Roundtrip: A Deep Generatative Neural Density Estimator

Qiao Liu[1,2], **Jiaze Xu**[1,2], **Rui Jiang**[1], and **Wing Hung Wong**[2]

[1]Tsinghua University
{liu-q16,xjz16}@mails.tsinghua.edu.cn, ruijiang@tsinghua.edu.cn
[2]Stanford University
{liuqiao,jxu3,whwong}@stanford.edu

Abstract

Density estimation is a fundamental problem in both statistics and machine learning. We proposed Roundtrip as a universal neural density estimator based on deep generative models. Roundtrip exploits the advantage of GANs for generating samples and estimates density by either importance sampling or Laplace approximation. Unlike prior studies modeling target density by constructing a tractable Jacobian $w.r.t$ to a base density (e.g., Gaussian), Roundtrip learns target density by generating a manifold from a base density to approximate the distribution of observation data. In a series of experiments, Roundtrip achieves state-of-the-art performance in a diverse range of density estimation tasks.

1 Introduction

Density estimation is a fundamental problem in statistics. Let $p(\cdot)$ be a density on a n-dimensional Euclidean space $\mathcal{X}$. Our task is to estimate the density $p(\cdot)$ based on a set of independently and identically distributed data points $\{\mathbf{x}_i\}_{i=1}^N$ drawn from this density.

A large number of density estimators have been proposed. Histogram is perhaps the simplest nonparametric density estimator which partitions $\mathcal{X}$ into rectangular regions and estimate the density by calculating the frequency of data points in each region. Data-driven histograms were studied by [28, 16]. Histogram-based methods are usually used in a univariate case due to the low efficiency. Kernel-based method [26, 22] is another popular nonparametric density estimator where the density is estimated as a convolution of a kernel function with the empirical distribution of the observations. However, the success of kernel density estimator (KDE) heavily depends on the bandwidth and kernel function. KDE is typically not effective in problems of dimension higher than five [15]. Density estimation in high-dimensional or highly structured data remains a challenging problem.

Recently, neural networks have been applied for constructing density estimators. There are mainly two families of such neural density estimators: *autoregressive models* [31, 6, 21] and *normalizing flows* [25, 2, 4]. Autoregression-based neural density estimators decompose the density into the product of conditional distribution based on probability chain rule $p(\mathbf{x}) = \prod_i p(x_i|\mathbf{x}_{1:i-1})$. Then each conditional probability $p(x_i|\mathbf{x}_{1:i-1})$ is modeled by a parametric density (e.g., Gaussian or mixture of Gaussian), of which the parameters are learned by a neural network. Density estimators based on normalizing flows represent $\mathbf{x}$ as an invertible transformation of a latent variable $\mathbf{z}$ with known density, where the invertible transformation is a composition of a series of simple funcitons whose Jacobian is easy to compute. The parameters of these component functions are then learned by neural networks.

Kingma et al [11] first pointed out autoregressive models can be equivalently interpreted as normalizing flow where the tractable Jacobian in the autoregressive model is a triangular structure. In

Preprint. Under review.

principle, most of the current neural density estimators can be summarized as follows. Given a differentiable and invertible mapping $g : \mathbb{R}^n \to \mathbb{R}^n$ and a base density $p(\mathbf{z})$, the density of $\mathbf{x} = G(\mathbf{z})$ can be represented using the *change of variable rule* as

