MILE: MUTUAL INFORMATION LOGDET ESTIMATOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Mutual information (MI) estimation plays an important role in representational learning. However, accurately estimating mutual information is challenging, especially for high-dimensional variables with limited batch data. In this work, we approach the mutual information estimation problem via the logdet function of data covariance. To extend the logdet function for entropy estimation of non-Gaussian variables, we assume that the data can be approximated well by a Gaussian mixture distribution and introduce a lower and upper bound for the entropy of such distributions. To deal with high dimensionality, we introduce "ridge" term in the logdet function to stabilize the estimation. Consequently, the mutual information can be estimated by the entropy decomposition. Our method MILE significant outperforms conventional neural network-based MI estimators in obtaining low bias and low variance MI estimation. Besides, it well pass the challenging self-consistency tests. Simulation studies also show that, beyond a better MI estimator, MILE can simultaneously gain competitive performance with SOTA MI based loss in self-supervised learning.

1 INTRODUCTION

Mutual information (MI) estimation and optimization are crucial in various machine learning tasks, particularly for representation learning (Tishby & Zaslavsky, 2015; Chen et al., 2016; Belghazi et al., 2018; Hjelm et al., 2019; Oord et al., 2018; Song & Ermon, 2019). Unlike other similarity measures, MI has the advantage of capturing both linear and non-linear dependencies between different entities such as features. However, estimating MI accurately from limited data samples with an unknown distribution for many modern machine-learning tasks is especially challenging (McAllester & Stratos, 2020), often dwarfing traditional parametric and non-parametric approaches in statistics and machine-learning communities (Nemenman et al., 2004; Gao et al., 2015).

Recently, there has been a growing interest in estimating MI with variational approaches (Barber & Agakov, 2003; Nguyen et al., 2010), which can be naturally combined with deep learning methods (Alemi et al., 2016; Oord et al., 2018; Poole et al., 2019). The seminal one, Mutual Information Neural Estimation (MINE) method (Belghazi et al., 2018), iteratively estimates the variational function using a neural network and demonstrates successful applications in various learning tasks. However, MINE and its variants, such as Smoothed Mutual Information Lower-bound Estimator (Song & Ermon, 2019) and Neural Entropic Estimation (NEE) (Chan et al., 2019) are known to be difficult to balance the bias and variance, and the convergence rate of this type of estimators is relatively slow. Alternatively, Information Noise Contrastive Estimation (InfoNCE (Gutmann & Hyvärinen, 2012; Oord et al., 2018) is a lower bound of MI and has been widely used as a contrastive loss function for self-supervised learning (SSL). It is widely known that InfoNCE outperforms MINE in representation learning, which leads to a puzzle that a tighter bound of MI may not lead to a better performance in SSL (Tian et al., 2020; Wang & Isola, 2020). However, InfoNCE is not an effective estimator for MI, which may result in a large bias, particularly when MI is large (Song & Ermon, 2019).

In this work, we address an alternative estimator to resolve these shortcomings. We propose to estimate the mutual information base on the logdet function of data covariance. In order to extend the logdet function for entropy estimation of Gaussian distributed variables to fit for entropy estimation of non-Gaussian variables, we assume the data is subjected to Gaussian mixture distribution and introduce a lower and upper bound for the entropy of such variable. To deal with high dimensional variables, we introduce an extra identify matrix in the logdet function to gain stable

estimation. The mutual information can be consequently estimated by the entropy decomposition. Unlike conventional neural network based MI estimators, our MILE does not require additional training. Simulation studies demonstrate that our proposed MILE gains significant improvements in both the estimation speed and accuracy. Moreover, we find that the MILE can perform well in the challenging self-consistency test. We also show that being a tighter estimation of MI, MILE can gain competitive performance with SOTA MI based loss in self-supervised learning.

2 MUTUAL INFORMATION LOGDET ESTIMATOR

2.1 BACKGROUND

Information theory tells us the differential entropy of a variable X is defined by its probability density function $p(X)$ as follows,

$$\mathbf{H}(X) = \int -p(X) \log p(X) dX \tag{1}$$

When X subjects to a Gaussian, the integral in Eqn (1) has a closed-form expression. To be specific, for a multi-variable Gaussian variable $X \sim N(\mu, \Sigma_X)$, its differential entropy can be represented by the Logarithm Determinant (LogDet for short) of its covariance matrix as:

$$\mathbf{H}(X) = \frac{1}{2} \log\det \Sigma_X + \frac{d}{2}(\log 2\pi + 1) \tag{2}$$

$$= \frac{1}{2} \sum_{i}^{d} \log \lambda_i + \frac{d}{2}(\log 2\pi + 1) \tag{3}$$

where d is the dimension of variables and λ_i denote the ith eigenvalue of its covariance matrix. Obviously, the second term $\frac{d}{2}(\log 2\pi + 1)$ in Eqn(2) is constant, which will be denoted as C_d later on. The entropy measurement is determined by $\log\det \Sigma_x$. We refer to this entropy estimator as the LogDet entropy estimator.

Despite its advantages in computation, we need to pay attention to some special cases when using the LogDet entropy estimator. For example, when having finite but strongly correlated variables, the covariance matrix may become singular, leading to zero determinant. For instance, this case occurs in dealing with image data, where the original image data is always sparse and strongly correlated. Prior works (Zhouyin & Liu, 2021) on such entropy estimators provide a solution: by adding independent Gaussian noise $z \sim N(0, \beta * I)$ to X as $\hat{X} = X + z$, we have a 'noise' $\hat{X}$ to any given X. In such cases, we can apply logdet operation to covariance matrix $\mathrm{Cov}(\hat{T}) = \Sigma + \beta I$ to estimate its entropy.

It is worth noting that previous work (Zhouyin & Liu, 2021) has attempted to reduce the influence of added noise by enlarging the original covariance matrix through an expanding factor, denoted as β. However, it should be recognized that enlarging the original covariance matrix can introduce a substantial bias in the estimation of mutual information. So here we scale down noise's variance to eliminate artificial bias brought by added noise. We define the entropy estimator more formally as follows: given n samples $X = [x_1, x_2, \ldots, x_n]$ where each sample have d dimensional features that $X \in \mathbb{R}^{n \times d}$, its covariance matrix is calculated by $\Sigma_X = \frac{X^T X}{n}$ (we assume X is zero-mean in this work), the LogDet entropy estimator can be refined as:

$$\mathbf{H}_D(X) = \frac{1}{2} \log\det(\Sigma_X + \beta I) + C_d = \frac{1}{2} \log\det\left(\frac{X^T X}{n} + \beta I\right) + C_d \tag{4}$$

where β is the scaling parameter. By performing the singular value decomposition to Σ_X, Eqn 4 is equivalent to:

$$\mathbf{H}_D(X) = \sum_{i=1}^{k} \frac{1}{2} \log(\lambda_i + \beta) + C_d \tag{5}$$

where $k = \mathrm{rank}(X)$ and λ_i denotes the i th eigenvalue of Σ_X. We see the first term of Eqn 5 approximate the dominant term in Eqn 2, which is the differential entropy of multivariate Gaussian

random variables. Therefore, LogDet can estimate the entropy of multivariate Gaussian variables by approximating the differential entropy.

