chain rule for conditional densities with a fixed ordering $1, \dots, d$, we have

$$p(\mathbf{s} \mid \mathbf{U}) = \prod_{i=1}^d p(s_i \mid \mathbf{s}_{1:i-1}, \mathbf{U}). \quad (17)$$

By the conditional independence assumption in Equation 4,

$$s_i \perp\!\!\!\perp (\mathbf{s}_{-i}, \mathbf{u}_{-i}) \mid \mathbf{u}_i, \quad (18)$$

we have

$$p(s_i \mid \mathbf{s}_{1:i-1}, \mathbf{U}) = p(s_i \mid \mathbf{u}_i). \quad (19)$$

Substituting this back into the chain rule gives

$$p(\mathbf{s} \mid \mathbf{U}) = \prod_{i=1}^d p(s_i \mid \mathbf{u}_i), \quad (20)$$

as required.

The other direction is also true. Fix i . Using Bayes' rule and equation 5,

$$\begin{aligned} p(s_i \mid \mathbf{s}_{-i}, \mathbf{U}) &= \frac{p(s_i, \mathbf{s}_{-i} \mid \mathbf{U})}{p(\mathbf{s}_{-i} \mid \mathbf{U})} = \frac{p(\mathbf{s} \mid \mathbf{U})}{\int p(\mathbf{s} \mid \mathbf{U}) ds_i} \\ &= \frac{[p(s_i \mid \mathbf{u}_i) \prod_{j \neq i} p(s_j \mid \mathbf{u}_j)]}{\prod_{j \neq i} p(s_j \mid \mathbf{u}_j) \int p(s_i \mid \mathbf{u}_i) ds_i} \\ &= \frac{p(s_i \mid \mathbf{u}_i)}{1} = p(s_i \mid \mathbf{u}_i), \end{aligned}$$

since $\int p(s_i \mid \mathbf{u}_i) ds_i = 1$ (replace the integral by a sum in the discrete case). Hence $s_i \perp\!\!\!\perp (\mathbf{s}_{-i}, \mathbf{u}_{-i}) \mid \mathbf{u}_i$. □

A.2 Proof of Theorem 3.1

Proof. Let us rewrite

$$D_{\text{KL}}^{(j)}(\mathbf{l}^{(j)}) - D_{\text{KL}}^{(j-1)}(\mathbf{l}^{(j-1)}) \quad (21)$$

$$= \underbrace{D_{\text{KL}}^{(j)}(\mathbf{l}^{(j)}) - D_{\text{KL}}^{(j-1)}(\mathbf{l}^{(j)})}_A + \underbrace{D_{\text{KL}}^{(j-1)}(\mathbf{l}^{(j)}) - D_{\text{KL}}^{(j-1)}(\mathbf{l}^{(j-1)})}_B. \quad (22)$$

The optimization in equation 9 implies that $B \leq 0$ as a result of optimization. We only need to prove that the A is smaller than zero.

Recall,

$$\begin{aligned} D_{\text{KL}}^{(j)}(\mathbf{l}^{(j)}) &:= D_{\text{KL}} \left[p(\mathbf{z}_t^{(j)} | \tilde{\mathbf{Z}}_{:, -t}^{(j)}) \parallel \prod_{i=1}^d p(z_{i,t}^{(j)} | \tilde{\mathbf{Z}}_{i, -t}^{(j)}) \right] \\ D_{\text{KL}}^{(j-1)}(\mathbf{l}^{(j)}) &:= D_{\text{KL}} \left[p(\mathbf{z}_t^{(j)} | \tilde{\mathbf{Z}}_{:, -t}^{(j-1)}) \parallel \prod_{i=1}^d p(z_{i,t}^{(j-1)} | \tilde{\mathbf{Z}}_{i, -t}^{(j-1)}) \right]. \end{aligned} \quad (23)$$