$$p(\mathbf{x}) = p(\mathbf{z})|\det(\mathbf{J}_z)|^{-1} \tag{1}$$

where Jacobian matrix $\mathbf{J}_z = \frac{\partial G(\mathbf{z})}{\partial \mathbf{z}^T}$. To achieve efficient and practical computation of the Jacobian matrix, previous neural density estimators carefully designed model architectures to impose a strong constrain on the Jacobian matrix. For example, [3] constructed a low rank perturbations of a diagonal matrix as Jacobian, [21, 4, 11] used triangular matrices as Jacobian, [10, 9] designed special convolutions to achieve a block diagonal Jacobian. However, such neural density estimators based on tractable Jacobian may have the following two major limitations. 1) The strong constrain on Jacobian may sacrifice the expressiveness of neural networks due to the feature dependency. For example, density estimators based on autoregressive model which assume that the latter features only depend on previous ones are naturally sensitive to the order of the features. 2) *change of variable rule* requires equal dimension in base density and target density. The computation of Jacobian might still be inefficient in high dimensional cases. No previous density estimators have considered the case where dimension in base density and target density is not equal.

Motivated by the advances of deep generative models, such as GAN [7], cycleGAN [33] and Adversarial autoencoder [17], we proposed Roundtrip as a universal neural density estimator using deep generative models (Figure 1). There are two major characteristics that differ Roundtrip from previous neural density estimators. 1) Roundtrip directly uses neural networks to generate samples that are similar to observation data based on deep generative models. In contrast, previous neural density estimators use neural networks only to represent the component functions that are used for building up the invertible transformation. 2) Roundtrip maps the base density in a low dimension space to the target density which can be approximated by a learned manifold while previous methods require equal dimension in base density and target density. More importantly, we proposed two strategies to estimate the density by either importance sampling or a derived closed-form approximation. Here, we summarize our major contributions in this study as follows.

- We proposed Roundtrip as a universal neural density estimator based on deep generative models. Roundtrip requires much less model assumptions compared to previous neural density estimators.
- We provided theoretical guarantees which ensure the feasibility of density estimation with deep generative models. Specifically, we proved that *change of variable rule* contained in previous neural density estimators is a special case in our Roundtrip framework.
- We demonstrated state-of-the-art performance of Roundtrip model through a series of experiments, including density estimation tasks in simulations as well as in several real data applications.

2 Methods

2.1 Density modeling in Roundtrip

Consider two random variables $\mathbf{z} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$ where $\mathbf{z}$ has a known density $p(\mathbf{z})$ (e.g., standard Gaussian) and $\mathbf{x}$ is distributed according to a target density $p(\mathbf{x})$ that we intend to estimate based on $i.i.d$ observations from it. We introduced two functions $G(\cdot)$ and $H(\cdot)$ for learning an forward and backward mapping relationship between the above two distributions. The above two functions are two major components in Roundtrip which are implictly learned by two neural netowrks (Figure 1). We denote $G(\mathbf{z}) = \tilde{\mathbf{x}}$ and $H(\mathbf{x}) = \tilde{\mathbf{z}}$. We assume that the forward mapping error follows a Gaussian distribution

$$\mathbf{x} = \tilde{\mathbf{x}} + \boldsymbol{\epsilon}, \epsilon_i \sim N(0, \sigma^2) \tag{2}$$

Typically, we set $m < n$, which means that $\tilde{\mathbf{x}}$ takes values in a manifold of $\mathbb{R}^n$ with intrinsic dimension m. Basically, this roundtrip model utilizes $G(\cdot)$ to produce a manifold and then approximate the target density as a mixture of Gaussians where the mixing density is the induced density $p(\tilde{\mathbf{x}})$ on the manifold. In what follows, we will set $p(\mathbf{z})$ to be a standard Gaussian $p(\mathbf{z}) = (\frac{1}{\sqrt{2\pi}})^m e^{-\frac{\|\mathbf{z}\|_2^2}{2}}$. Then,