To obtain the mutual information, we also need to define the joint LogDet entropy estimator for two variables. Given samples sets $X_1 \in \mathbb{R}^{n\times d_1}, X_2 \in \mathbb{R}^{n\times d_2}$, let $Z = [X_1, X_2] \in \mathbb{R}^{n\times(d_1+d_2)}$ denotes the concatenation of X_1 and X_2. The covariance matrix of Z is

$$\Sigma_Z = \begin{pmatrix} \Sigma_1 & F \\ F^T & \Sigma_2 \end{pmatrix},$$

where Σ_1 and Σ_2 is covariance matrices of X_1 and X_2 respectively, and F equals to $\frac{X_1^T X_2}{n}$, the joint LogDet entropy of X_1 and X_2 is:

$$\begin{aligned} \mathbf{H}_D(X_1, X_2) &= \frac{1}{2}\log\det\left(\Sigma_Z + \beta I\right) + C_{d'} && (6) \\ &= \frac{1}{2}\log\det\left(\frac{Z^T Z}{n} + \beta I\right) + C_{d'} && (7) \end{aligned}$$

Furthermore, some prior works summarize determinant inequalities from the aspect of information theory (Cover & Thomas, 1988; Cai et al., 2015). These inequalities give a theoretical background to define the LogDet based mutual information measurements. Zhouyin & Liu (2021) show that given random variables X_1 and X_2, their joint LogDet entropy satisfies the following inequalities:

$$\mathbf{H}_D(X_1, X_2) \leq \mathbf{H}_D(X_1) + \mathbf{H}_D(X_2) \quad (8)$$

Given the joint LogDet entropy definition and the inequalities to guarantee the nonnegativity, we can safely define the LogDet based mutual information estimator as

$$\mathbf{I}_D(X_1; X_2) = \mathbf{H}_D(X_1) + \mathbf{H}_D(X_2) - \mathbf{H}_D(X_1, X_2) \quad (9)$$

2.1.1 LogDet Entropy estimator for non-Gaussian variable

As we discussed in the last section, when X is a Gaussian variable, the entropy $H(X)$ has a LogDet estimator expressed in Eqn 4. However, when X is a non-Gaussian variable, it is not straightforward to obtain the LogDet estimator. Here, we provide a LogDet-based upper bound of the entropy and also a convincing lower bound. The idea comes from the fact that we can approximate the distribution of non-Gaussian variables by a Gaussian mixture model, as the GMM is a powerful statistical tool that could approximate any density defined on $\mathbb{R}^d$ with a large enough number of mixture components. Algorithms for GMM fitting have been studied thoroughly. We can use the standard EM algorithm (Dempster et al., 1997) or its variants. Without going into the details of the GMM fitting algorithm, we assume the GMM has been fitted for the non-Gaussian variable. Support the fitted GMM of X is $p_\theta(X) = \sum_{i=1}^K \pi_i \mathcal{N}(\cdot|\mu_i, \Sigma_i)$. The entropy can be defined as follow,

$$\mathbf{H}(X) = \int -p_\theta(X)\log p_\theta(X) dX. \quad (10)$$

However, in this case, the entropy generally cannot be calculated in closed-form, due to the logarithm of a sum of exponential functions. We can use the Monte Carlo approach to numerically approximate the integration. However, the Monte Carlo approach always takes heavy computation efforts to obtain an accurate estimation.

Here, we provide a pair of computationally cheap lower and upper bounds to approximate the true entropy values. Both bounds can be calculated in closed form. The upper and lower bounds are given as follows; the detailed derivation is present in Appendix A.

Upper Bound. The upper bound consists of a weighted sum of the individual entropies of the Gaussian components. To be specified,

$$\begin{aligned} \mathbf{H}_D^u(X) &= \sum_{i=1}^K \pi_i \cdot \left(-\log\pi_i + \frac{1}{2}\log\left((2\pi e)^d |\Sigma_i|\right)\right). \\ &= \sum_{i=1}^K \pi_i \cdot \left(-\log\pi_i + \frac{1}{2}\log\det\left(\frac{X^T X}{n}\right) + C_d\right) \end{aligned}$$

Lower Bound. A lower bound of Eqn(10) can be obtained by employing the Jensen's inequality: the logarithm is moved outside the integral and only an integral of a product of two Gaussian densities remains, which has a well-known closed-form solution. A lower bound of Eqn(10) is given by

$$\mathbf{H}_D^l(X) = -\sum_{i=1}^{K} \pi_i \cdot \log \left(\sum_{j=1}^{K} \pi_j \cdot \eta_{i,j} \right),$$

with $\eta_{i,j} = \mathcal{N}(\mu_i; \mu_j, \Sigma_i + \Sigma_j)$.

Furthermore, we can also introduce the identity matrix in the LogDet function and covariance in the lower bound to obtain a more robust estimator, the resulting bounds become,

$$\mathbf{H}_D^l(X) = -\sum_{i=1}^{K} \pi_i \cdot \log \left(\sum_{j=1}^{K} \pi_j \cdot \mathcal{N}(\mu_i; \mu_j, \Sigma_i + \Sigma_j + \beta I) \right) \tag{11}$$

$$\mathbf{H}_D^u(X) = \sum_{i=1}^{K} \pi_i \cdot \left(-\log \pi_i + \frac{1}{2} \log \det \left(\frac{X^T X}{n} + \beta I \right) + C_d \right) \tag{12}$$

Given these bounds, we can approximate the entropy either by its bounds such as

$$\mathbf{H}_D^\star(X) \triangleq \mathbf{H}_D^u(X) \text{ or } \mathbf{H}_D^\star(X) \triangleq \mathbf{H}_D^l(X) \tag{13}$$

or by the means of the upper and lower bound

$$\mathbf{H}_D^\star(X) \triangleq= \mathbf{H}_D^m(X) = \frac{1}{2}\mathbf{H}_D^u(X) + \frac{1}{2}\mathbf{H}_D^l(X). \tag{14}$$

Finally, we obtain the mutual information LogDet based estimator (MILE) as follows,

$$\mathbf{I}_D^\star(X_1; X_2) \quad = \quad \mathbf{H}_D^\star(X_1) + \mathbf{H}_D^\star(X_2) - \mathbf{H}_D^\star(X_1, X_2) \tag{15}$$

where $\star$ represents $\{u, m, l\}$, which indicates the choice of different entropy estimator.