First, we notice that

$$\begin{aligned} p(\mathbf{z}_t^{(j)} | \tilde{\mathbf{Z}}_{:, -t}^{(j-1)}) &= p(\mathbf{z}_t^{(j)} | \tilde{\mathbf{Z}}_{:, -t}^{(j)}), \\ p(z_{i,t}^{(j-1)} | \tilde{\mathbf{Z}}_{i, -t}^{(j-1)}) &= p(z_{i,t}^{(j-1)} | \tilde{\mathbf{Z}}_{i, -t}^{(j)}), \end{aligned} \quad (24)$$

since $\tilde{\mathbf{Z}}_{:, -t}^{(j)}$ is $\tilde{\mathbf{Z}}_{:, -t}^{(j-1)}$ transformed via an invertible function $\mathbf{l}$. Apply the equalities in equation 24 to equation 23 and expand both KL divergences. Some algebra shows,

$$A = \sum_i \left\{ \mathbb{E}_{p_i^{(j)}} \left[\log p(z_{i,t}^{(j-1)} | \tilde{\mathbf{Z}}_{i, -t}^{(j)}) \right] - \mathbb{E}_{p_i^{(j)}} \left[\log p(z_{i,t}^{(j)} | \tilde{\mathbf{Z}}_{i, -t}^{(j)}) \right] \right\} \leq 0,$$

due to Gibbs inequality. $p_i^{(j)}$ is short for $p_i(z_{i,t}^{(j)} | \tilde{\mathbf{Z}}_{i, -t}^{(j)})$. Since both $A \leq 0$ and $B \leq 0$, we obtain the desired result from equation 22. $\square$

A.3 Proof of Theorem 3.3

Proof. First, let us define a few symbols. Let $\mathbf{w} = \mathbf{l}(\mathbf{z}_t, \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) := \mathbf{y}(0) + \int_0^1 \mathbf{v}^*(\mathbf{y}(\tau), \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}, \tau) d\tau$, with $\mathbf{y}(0)$ set to $\mathbf{z}_t$. and define the alias $\mathbf{l}(\mathbf{Z}, t) := \mathbf{l}(\mathbf{z}_t, \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t})$. Due to Lemma A.1,

$$p(\mathbf{w} | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) = p(\mathbf{z}_t | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}). \quad (25)$$

Since $\mathbf{z}'_t$ is independent of $\tilde{\mathbf{Z}}_{:, -t}$ by construction,

$$p(\mathbf{z}'_t | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) = p(\mathbf{z}'_t | \tilde{\mathbf{Z}}'_{:, -t}) = \prod_i p(z'_{i,t} | \tilde{\mathbf{Z}}'_{i, -t}), \quad (26)$$

where the last equality is by the construction of $(\mathbf{z}'_t, \tilde{\mathbf{Z}}'_{:, -t})$ in the Section 3.1. Combining equation 25 and the factorization rule in equation 26, we get the factorization

$$p(\mathbf{w} | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) = \prod_i p(z'_{i,t} | \tilde{\mathbf{Z}}'_{i, -t}). \quad (27)$$

Substitute $\tilde{\mathbf{Z}}_{i, -t}$ for $\tilde{\mathbf{Z}}'_{i, -t}$ and the independence of $\mathbf{l}(\mathbf{Z})$ and $\tilde{\mathbf{Z}}'_{:, -t}$, we can rewrite equation 27 as

$$p(\mathbf{l}(\mathbf{Z}) | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{i, -t} = \tilde{\mathbf{Z}}_{i, -t}) = p(\mathbf{l}(\mathbf{Z}) | \tilde{\mathbf{Z}}_{:, -t}) = \prod_i p(z'_{i,t} | \tilde{\mathbf{Z}}'_{i, -t} = \tilde{\mathbf{Z}}_{i, -t}).$$

Since $(z'_{i,t}, \tilde{\mathbf{Z}}'_{i, -t})$ is identically distributed as $(z_{i,t}, \tilde{\mathbf{Z}}_{i, -t})$ by construction in the Section 3.1, the RHS $\prod_i p(z'_{i,t} | \tilde{\mathbf{Z}}'_{i, -t} = \tilde{\mathbf{Z}}_{i, -t}) = \prod_i p(z_{i,t} | \tilde{\mathbf{Z}}_{i, -t})$ and we obtain the desired result. $\square$

Lemma A.1.

$$p(\mathbf{l}(\mathbf{z}_t, \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) = p(\mathbf{z}'_t | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}).$$