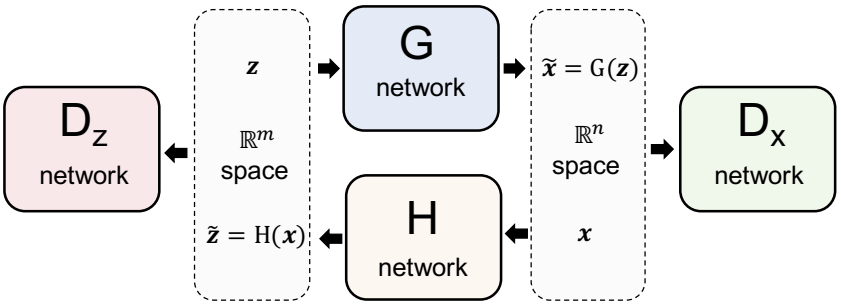

Figure 1: The overview framework of Roundtrip.

the target density can be expressed as

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} = (\frac{1}{\sqrt{2\pi}})^{m+n}\sigma^{-n}\int e^{-\frac{v(\mathbf{x},\mathbf{z})}{2}}d\mathbf{z} \tag{3}$$

where $v(\mathbf{x},\mathbf{z}) = \|\mathbf{z}\|_2^2 + \sigma^{-2}\|\mathbf{x} - G(\mathbf{z})\|_2^2$. The density estimation problem has been tranformed to computing the integral in equation (3). We can compute this integral approximately by either importance sampling or Laplace approximation.

Importance sampling If we directly sample from $p(\mathbf{z})$ by discretizing expectation, then $\int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} = E_{z\sim p(\mathbf{z})}p(\mathbf{x}|\mathbf{z}) \approx \frac{1}{N}\sum_i^N p(\mathbf{x}|\mathbf{z}_i)$ where $\mathbf{z}_i \sim p(\mathbf{z})$. Such sampling maybe extremely inefficient as $p(\mathbf{x}|\mathbf{z}_i)$ typically takes low values at most values of $\mathbf{z}_i$ sampled from $p(\mathbf{z})$. In importance sampling, we sample $\mathbf{z}_i$ from an importance distribution $q(\mathbf{z})$ instead of base density $p(\mathbf{z})$ which can be represented by

$$\int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z})w(\mathbf{z})q(\mathbf{z})d\mathbf{z} = E_{z\sim q(\mathbf{z})}p(\mathbf{x}|\mathbf{z})w(\mathbf{z}) \approx \frac{1}{N}\sum_i^N p(\mathbf{x}|\mathbf{z}_i^q)w(\mathbf{z}_i^q) \tag{4}$$

where N is the sample size, $w(\mathbf{z}) = \frac{p(\mathbf{z})}{q(\mathbf{z})}$ is the importance weight function, $\{\mathbf{z}_i^q\}_{i=1}^N$ are $i.i.d$ samples from $q(\mathbf{z})$. We propose to set $q(\mathbf{z})$ to be a Student's t distribution with the center at $\tilde{\mathbf{z}} = H(\mathbf{x})$. This choice is motivated by the following considerations. 1) For a given $\mathbf{x}$, p($\mathbf{x}$|$\mathbf{z}$) is likely to be maximized at values of $\mathbf{z}$ near $\tilde{\mathbf{z}} = H(\mathbf{x})$. 2) Student's t distribution has a heavier tail than Gaussian which provides a control of the variance of the summand in (4) . See more details in Appendix A.

Laplace approximation We can also obtain an approximation to the integral in (3) by Laplace's method. To achieve this goal, we expand $G(\mathbf{z})$ around $\tilde{\mathbf{z}} = H(\mathbf{x})$ to obtain a quadratic approximation to $v(\mathbf{x},\mathbf{z})$, which then leads to a multivariate Gaussian integral that is solvable in closed-form. The expansion of $G(\mathbf{z})$ can be represented as

$$\mathbf{x} - G(\mathbf{z}) = \mathbf{x} - G(\tilde{\mathbf{z}}) - \nabla G(\tilde{\mathbf{z}})(\mathbf{z} - \tilde{\mathbf{z}}) \tag{5}$$

where $\nabla G(\tilde{\mathbf{z}}) \in \mathbb{R}^{n\times m}$ is the Jacobian matrix at $\tilde{\mathbf{z}}$. Substitute (5) into $\|\mathbf{x} - G(\mathbf{z})\|_2^2$, we have