2.2 The issue of model selection

When introducing the GMM for entropy estimation, we actually introduce an extra model assumption for the data. From the Bayesian perspective, given the model assumption $\mathcal{K}$ (number of mixture components), the probability distribution of X is a marginal distribution by integrating out the model parameters, $p(X|\mathcal{K}) = \int p(X|\theta, \mathcal{K}) p(\theta|\mathcal{K}) d\theta$, where $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ is the model parameter of the GMM. Theoretically, a parametric entropy estimation should depend on the model assumption, $\mathbf{H}_\mathcal{K}(X) = \int -p(X|\mathcal{K}) \log p(X|\mathcal{K}) dX$. However, in practice, it is always infeasible to compute the marginal distribution $p(X|\mathcal{K})$ to evaluate the entropy. Instead, we find a maximum likelihood point estimation for θ, and use $p(X|\theta, \mathcal{K})$ in the LogDet based entropy estimation. Thus, the proposed entropy estimator depends on the model choice. Here, we use $\mathbf{I}_\mathcal{K}^\star$ to denote the MILE with K-components GMM. As shown in the simulation study, different model assumptions, such as different number of mixture components, will lead to significantly different entropy results. An improper choice of model assumption may lead to a large bias compared to the estimator with the known ground true model. Although finding the optimal number of mixture components when fitting a general dataset with GMM can be challenging, it is fortunate that in the context of self-supervised learning, which is the target application of the MILE estimator, the number of mixture components is known. This is because we treat the augmented data from one instance as a cluster, and this data augmentation strategy automatically clusters the data.

2.3 Issue of Self-consistency for Mutual Information Estimators

In line with previous research (Song & Ermon, 2019), we propose a set of self-consistency tests to evaluate the performance of the proposed estimator $\mathbf{I}$. These tests include: I. Independence: If X and Y are independent, then the estimated mutual information $\mathbf{I}(X; Y)$ should be zero. II. Data processing: For any functions g and h, the estimated mutual information $\mathbf{I}(X; Y)$ should be greater than or equal to $\mathbf{I}(g(X); h(Y))$. Additionally, the estimated mutual information should approximate

$\mathbf{I}([X, g(X)]; [Y, h(Y)])$, where $[\cdot, \cdot]$ represents concatenation. III. Additivity: If X_1 and X_2 are independent random variables with the same distribution as X (similarly defined for Y_1 and Y_2), then the estimated mutual information $\mathbf{I}\left([X_1, X_2]; [Y_1, Y_2]\right)$ should approximate twice the mutual information $\mathbf{I}(X, Y)$.

While these conditions are necessary, they are insufficient for accurate mutual information estimation. However, satisfying these self-consistency tests is highly desirable for effective representation learning (Chen et al., 2016). It is important to note that previous studies (Song & Ermon, 2019) have shown that the MINE estimator and its variants (such as SMILE) struggle to satisfactorily meet the self-consistency test. Simulation studies demonstrate that our proposed MILE estimator exhibits favorable performance on these self-consistency tests.

3 Application in Self-supervised learning

In self-supervised learning, a set of data X is given without knowing of the label. We aim to learn a deep neural network (DNN) ϕ to extract the feature Z from input X: $\phi : X \to Z$, which could perform well in downstream tasks such as classification or clustering. There are two potential perspectives on using the MI as loss for SSL: 1) maximize the mutual information between different views of data (Hjelm et al., 2019; Oord et al., 2018; Wang & Isola, 2020), that $\mathbf{I}(Z, Z')$, where $Z = \phi(X)$ and $Z' = \phi(X')$, X' is an augmentation or different view of X; 2) maximize the mutual information between the extracted feature Z and classification label Y_c (categorical variable) (Caron et al., 2018), here denoted as $\mathbf{I}(Z, Y_c)$. As the ground true label Y_c is unavailable, we can either generate pseudo labels (Caron et al., 2018) or create instance-based labels by data augmentation strategy Chen et al. (2020); Tong et al. (2023).

3.1 Maximize $\mathbf{I}(Z, Z')$ with MILE: alternative of InfoNCE loss

The primary perspective of using MI for SSL is to maximize the mutual information between different view of data, that $\mathbf{I}(Z, Z')$, where $Z = \phi(X)$ and $Z' = \phi(X')$, X' is an augmentation or different view of X. Here, we propose to use the MILE estimator for the learning loss,

$$\mathcal{L}(X, \phi) = -\mathbf{I}^\star_D\left(Z; Z'\right) = -\mathbf{H}^\star_D\left(Z\right) - \mathbf{H}^\star_D\left(Z'\right) + \mathbf{H}^\star_D\left(Z, Z'\right) \tag{16}$$

Notice that when we use the MI as loss function, the term C_d has no contribution to the gradient of Z, so we drop out this term in the entropy definition and the loss function.

As we discussed in section 2.2, we should pay attention to the model selection when using GMM based MILE. Generally, the entire set of feature Z or Z' should be Gaussian distributed, as Gaussian variables (have the largest entropy) can carry more information from the input X. So we should use $\mathbf{H}^\star_1(\cdot)$ that use one component in GMM, reduced to conventional LogDet estimator defined in Eqn (4), to estimate the entropy for Z and Z'. However, the paired data (Z, Z') has a strong clustering property: for each of the instances Z_c ($c = 1, ..., C$), we generate a small batch of augmented data $\{{Z_c}^{(i)'}\}_{i=1}^{I}$. Thus, the joint data $[Z_c, {Z_c}^{(i)'}]$ ($i = 1, ..I$) formulate an instance-based cluster that all the data $[Z_c, {Z_c}^{(i)'}]$ is close to the centroid $[Z_c, \hat{Z}'_c]$, where $\hat{Z}'_c$ is the mean of all augmented data from instance c. So, we can easily define a GMM by trading each $[Z_c, \hat{Z}'_c]$ as the clustering centers. Given the GMM, we can safely apply the entropy estimator defined in Eqn (13) or (14). For example, if we use the upper bound as the estimator, we have

$$\mathbf{H}^\star_D\left(Z, Z'\right) \triangleq \mathbf{H}^u_D(Z, Z') = \sum_{c=1}^{C} \pi_c \cdot \left(-\log \pi_c + \frac{1}{2} \log\det\left(\frac{\xi_c^T \xi_c}{n} + \beta I \right) \right) \tag{17}$$

where $\xi_c = [Z_c, Z_c^{(i)'}]$ ($c = 1, ..., C$ and $i = 1, ..., I$). We also drop out the term C_d in Eqn(17).