Proof. This proof is a conditional version of the Marginal Preserving Theorem (Theorem 3.3) in (Liu et al., 2023). Let $\rho_\tau(\mathbf{z} | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t})$ be the conditional density function of the interpolation $\mathbf{z}_t(\tau)$. First, we can see that for any $\mathbf{h}(\cdot)$ that vanishes at the boundary of the domain,

$$\partial_\tau \mathbb{E}[h(\mathbf{z}_t(\tau)) | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}] = \int h(\mathbf{z}) \partial_\tau p_\tau(\mathbf{z} | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) d\mathbf{z}. \quad (28)$$

Second, since the interpolation $\mathbf{z}(\tau)$ is a deterministic function of $\mathbf{z}(0)$ and $\mathbf{z}(1)$, we can see that

$$\partial_\tau \mathbb{E}[h(\mathbf{z}_t(\tau)) | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}] = \mathbb{E}_{\mathbf{z}(0), \mathbf{z}(1)}[\partial_\tau h(\mathbf{z}_t(\tau)) | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}] \quad \text{reparameterization trick} \quad (29)$$

$$= \mathbb{E}[\nabla^\top h(\mathbf{z}_t(\tau)) \partial_\tau \mathbf{z}_t(\tau) | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}] \quad \text{law of unconscious statistician} \quad (30)$$

$$= \mathbb{E}[\nabla^\top h(\mathbf{z}_t(\tau)) \mathbb{E}[\partial_\tau \mathbf{z}_t(\tau) | \mathbf{z}_t(\tau), \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}] | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}] \quad (31)$$

$$= \mathbb{E}[\nabla^\top h(\mathbf{z}_t(\tau)) \mathbb{E}[\mathbf{z}_t(1) - \mathbf{z}_t(0) | \mathbf{z}_t(\tau), \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}] | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}] \quad (32)$$

$$= \mathbb{E}[\nabla^\top h(\mathbf{z}_t(\tau)) \mathbf{v}^* \left(\mathbf{z}_t(\tau), \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}, \tau \right) | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}] \quad \text{optimal solution} \quad (33)$$

$$= \int \nabla^\top h(\mathbf{z}) \mathbf{v}^* \left(\mathbf{z}, \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}, \tau \right) \rho_\tau(\mathbf{z} | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) d\mathbf{z} \quad (34)$$

$$= - \int h(\mathbf{z}) \nabla \cdot \left[\mathbf{v}^* \left(\mathbf{z}, \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}, \tau \right) \rho_\tau(\mathbf{z} | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) \right] d\mathbf{z}, \quad \text{integration by parts} \quad (35)$$

where equation 33 is due to the fact that the conditional expectation $\mathbb{E}[\mathbf{z}_t(1) - \mathbf{z}_t(0) | \mathbf{z}_t(\tau), \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}]$ is the optimal solution of the training objective equation 16. From both equation 28 and equation 35, we can see that the continuity equation holds for $\mathbf{v}^*$

$$\partial_\tau \rho_\tau(\mathbf{z} | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) = -\nabla \cdot \left[\mathbf{v}^* \left(\mathbf{z}, \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}, \tau \right) \rho_\tau(\mathbf{z} | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t}) \right], \text{ a.s.} \quad (36)$$

Under common regularity conditions (continuity of ρ_τ , $\mathbf{v}^*$, and the law of $\mathbf{z}_t$ are fully supported), the equality holds everywhere. It means an ODE defined by the vector field, with $\mathbf{z}_t(0)$ as the initial value, will have the same marginal distribution ρ_τ as the interpolation $\mathbf{z}_t(\tau)$ for all $\tau \in [0, 1]$. The fact that $\rho_1 = p(\mathbf{z}'_t | \tilde{\mathbf{Z}}_{:, -t}, \tilde{\mathbf{Z}}'_{:, -t})$ concludes the proof. $\square$