$$\begin{aligned}\|\mathbf{x} - G(\mathbf{z})\|_2^2 &= (\mathbf{x} - G(\mathbf{z}))^T(\mathbf{x} - G(\mathbf{z})) \\ &= \|\mathbf{x} - G(\tilde{\mathbf{z}})\|_2^2 - 2(\mathbf{x} - G(\tilde{\mathbf{z}}))^T\nabla G(\tilde{\mathbf{z}})(\mathbf{z} - \tilde{\mathbf{z}}) \\ &+ (\mathbf{z} - \tilde{\mathbf{z}})^T\nabla G^T(\tilde{\mathbf{z}})\nabla G(\tilde{\mathbf{z}})(\mathbf{z} - \tilde{\mathbf{z}})\end{aligned} \tag{6}$$

Next, We made variable substitutions as

$$\begin{cases}\mathbf{A} = \nabla G^T(\tilde{\mathbf{z}})\nabla G(\tilde{\mathbf{z}}) & \in \mathbb{R}^{m\times m} \\ \mathbf{b} = \nabla G^T(\tilde{\mathbf{z}})(\mathbf{x} - G(\tilde{\mathbf{z}})) & \in \mathbb{R}^m \\ \mathbf{w} = \mathbf{z} - \tilde{\mathbf{z}} & \in \mathbb{R}^m \\ \lambda = \sigma^{-2}\end{cases} \tag{7}$$

Taking equations (6) and (7) into $v(\mathbf{x},\mathbf{z})$ in equation (3) and we can finally get

$$\begin{aligned}v(\mathbf{x},\mathbf{z}) = \tilde{v}(\mathbf{x},\mathbf{w}) &= \|\mathbf{w}\|_2^2 + 2\mathbf{w}^T\tilde{\mathbf{z}} + \|\tilde{\mathbf{z}}\|_2^2 + \lambda(\|\mathbf{x} - G(\tilde{\mathbf{z}})\|_2^2 - 2\mathbf{b}^T\mathbf{w} + \mathbf{w}^T\mathbf{A}\mathbf{w}) \\ &= \mathbf{w}^T(\mathbf{I} + \lambda\mathbf{A})\mathbf{w} - 2(\lambda\mathbf{b} - \tilde{\mathbf{z}})^T\mathbf{w} + c_1(\mathbf{x})\end{aligned} \tag{8}$$

where $\mathbf{I} \in \mathbb{R}^{m \times m}$ is the identity matrix and $c_1(\mathbf{x}) = \|\tilde{\mathbf{z}}\|_2^2 + \lambda \|\mathbf{x} - G(\tilde{\mathbf{z}})\|_2^2$. The integral in (3) $w.r.t$ $\mathbf{z}$ can now be solved by constructing a multivariate Gaussian distribution $w.r.t$ $\mathbf{w}$ in (8) as the following

$$\begin{aligned}\int e^{-\frac{v(\mathbf{x},\mathbf{z})}{2}} d\mathbf{z} &= \int e^{-\frac{\tilde{v}(\mathbf{x},\mathbf{w})}{2}} d\mathbf{w} = \int e^{-\frac{(\mathbf{w}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu}) + c_2(\mathbf{x})}{2}} d\mathbf{w} \\ &= e^{-\frac{c_2(\mathbf{x})}{2}} \int e^{-\frac{(\mathbf{w}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})}{2}} d\mathbf{w} \\ &= e^{-\frac{c_2(\mathbf{x})}{2}} \sqrt{(2\pi)^m \det(\boldsymbol{\Sigma})}\end{aligned} \tag{9}$$

where $c_2(\mathbf{x}) = c_1(\mathbf{x}) - \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$, $\det(\boldsymbol{\Sigma})$ denotes the determinant of the covariant matrix. The constructed mean and covariant matrix of the multivariate Gaussian should be

$$\begin{cases} \boldsymbol{\Sigma} = (\mathbf{I} + \lambda \mathbf{A})^{-1} \\ \boldsymbol{\mu} = \boldsymbol{\Sigma}(\lambda \mathbf{b} - \tilde{\mathbf{z}}) \end{cases} \tag{10}$$