3.2 Maximize $\mathbf{I}(Z, Y_c)$ with entropy decomposition $\mathbf{I}(Z, Y_c) = \mathbf{H}(Z) - \mathbf{H}(Z|Y_c)$

The mutual information $\mathbf{I}(Z, Y_c)$ has a decompostion: $\mathbf{I}(Z, Y_c) = \mathbf{H}(Z) - \mathbf{H}(Z|Y_c)$. To build the MI loss, we first need to estimate the marginal entropy $\mathbf{H}(Z)$ and conditional entropy $\mathbf{H}(Z|Y_c)$. For $\mathbf{H}(Z)$, we can use the LogDet entropy estimator defined in Eqn 4. As the Gaussian variable has the largest entropy, maximizing $\mathbf{H}(Z)$ will encourage Z to become a Gaussian distributed variable;

therefore, the $\mathbf{H}_1^\star(\cdot)$ (aka the LogDet entropy estimator) is a proper choice. To estimate the conditional entropy $\mathbf{H}(Z|Y_c)$, a straightforward way is to augment the instance Z_c multiple times and gain a set $\xi_c = \{Z_c^{(i)}\}_{i=1}^I$, which served as a batch of data with the same label Y_c.

$$\mathbf{H}_D(Z|Y_c) = \frac{1}{2}\log\det(\Sigma_c + \beta I) = \frac{1}{2}\log\det\left(\frac{\xi_c^T\xi_c}{n_c} + \beta I\right)$$

Then, we can define loss function as follow,

$$\begin{aligned}\mathcal{L}(X,\phi) &\propto -\mathbf{H}(Z) + \lambda\mathbf{H}(Z|Y_c)\\ &= -\frac{1}{2}\log\det\left(\frac{Z^TZ}{n} + \beta I\right) + \frac{\lambda}{2}\log\det\left(\frac{\xi_c^T\xi_c}{n_c} + \beta I\right)\end{aligned} \tag{18}$$

where we use λ to balance the learning between maximizing the marginal entropy and minimizing the conditional entropy.

4 Related Works

The LogDet function has been widely regarded as an excellent matrix-based estimator for many years. Various applications, such as Jensen-Bregman LogDet Divergence (Cherian et al., 2012), Alpha-Beta and Gamma Divergences (Cichocki et al., 2015), have shown that it possesses desirable properties, including robustness to noise, positional bias, and outliers, while maintaining computational efficiency. Additionally, it has been extensively studied as a universal entropy measurement for multivariate random Gaussian variables. The LogDet entropy estimator has been utilized to estimate mutual information and applied in the analysis of information compression in deep neural networks (Zhouyin & Liu, 2021). However, previous studies have predominantly focused on multivariate Gaussian variables. It has been observed through our simulation study, that the LogDet entropy estimator exhibits substantial bias when applied to non-Gaussian distributed variables. In this work, we address the challenge of entropy estimation for non-Gaussian variables and identify suitable applications in both MI estimation and SSL for deep neural networks.

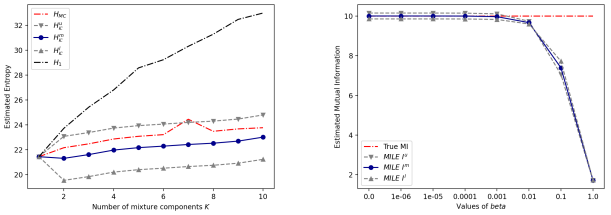
The LogDet entropy estimator can also be viewed as an extension of the Coding Length Function (CLF). The coding length function is raised to estimate Distortion Rate from Rate Distortion Theory (Cover, 1999). The CLF of n samples from arbitrary distribution, $\mathbf{X} \in \mathbb{R}^{n\times d}$, is given by: $R(X) = \frac{1}{2}\log\det\left(I + \frac{d}{n\epsilon^2}X^TX\right)$, where ϵ is the allowable distortion. Compared to our proposed LogDet estimator in Eqn(4), the difference is we put a scaling hyperparameter β on the identity matrix I, while the CLF put a controlling hyperparameter on the covariance. As we show in the experiments, our LogDet estimator can lead to an accurate estimation for both entropy and mutual information. However, the CLF and also the CLF based principle of Maximal Coding Rate Reduction (Yu et al., 2020; Tong et al., 2023) performs well in SSL but can not be used as MI estimator.

5 Simulation studies

5.1 Ablation study

As discussed in section 2.2, we should choose a proper number of components in using the GMM based entropy estimator $\mathbf{H}^\star(\cdot)$ defined in Eqn(13) or (14). Here, we present an experiment to show that when the ground true data is a mixture of Gaussian, the difference of estimated entropy using $\mathbf{H}_\mathcal{K}^\star(\cdot)$ (K-components GMM based entropy estimator) vs $\mathbf{H}_1^\star(\cdot)$ (single Gaussian based entropy estimator in Eqn (4)). The data is sampled from 20-dimensional Gaussian mixtures with the number of components varying from $K = 1$ to 10. The mean of each component is randomly drawn from $N(\cdot|0, I_{20})$ while the covariance is fixed $0.5 * I_{20}$. We use the following five estimators: $\mathbf{H}_{MC}$ the Monte Carlo entropy estimation, $\mathbf{H}_\mathcal{K}^u$ upper bound entropy estimator with known K-component, $\mathbf{H}_\mathcal{K}^m$ lower bound estimator with known K-components, $\mathbf{H}_\mathcal{K}^l$the mean of $\mathbf{H}_\mathcal{K}^u$ and $\mathbf{H}_\mathcal{K}^l$, $\mathbf{H}_1$ the LogDet entropy estimator. As shown in Figure 1(a), the single Gaussian based LogDet estimator $\mathbf{H}_1$ highly overestimates the entropy. It also confirms that when the data has intrinsic clustering shaping, the single Gaussian based LogDet entropy estimator can not provide a reasonable estimation, while our proposed entropy estimator $\mathbf{H}^\star(\cdot)$ is favorable.