B EXPERIMENTS SETTINGS

B.1 Neural Network Architecture

For both SICA-RF and SICA-WGF, we use a three-layer Convolutional Neural Network (CNN) to model the vector field $\mathbf{v}$. Since CNN cannot directly handle NaN values, we use masks to indicate the missing column in $\tilde{\mathbf{Z}}_t$. For a d -dimensional sequence $\mathbf{Z}$ with length T , the inputs to our neural network are three $d \times T$ matrices: $[\mathbf{M}_t, \mathbf{Z}_t, \tilde{\mathbf{Z}}_{:, -t}]$,

- $\mathbf{M}_t$ is a mask matrix whose elements are all zeros except the t -th column are set to ones.
- $\mathbf{Z}_t$ is a matrix whose columns are $\mathbf{z}_t$ repeated T times.
- $\tilde{\mathbf{Z}}_{:, -t}$ is $\mathbf{Z}$ but its t -th column is replaced with an arbitrary constant (e.g., zero).

These inputs are fed to the CNN block as a $3d$ -channel sequence of length T .

The CNN block of our neural network contains three layers, each with 16 hidden channels, followed by a linear projection layer. It projects the output from the CNN block to a $\mathbb{R}^d$ vector, as the output of $\mathbf{v}$.

Using `torch.nn` PyTorch package, the neural network is defined as

```
self.net = nn.Sequential(
    nn.Conv1d(3*d, 16, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.Conv1d(16, 16, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.Conv1d(16, 16, kernel_size=3, padding=1),
    nn.Flatten(startdim = 1)
)

self.fc = nn.Linear(hidden_channels * T, d)
```

Note that for SICA-RF, we need to add another three matrices representing $\tilde{\mathbf{Z}}'_{:, -t}$ and an additional time-encoding to the network to reflect that $\mathbf{v}$ is a time-varying function.

This neural network can be easily modified to handle 2-dimensional indices of t , by reshaping the inputs as $d \times \sqrt{T} \times \sqrt{T}$ tensors (assuming the sequences are flattened $\sqrt{T} \times \sqrt{T}$ images) and change `nn.Conv1d` to `nn.Conv2d`.

B.2 Hyperparameters of the Proposed Method

In addition to the neural network model, which we detailed in Section B.1, the proposed method has only a few other parameters.

For the SICA-WGF method, we train the density ratio estimator using the Adam optimizer with a learning rate of 0.00001, with a batch size of 100, and run for 10 epochs. This setting is kept for all experiments in our paper.

For the SICA-RF method, we train the rectified flow using Adagrad optimizer with a learning rate of 0.00001, with a batch size of 100, and run for 100 epochs. In the sampling stage, we set the number of Euler steps to be 100. This setting is kept for all experiments in our paper.

The Algorithm 2 requires a maximum number of iterations (J). For SICA-WGF, we set it to 10, and for SICA-RF, we set it to 30 for AR (7) and 20 for MNIST data.

B.3 Implementations of Baseline Methods

- FastICA: We use the implementation provided by `sklearn.decomposition.FastICA`.
 - We use the default hyperparameters provided with the implementation and set the maximum iteration to 20000.
- Least-squares ICA: We use the implementation provided by the original authors <https://ibis.t.u-tokyo.ac.jp/suzuki/software/LICA/>.
 - We provide a list of kernel bandwidth $\sigma = [0.1, 0.5]$ and regularization parameters $\lambda = [0.01, 0.1]$ for the method to select using the built-in cross-validation procedure. We set the number of basis functions for the RBF kernel to 50 to speed up the computation.
- iVAE: We use the implementation provided by the original authors <https://github.com/ilkhem/icebeem/tree/master/models/ivae>.
 - We use the default hyperparameters provided with the implementation, except the number of epochs was changed from 7e4 to 2e4, which does not seem to impact the performance but saves a significant amount of time.
- Permutation Contrastive Learning (PCL): We implemented it ourselves, and were inspired by https://github.com/RyanBalshaw/Nonlinear_ICA_implementations.
 - We tried various settings of neural network structures and settled on an MLP with two hidden layers and 256 neurons in each layer for the most reliable performance.

The code for these methods can be found under the `competitors` folder in the attached code repository.

B.4 Reproducing Results

To reproduce Illustrative results, please run: `demo_heart.py`, `demo_AR7.py`, `demo_MNIST.py`, `demo_rep_AR7.py` and `demo_rep_MNIST.py`.