Substitute (9) into (3) and we can get the final closed-form solution for density of point $\mathbf{x}$

$$p(\mathbf{x}) = (\frac{1}{\sqrt{2\pi}})^n \sigma^{-n} \sqrt{\det(\Sigma)} e^{-\frac{c_2(\mathbf{x})}{2}} \tag{11}$$

There are two important properties related to the closed-form approximation which we list as follows.

- The constructed covariance matrix $\boldsymbol{\Sigma} = (\mathbf{I} + \lambda \mathbf{A})^{-1}$ is positive definite and all the eigenvalues are positive, which ensure the constructed multivariate Gaussian is valid. See *proof* in Appendix B1.
- The *change of variable rule* represented by (1) where $G(\cdot)$ is a differentiable and invertible function is a special case in our framework if $m = n$, $H(\cdot) = G^{-1}(\cdot)$ and taking a limit of $\sigma \to 0$. See *proof* in Appendix B2.

2.2 Adversarial training loss

The key idea of Roundtrip is to approximate the target distribution as a convolution of a Gaussian with a distribution induced on a manifold by transforming a base distribution where the transformation is learned by joint training of two GAN models (Figure 1). For the forward mapping function G and the discriminator D_x, G aims at generating samples $\{\tilde{\mathbf{z}}_i\}_{i=1}^N$ that are similar to observation data $\{\mathbf{x}_i\}_{i=1}^N$ while the discriminator D_x tries to discern observation data (positive) from generated samples (negative). The backward mapping function H and the discriminator D_z have the same training principle. The overall training process can be regarded as two min-max problems: $\min_G \max_{D_x} \mathcal{L}_{GAN}(G, D_x)$ and $\min_H \max_{D_z} \mathcal{L}_{GAN}(H, D_z)$ where

$$\begin{cases} \mathcal{L}_{GAN}(G, D_x) = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}(1 - D_x(G(\mathbf{z})))^2 + \mathbb{E}_{\mathbf{x} \sim Data(\mathbf{x})} D_x^2(\mathbf{x}) \\ \mathcal{L}_{GAN}(H, D_z) = \mathbb{E}_{\mathbf{x} \sim Data(\mathbf{x})}(1 - D_z(H(\mathbf{x})))^2 + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} D_z^2(\mathbf{z}) \end{cases} \tag{12}$$

Note that the least square loss we used in equation (12) was detailedly discussed in LSGAN [18].

2.3 Roundtrip loss

During the training, we also aim to minimize the roundtrip loss which is defined as $\rho(\mathbf{z}, H(G(\mathbf{z})))$ and $\rho(\mathbf{x}, G(H(\mathbf{x})))$. The principle is to minimize the distance when a data point goes through a roundtrip transformation between two data domains. e.g., $\mathbf{x} \to H(\mathbf{x}) \to G(H(\mathbf{x})) \approx \mathbf{x}$ which ensures that the observation data point will stay close to the learned manifold after a roundtrip data transformation. In practice, we used l_2 loss for both $\rho(\mathbf{z}, H(G(\mathbf{z})))$ and $\rho(\mathbf{x}, G(H(\mathbf{x})))$ as minimizing l_2 loss implies the data is drawn from a Gaussian distribution [19] which exactly matches our model assumption. We denoted roundtrip loss as

$$\mathcal{L}_{RT}(G, H) = \alpha \|\mathbf{x} - G(H(\mathbf{x}))\|_2^2 + \beta \|\mathbf{z} - H(G(\mathbf{z}))\|_2^2 \tag{13}$$

where α and β are two constant coefficients. The idea of roundtrip loss which exploits transitivity for regularizing structured data can also be found in previous works [33, 32].