Here, we instigate the hyperparameter β. We take the MI estimation of correlated Gaussian random variables. We show that the estimated MI varies over $\beta = [0.0, 1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1]$. As shown in Figure 1(b), the estimated MI is close to the ground true when β is sufficiently small. So, we recommend $\beta = 1e^{-3}$ in the following simulation studies. Besides, we can see the bounds also well bracket the true MI values when β is small enough.



(a) Entropy estimation (b) MI estimation with different β

Figure 1: Result of ablation study.

5.2 Benchmarking on Multivariate Gaussians

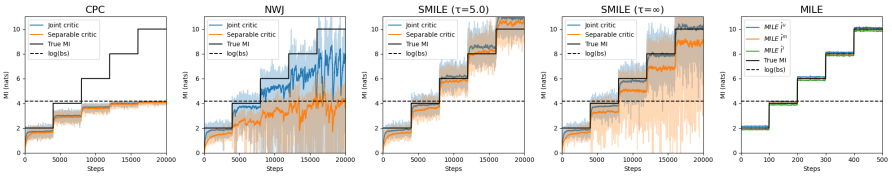
In this experiment, we evaluate the performance of mutual information (MI) bounds on three toy tasks described in detail in the works ofBelghazi et al. (2018); Poole et al. (2019); Song & Ermon (2019). The first task, referred to as 'Gaussian' involves drawing samples (X_a, X_b) from a 20-dimensional Gaussian distribution with correlation. The second and third tasks, known as 'Sin' and 'Cubic' are similar to the Gaussian task, but apply a specific transformation ($X_b \leftarrow sin(X_b)$,$X_b \leftarrow X_b^3$). We generate multivariate Gaussian random variables, denoted as X_a and X_b, with component-wise correlation to conduct the experiment. The correlation between X_a^i and X_b^j is denoted as $corr(X_a^i, X_b^j) = \delta_{ij}\rho$, where ρ takes values between -1 and 1, and δ_{ij} represents Kronecker's delta. We consider five approaches for estimating mutual information: CPC: contrastive predictive coding (Oord et al., 2018), NWJ: NWJ variational approximation based Mutual Information Neural Estimator (Belghazi et al., 2018), SMILE: moothed Mutual Information Lower-bound Estimator with hyperparameter τ (Song & Ermon, 2019) ($\tau = 5$ and $\tau = \infty$) and our proposed MILE estimator. The first four models are trained for 20,000 iterations, with the ground truth mutual information increasing by 2 every 4,000 iterations. The first four models require extra training of the DNN based critic function, so they need pretty long steps to reach a stable MI estimation. However, our MILE estimator does not require extra training, so we show only 100 steps for each MI step. Additional training details can be found in Appendix B.

Figure 2 shows the estimated MI over the number of iterations. In all tasks, CPC has high bias; NWJ and SMILE ($\tau = \infty$) has high variances. SMILE ($\tau = 5$) has a much smaller variance. All these four estimators take certain long steps to reach a stable MI estimation. As it is obvious, our MILE significantly outperforms these previous estimators in obtaining low bias and low variance estimation. Moreover, it does not require extra training, so it works well since the first step.

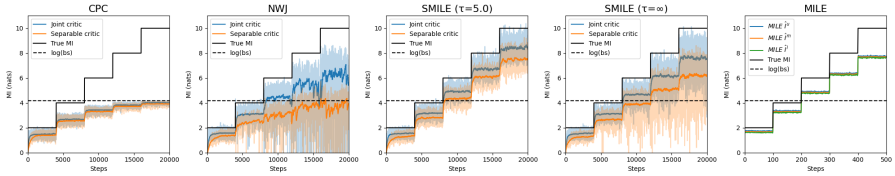
We also compare the bias, variance and mean squared error (MSE) of the five estimators. The result is shown in Appendix B. For some estimator, such as NWJ, the variance increases exponentially with mutual information. With the smooth hyperparamter for the critic function controlled by τ, SMILE have a significantly smaller variance than while having a similar bias. However, our MILE estimator produces extremely small variance while keeping a favorable small bias.

5.3 Self-consistency Tests on Images

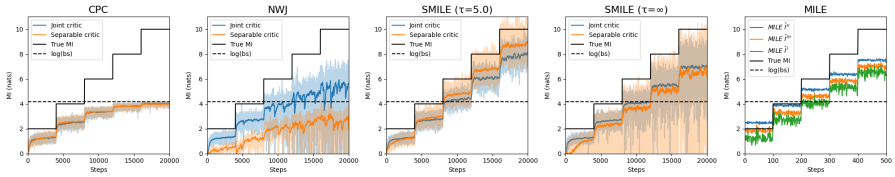
In line with the work of Song & Ermon (2019), we conduct evaluations of self-consistency tests on high-dimensional images from the MNIST and CIFAR10 datasets. The first setting involves an image denoted as X_a, where X_b is obtained by masking the bottom rows of X_a, leaving only the top t rows. This choice of X_b serves two purposes: 1) $\mathbf{I}(X_a; X_b)$ should exhibit a non-decreasing trend with increasing t, and 2) it provides a more intuitive understanding of the amount of information retained in X_b compared to low-dimensional representations. In the second setting, X_a represents



(a) 'Gaussian'



(b) 'Sin'



(c) 'Cubic'

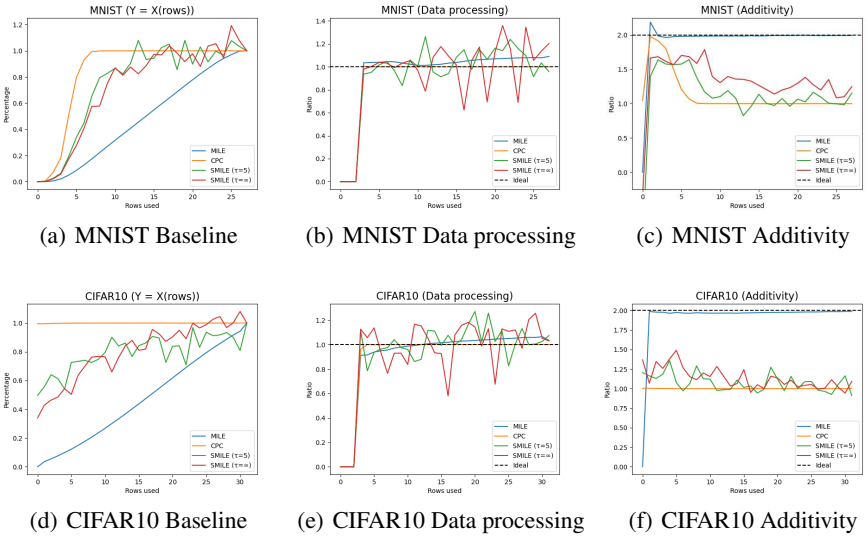
Figure 2: Performance of mutual information estimation approaches on 'Gaussian', 'Sin' and 'Cubic'.

two identical images, while X_b is constructed by taking the top t_1 and t_2 rows from the two images, respectively (with $t_1 \geq t_2$). This setting explores the data-processing property of MI. The third setting considers X_a as two independent images and X_b as the top t rows from both images. This setting examines the additivity property of MI.

We compare four approaches: CPC, SMILE ($\tau = 5$), SMILE ($\tau = \infty$) and SMILE. The result is summarised as follows: 1) *Baselines.* We evaluate the first setting with X_b having varying number of rows t in Figure 3(a) and 3(d), where the estimations are normalized by the estimated $\mathbf{I}(X_a; X_a)$. Most methods predict zero MI when X_a and X_b are independent, passing the first self-consistency test. Moreover, the estimated MI is non-decreasing with increasing t, but with different slopes. Our MILE presents a linear increase with t; 2) *Data-processing.* In the second setting we set $t_2 = t_1 - 3$. Ideally, the estimator should satisfy $\mathbf{I}([X_a, X_a]; [X_b, h(X_b)])/\mathbf{I}(X_a, X_b) \approx 1$, as additional processing should not increase information. We show the above ratio in Figure 3(b) and 3(e) under varying t_1 values. All methods perform well in both datasets, while MILE has a significantly small variance and is much closer to the ground true value 1; 3) *Additivity*. In the third setting, the estimator should double its value compared to the baseline with the same t, i.e. $\mathbf{I}\left([X_{a1}, X_{a2}]; [X_{a1}, X_{a2}]\right)/\mathbf{I}(X_a, X_b) \approx 2$. Figure 3(c) and 3(f) show the above ratio under different values of t. None of the MINE type estimators worked well in this case except when t is very small, when t is large this ratio converges to 1. However, our MILE performs perfectly on this test for all values of t.

5.4 Self-supervised learning

We apply the MILE as loss function for self-supervised learning. Following the extreme multi-patch (EMP) strategy based SSL (EMP-SSL)Tong et al. (2023), we replace the loss function in EMP-SSL with three mutual information-based loss functions: InfoNCE (Oord et al., 2018), SMILE (Song & Ermon, 2019) and our MILE (denoted as EMP-SMILE, EMP-InfoNCE, and EMP-MILE). We provide empirical results on the standard dataset CIFAR-10. For all the experiments, we use a ResNet-18 (He et al., 2016) as the backbone and train for at most 30 epochs. We use a batch size of 100, the LARS optimizer You et al. (2017) with η set to 0.005, and a weight decay of $1e^{-4}$. The learning rate is set to 0.3 and follows a cosine decay schedule with a final value 0. In the MILE based loss, λ is set to [0.01,0.1,1.0,2.0] and β is set to $[1e^{-2}, 1e^{-3}]$. By validation, $\lambda = 1.0$ and



(a) MNIST Baseline (b) MNIST Data processing (c) MNIST Additivity

(d) CIFAR10 Baseline (e) CIFAR10 Data processing (f) CIFAR10 Additivity

Figure 3: Evaluation of self-consistency.

	CIFAR-10			
Methods	1 Epoch	10 Epochs	30 Epochs	1000 Epochs
SimCLR	0.282	0.565	0.663	0.910
BYOL	0.249	0.489	0.684	0.926
VICReg	0.406	0.697	0.781	0.921
SwAW	0.245	0.532	0.767	0.923
EMP-SSL(20 patches)	0.806	0.907	0.931	\
EMP-SMILE(20 patches)	0.315	0.242	0.325	\
EMP-InfoNCE(20 patches)	0.753	0.789	0.818	\
EMP-MILE(20 patches)	0.768	0.802	0.859	\

Table 1: Performance of the MI based SSL and standard self-supervised SOTA methods with different epochs. Accuracy is measured by training linear classifier on learned representation.

$\beta = 1e^{-2}$ gives the best performance in the downstream task. The projector network consists of 2 linear layers with respectively 4096 hidden units and 1024 output units. The data augmentations used are identical to those of VICReg (Bardes et al., 2022).

For evaluation metrics, We train an additional linear classifier to evaluate the performance of the learned representation. The additional classifier is trained with 100 epochs, optimized by SGD optimizer with a learning rate of 0.03. We also compared our results with several self-supervied SOTA methods: SimCLR, BYOL, VICReg, SwAW. The result is summarized in Table 1 (more results are provided in Appendix B). From the comparison, we find that the SMILE can not obtain competitve results. EMP-SSL gains superior performance. InfoNCE can obtain a favorable performance, while our MILE can outperform InfoNCE loss. This experiments confirm that as a tighter estimator of MI, MILE can also gain comparable performance with InfoNCE.

6 Discussion

This work addresses a simple but effective way to estimate mutual information between high-dimensional variables given samples from them. Unlike conventional variational mutual information, which needs the training of neural network based critic function, which is known as the reason to cause a high variance of variational mutual information estimator, our approach can gain high accuracy and low variance estimate without this extra training. Besides, the MILE well pass the self-consistency test. We also show that being a tighter estimation of MI, MILE can gain competitive performance in self-supervised learning.

REFERENCES

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2016.

David Barber and Felix Agakov. The im algorithm: a variational approach to information maximization. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, pp. 201–208, 2003.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *10th International Conference on Learning Representations, ICLR 2022*, 2022.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pp. 531–540. PMLR, 2018.

T Tony Cai, Tengyuan Liang, and Harrison H Zhou. Law of log determinant of sample covariance matrix and optimal estimation of differential entropy for high-dimensional gaussian distributions. *Journal of Multivariate Analysis*, 137:161–172, 2015.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 132–149, 2018.

Chung Chan, Ali Al-Bashabsheh, Hing Pang Huang, Michael Lim, Da Sun Handason Tam, and Chao Zhao. Neural entropic estimation: A faster path to mutual information estimation. *arXiv preprint arXiv:1905.12957*, 2019.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

Anoop Cherian, Suvrit Sra, Arindam Banerjee, and Nikolaos Papanikolopoulos. Jensen-bregman logdet divergence with application to efficient similarity search for covariance matrices. *IEEE transactions on pattern analysis and machine intelligence*, 35(9):2161–2174, 2012.