2.4 Full training loss

Combining the adversarial training loss and Roundtrip loss together, we can get the full training loss as $\mathcal{L}(G, D_x, H, D_z) = \mathcal{L}_{GAN}(G, D_x) + \mathcal{L}_{GAN}(H, D_z) + \mathcal{L}_{RT}(G, H)$. To achieve joint training of two of GAN models. We iteratively updated the parameters in two generative models and two discriminative models, respectively. Thus, the roundtrip transformation used in our density estimator can be represented as

$$G^*, D_x^*, H^*, D_z^* = arg \min_{G,H} \max_{D_x,D_z} \mathcal{L}(G, D_x, H, D_z) \tag{14}$$

2.5 Model architecture

The model architecture for Roundtrip model is highly flexible. In most cases, when it is utilized for density estimation tasks with vector-valued data, we used fully-connected networks for both generative networks and discriminative networks. Specifically, the G network contains 10 fully-connected layers and each layer has 512 hidden nodes while the H network contains 10 fully-connected layers and each layer has 256 hidden nodes. The D_x networks contains 4 fully-connected layers and each layer has 256 hidden nodes while the H network contains 2 fully-connected layers and each layer has 128 hidden nodes. The leaky-Relu activation function is deployed after each layer propagation.

We also extended Roundtrip for estimating the density of tensor-valued data (e.g., images) by introducing a one-hot encoded class label $\mathbf{y}$ as an additional input to both G and D_x networks in a conditional GAN manner [20]. $\mathbf{y}$ will be combined in the hidden representations in G and D_x networks by concatenation. Compared to vector-valued data, tensor-valued data such as images will be flattened and reshaped when taken as input and output to all networks in Roundtrip, respectively. Similar to the model architecture in DCGAN [24], we used transposed convolutional layers for upsampling images from latent space for G network. Besides, we used traditional convolutional neural networks for H, D_x while D_z still adopts a fully-connected network architecture. Note that Batch normalization [8] is applied after each convolutional layer or transposed convolutional layer.

3 Results

3.1 Experiement setup

We test the performance of Roundtrip model in a series of experiments, including simulation studies and real data studies. In these experiments, we compared Roundtrip to the widely used Gaussian kernel density estimator as well as several neural density estimators, including MADE [6], Real NVP [4] and MAF [21]. In the outlier detection experiment, we additionally compared to two commonly used outlier detection methods: One-class SVM [27] and Isolation Forest [14]. Note that the default setting of Roundtrip model was based on the importance sampling strategy. Results of Roundtrip density estimator based on Laplace approximation are reported in Appendix C.

The neural networks in Roundtrip model were implemented with TensorFlow [1][1] In all experiments, we set α=10 and β=10 in equation (13). For the parameter σ in our model assumption, we selected from $\{0.01, 0.05, 0.1, 0.2, 0.4, 0.5\}$ of which the value maximizes the average likelihood on validation test. Sample size N in importance sampling is set to 40,000. An adam optimizer with a learning rate of 0.0002 was used for backpropagation and updating model parameters. We took Gaussian kernel density estimator (KDE) as a baseline where the bandwidth is selected by Silverman's "rule of thumb" [30] or Scott's rule [29]. We choose the one with better results to present. The three alternative neural density estimators (MADE, Real NVP, and MAF) were implemented through `https://github.com/gpapamak/maf`. In outlier detection tasks, we implemented One-class SVM and Isolation Forest using scikit-learn library [23], where the default parameters were used. To ensure fair model comparison, both simulation and real data were randomly split into 90% traning set and 10% test set. For neural density estimators including Roundtrip, 10% of the training set were kept as a validation set. The image datasets with training and test set were directly provided which require no further data split.