Andrzej Cichocki, Sergio Cruces, and Shun-ichi Amari. Log-determinant divergences revisited: Alpha-beta and gamma log-det divergences. *Entropy*, 17(5):2988–3034, 2015.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Thomas M Cover and A Thomas. Determinant inequalities via information theory. *SIAM journal on Matrix Analysis and Applications*, 9(3):384–392, 1988.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM - algorithm plus discussions on the paper. *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1997.

Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. Efficient estimation of mutual information for strongly dependent variables. In *Artificial intelligence and statistics*, pp. 277–286. PMLR, 2015.

Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research*, 13 (2), 2012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pp. 875–884. PMLR, 2020.

Ilya Nemenman, William Bialek, and Rob De Ruyter Van Steveninck. Entropy and information in neural spike trains: Progress on the sampling problem. *Physical Review E*, 69(5):056111, 2004.

XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pp. 5171–5180. PMLR, 2019.

Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. In *International Conference on Learning Representations*, 2019.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839, 2020.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE information theory workshop (itw)*, pp. 1–5. IEEE, 2015.

Shengbang Tong, Yubei Chen, Yi Ma, and Yann Lecun. Emp-ssl: Towards self-supervised learning in one training epoch. *arXiv preprint arXiv:2304.03977*, 2023.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pp. 9929–9939. PMLR, 2020.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. *Advances in Neural Information Processing Systems*, 33:9422–9434, 2020.

Zhanghao Zhouyin and Ding Liu. Understanding neural networks with logarithm determinant entropy estimator. *arXiv preprint arXiv:2105.03705*, 2021.

A. Lower and upper bound for entropy of GMM

Derivation of upper bound

By separating the i-th component of $p_\theta(x) = \sum_{i=1}^{K} \pi_i \cdot \mathcal{N}(x; \mu_i, \Sigma_i)$, the entropy for x can be written as

$$
\begin{aligned}
H(x) &= -\int_{\mathbb{R}^N} \sum_{i=1}^{K} \pi_i \cdot \mathcal{N}(x; \mu_i, \Sigma_i) \cdot \log \left(\sum_{j=1}^{K} \pi_j \cdot \mathcal{N}(x; \mu_j, \Sigma_j) \right) \mathrm{d}y \\
&= -\sum_{i=1}^{K} \pi_i \int_{\mathbb{R}^N} \mathcal{N}(x; \mu_i, \Sigma_i) \cdot \log \left(\pi_i \cdot \mathcal{N}(x; \mu_i, \Sigma_i) \cdot (1 + \epsilon_i) \right) \mathrm{d}y \\
&= -\sum_{i=1}^{K} \pi_i \int_{\mathbb{R}^N} \mathcal{N}(x; \mu_i, \Sigma_i) \cdot \left(\log \left(\pi_i \cdot \mathcal{N}(x; \mu_i, \Sigma_i) \right) + \log(1 + \epsilon_i) \right) \mathrm{d}y,
\end{aligned}
$$

where

$$\epsilon_i = \frac{\sum_{i \neq j=1}^{K} \pi_j \cdot \mathcal{N}\left(x; \mu_j, \Sigma_j\right)}{\pi_i \cdot \mathcal{N}\left(x; \mu_i, \Sigma_i\right)}$$

Since $\log\left(1+\epsilon_i\right)$ in (9) is always non-negative, neglecting it yields the upper bound,

$$\begin{aligned} H(x) &\leq -\sum_{i=1}^{K} \pi_i \int_{\mathbb{R}^N} \mathcal{N}\left(x; \mu_i, \Sigma_i\right) \cdot \left(\log\left(\pi_i \cdot \mathcal{N}\left(x; \mu_i, \Sigma_i\right)\right)\right) \mathrm{d}x \\ &= -\sum_{i=1}^{K} \pi_i \cdot \left(\log \pi_i + \frac{1}{2} \log\left((2\pi e)^N \left|\Sigma_i\right|\right)\right) \end{aligned}$$

Derivation of lower bound

Since $-\log(x)$ is concave in x, Jensen's inequality can be employed. Thus, with $-\log \mathrm{E}[x] \leq \mathrm{E}[-\log(x)]$, we obtain a lower bound of the entropy

$$H(x) = \int_{\mathbb{R}^N} \sum_{i=1}^{K} \pi_i \cdot \mathcal{N}\left(x; \mu_i, \Sigma_i\right) \cdot \log p_\theta(x) \mathrm{d}x$$

according to

$$\begin{aligned} H(x) &= -\sum_{i=1}^{K} \pi_i \cdot \int_{\mathbb{R}^N} \mathcal{N}\left(x; \mu_i, \Sigma_i\right) \cdot \log p_\theta(x) \mathrm{d}x \\ &\geq -\sum_{i=1}^{K} \pi_i \cdot \log\left(\int_{\mathbb{R}^N} \mathcal{N}\left(x; \mu_i, \Sigma_i\right) \cdot p_\theta(x) \mathrm{d}x\right) \\ &= -\sum_{i=1}^{K} \pi_i \cdot \log\left(\sum_{j=1}^{L} \pi_j \cdot \eta_{i,j}\right) \end{aligned}$$

with the constant

$$\begin{aligned} \eta_{i,j} &= \int_{\mathbb{R}^N} \mathcal{N}\left(x; \mu_i, \Sigma_i\right) \cdot \mathcal{N}\left(x; \mu_j, \Sigma_j\right) \mathrm{d}x \\ &= \mathcal{N}\left(\mu_i; \mu_j, \Sigma_i + \Sigma_j\right) \end{aligned}$$

B. Additional Experimental Details

Benchmark Tasks

Tasks We sample each dimension of (X_a, X_b) independently from a correlated Gaussian with mean 0 and correlation of ρ, where $\mathcal{X} = \mathbb{R}^{20}$. The true mutual information is computed as: $I(X_a, X_b) = -\frac{d}{2}\log\left(1-\frac{\rho}{2}\right)$. The initial mutual information is 2 , and we increase the mutual information by 2 every $4k$ iterations, so the total training iterations is $20k$.

Architecture and training procedure For CPC, NWJ and SMILE methods, we follows the architectures detailed in (Song & Ermon, 2019): The joint architecture concatenates the inputs X_a, X_b, and then passes through a two layer MLP with 256 neurons in each layer with ReLU activations at each layer. The separable architecture learns two separate neural networks for X_a and X_b, each is a two layer MLP with 256 neurons in each layer with ReLU activations at each layer; the output are 32 dimensions. For all the cases, we use with the Adam optimizer with learning rate 5×10^{-4} and $\beta_1 = 0.9, \beta_2 = 0.999$ and train for $20k$ iterations with a batch size of 64 , following the setup in (Poole et al., 2019).

Additional results We show the bias, variance and mean squared error in Table 2-4.

	MI	2	4	6	8	10
Bias	CPC	0.20	0.99	2.31	4.00	5.89
	NWJ	0.10	0.21	0.34	2.37	5.80
	SMILE ($\tau = \infty$)	0.11	0.27	0.17	0.14	0.04
	SMILE ($\tau = 5.0$)	0.16	0.12	0.002	0.56	1.05
	MILE	**0.0111**	**0.0025**	**0.0072**	**0.0152**	**0.022**
Var	CPC	0.03	0.05	0.01	0.004	0.003
	NWJ	0.06	0.16	0.63	0.55	3.36
	SMILE ($\tau = \infty$)	0.08	0.12	0.26	0.66	1.51
	SMILE ($\tau = 5.0$)	0.05	0.10	0.15	0.24	0.36
	MILE	**0.0002**	**0.0003**	**0.0006**	**0.0007**	**0.0008**
MSE	CPC	0.07	1.03	5.34	16.01	34.74
	NWJ	0.07	0.21	0.74	6.18	36.96
	SMILE ($\tau = \infty$)	0.09	0.19	0.29	0.68	1.51
	SMILE ($\tau = 5.0$)	0.07	0.12	0.15	0.55	1.46
	MILE	**0.0003**	**0.0003**	**0.0007**	**0.001**	**0.0013**

Table 2: Bias, Variance and MSE of the MI estimators for 'Gaussian' data

	MI	2	4	6	8	10
Bias	CPC	0.51	1.29	2.57	4.17	5.99
	NWJ	0.44	0.94	1.39	2.36	3.09
	SMILE ($\tau = \infty$)	0.40	0.94	1.25	1.88	2.33
	SMILE ($\tau = 5.0$)	0.42	0.81	1.22	**1.21**	**1.87**
	MILE	**0.3177**	**0.7078**	**1.1646**	1.6941	2.3088
Var	CPC	0.03	0.04	0.03	0.01	0.004
	NWJ	0.04	0.12	0.23	1.74	3.10
	SMILE ($\tau = \infty$)	0.04	0.09	0.23	0.34	1.46
	SMILE ($\tau = 5.0$)	0.03	0.10	0.10	0.21	0.15
	MILE	**0.0002**	**0.0003**	**0.0005**	**0.0005**	**0.0006**
MSE	CPC	0.29	1.70	6.62	17.41	35.88
	NWJ	0.23	1.01	2.15	7.30	12.64
	SMILE ($\tau = \infty$)	0.20	0.97	1.80	3.88	6.90
	SMILE ($\tau = 5.0$)	0.20	0.75	1.59	**1.68**	**3.65**
	MILE	**0.1011**	**0.5013**	**1.3568**	2.8703	5.331

Table 3: Bias, Variance and MSE of the MI estimators for 'Sin' data

	MI	2	4	6	8	10
Bias	CPC	0.69	1.47	2.59	4.20	5.99
	NWJ	0.64	1.13	2.39	2.72	4.23
	SMILE ($\tau = \infty$)	0.69	1.26	1.83	2.55	2.65
	SMILE ($\tau = 5.0$)	0.68	1.08	1.60	1.47	3.33
	MILE	**0.1325**	**0.7117**	**1.3899**	**2.1509**	**2.9758**
Var	CPC	0.03	0.04	0.02	0.01	0.01
	NWJ	0.04	0.09	0.22	0.27	0.38
	SMILE ($\tau = \infty$)	0.03	0.10	0.17	0.60	0.93
	SMILE ($\tau = 5.0$)	0.03	0.07	0.10	0.14	0.34
	MILE	**0.0127**	**0.0117**	**0.0161**	**0.0215**	**0.0161**
MSE	CPC	0.52	2.19	6.74	17.63	35.91
	NWJ	0.45	1.36	5.93	7.67	18.26
	SMILE ($\tau = \infty$)	0.50	1.69	3.54	7.12	**7.94**
	SMILE ($\tau = 5.0$)	0.49	1.25	2.67	**2.29**	11.41
	MILE	**0.0303**	**0.5182**	**1.948**	4.6478	8.8718

Table 4: Bias, Variance and MSE of the MI estimators for 'Cubic' data

SELF-CONSISTENCY EXPERIMENTS

Tasks We consider three tasks with the MI estimator $\mathbf{I}$: 1. $\mathbf{I}(X_a; X_b)$ where X_a is an image from MNIST or CIFAR10 and X_b is the top t rows of X_a. To simplify architecture designs, we simply mask out the bottom rows to be zero, see Figure 3 . 2. $\mathbf{I}([X_a, X_a]; [X_b; h(X_b)])$ where X_a is an image, X_b is the top t rows of X_a, $h(X_b)$ is the top $(t-3)$ rows of X_b and $[\cdot, \cdot]$ denotes concatenation. The prediction should be close to $\mathbf{I}(X_a; X_b)$. 3. $\mathbf{I}\left([X_{a_1}, X_{a_2}], [X_{b_1}, X_{b_2}]\right)$ where X_{a_1} and X_{a_2} are independent images from MNIST or CIFAR10, X_{b_1} and X_{b_2} are the top t rows of X_{a_1} and X_{a_2} respectively. Ideally, this prediction should be close to $2 \cdot \mathbf{I}(X_a; X_b)$.

Architecture and training procedure For CPC, NWJ and SMILE methods, we follows the architectures detailed in (Song & Ermon, 2019): The first layer is a convolutional layer with 64 output channels, kernel size of 5 , stride of 2 and padding of 2 ; the second layer is a convolutional layer with 128 output channels, kernel size of 5 , stride of 2 and padding of 2 . This is followed another fully connected layer with 1024 neurons and finally a linear layer that produces an output of 1 . All the layers (except the last one) use ReLU activations. We stack variables over the channel dimension to perform concatenation. For all the cases, we use with the Adam optimizer with learning rate 10^{-4} and $\beta_1 = 0.9, \beta_2 = 0.999$.