[1]The reproducible code of Roundtrip can be found at `https://github.com/kimmo1019/Roundtrip`.

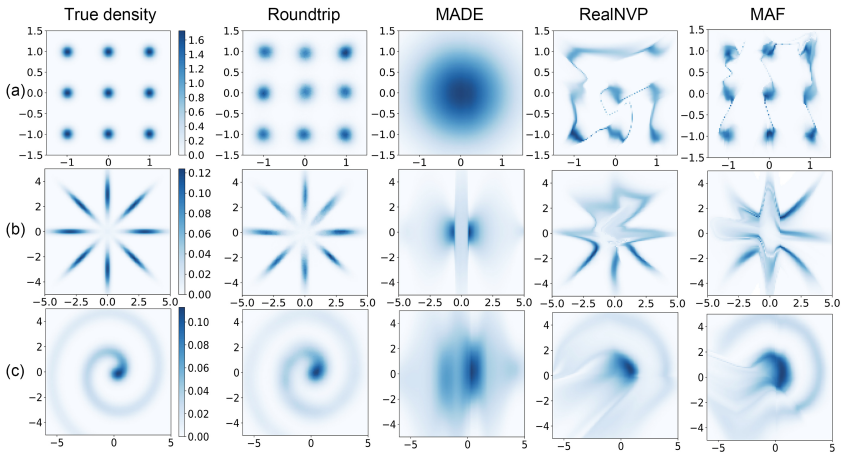

Figure 2: True density and estimated density by different neural density estimators with three simulation datasets. Density plots were shown on a 100×100 grid 2D bounded region.

3.2 Evaluation

For simulation datasets where the true density can be calculated, we evaluate different density estimators by calculating the Spearman (rank) correlation between true density and estimated density. For real data where the ground truth is not available, the average estimated density (natural log-likelihood) on the test set will be considered as a measurement. In the application of outlier detection, we measure performance by calculating the precision at k, which is defined as the proportion of correct results in the top k ranks. We set k to the number of outliers in the test set.

3.3 Simulation studies

We first designed three 2D simulation datasets to test the performance of different neural density estimators where the truth density can be calculated.

a) *Independent Gaussian mixture*. $x_i \sim \frac{1}{3}(N(-1, 0.5^2) + N(0, 0.5^2) + N(1, 0.5^2))$, i=1,2.

b) *8-octagon Gaussian mixture*. $\mathbf{x} \sim \frac{1}{8}\sum_{i=1}^{8} N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ where $\boldsymbol{\mu}_i = (3\cos\frac{\pi i}{4}, 3\sin\frac{\pi i}{4})$ and $\boldsymbol{\Sigma}_i = \begin{pmatrix} \cos^2\frac{\pi i}{4}+0.16^2\sin^2\frac{\pi i}{4} & (1-0.16^2)\sin\frac{\pi i}{4}\cos\frac{\pi i}{4} \\ (1-0.16^2)\sin\frac{\pi i}{4}\cos\frac{\pi i}{4} & \sin^2\frac{\pi i}{4}+0.16^2\cos^2\frac{\pi i}{4} \end{pmatrix}$, i=1,...,8.

c) *Involute*. $x_1 \sim N(r\sin(2r), 0.4^2)$, $x_2 \sim N(r\cos(2r), 0.4^2)$ where $r \sim U(0, 2\pi)$

20000 $i.i.d$ points were sampled from the true data distribution. After model training, we directly estimated the density in a 2D bounded region (100×100 grid) with different methods (Figure 2). For the independent Gaussian mixture in case (a), Roundtrip clearly separates the independent components in the Gaussian mixture while other neural density estimators either failed (MADE) or contain obvious trajectory between different components (Real NVP and MAF). Roundtrip can capture a better density distribution even for the highly non-linear structure in case (c). Then we took the case (a) for a further studies by increasing the dimension up to 10 (containing 3^{10} modes). The performance of kernel density estimator (KDE) will decrease dramatically when dimension increases. Roundtrip still achieves a Spearman correlation of 0.829 at dimension 10, compared to 0.669 of Real NVP, 0.595 of MAF and 0.14 of KDE (See Appendix C).

3.4 Real data studies

UCI datasets We collected five datasets (AReM, CASP, HEPMASS, BANK and YPMSD) from the UCI machine learning repository [5] with dimensions ranging from 6 to 90 and sample size from 42,240 to 515,345 (see more details about data description and data preprocessing in Appendix D). Unlike simulation data, these real datasets have no ground truth for the density. Hence, we evaluated different methods by calculating the average log-likelihood on the test set. Table 1 illustrates the performance of Roundtrip and other neural density estimators. A Gaussian kernel density estimator

Table 1: Performance of different methods on five UCI datasets. The average log likelihood (.nat) and 2 standard deviations are shown. The model with best performance is shown in bold.

	AReM	CASP	HEPMASS	BANK	YPMSD
KDE	6.26±0.07	20.47±0.10	-25.46±0.03	15.84±0.12	247.03±0.61
MADE	6.00±0.11	21.82±0.23	-15.15±0.02	14.97±0.53	273.20±0.35
Real NVP	9.52±0.18	26.81±0.15	-18.71±0.02	26.33±0.22	287.74±0.34
MAF	9.49±0.17	27.61±0.13	-17.39±0.02	20.09±0.20	290.76±0.33
Roundtrip	**11.74±0.04**	**28.38±0.08**	**-4.18±0.02**	**35.16±0.14**	**297.98±0.52**

Table 2: The precision at k of different methods in three ODDS datasets.

	OC-SVM	I-Forest	Real NVP	MAF	Roundtrip
Shuttle	0.953	**0.973**	0.784	0.929	**0.973**
Mammography	0.370	**0.482**	**0.482**	0.407	**0.482**
ForestCover	0.127	0.058	0.054	0.046	**0.177**

(KDE) fitted to the training data is reported as a baseline. Roundtrip outperforms other neural density estimators on every dataset, which again demonstrates the superiority of our model.

Image datasets We further applied Roundtrip model to generate images and assess the quality of the generated images by estimated density. Deep generative models have demonstrated their power in generating synthetic images. However, a deep generative model alone cannot provide quality scores for generated images. Here, we propose to use our Roundtrip method to generate images and quality score (e.g., the density of the image). We test this approach on two commonly used image datasets MNIST [13] and CIFAR-10 [12] where in each of the these datasets, the image comes from 10 distinct classes. Roundtrip model was modified by introducing an additional one-hot encoded class label $\mathbf{y}$ to both G and D_x network and convolutional layers were used in G, H and D_x (see Methods). We then model the conditional density estimation by $p(\mathbf{x}|\mathbf{y}) = \int p(\mathbf{x}|\mathbf{y}, \mathbf{z})p(\mathbf{z})d\mathbf{z}$ where $\mathbf{y} \sim$Cat(10) denoting a categorical distribution with 10 distinct classes. We use this modified Roundtrip model to simultaneously generage images conditional on a class label and compute the within class density of the image. The comparing neural density estimators typically require a lot of tricks, including rescaling pixel values to $[0, 1]$, transforming the bounded pixel values into a unbounded logit space and adding uniform noise, to achieve images generation and density estimation. Roundtrp did not require additional transformation except for rescaling. In Figure 3, the generated images of each class were sorted by decreased likelihood. It is seen that images generated by Roundtrip are more realistic than those generated by MAF (which is the best among alternative neural density estimators, see Figure 2 and Table 1). Furthermore, the density provided by Roundtrip seems to correlate well with the quality of the generated iamges.

3.5 Outlier detection

Finally, we applied Roundtrip model to an outlier detection task, where a data point with extremely low density value is regarded as likely to be an outlier. We tested this method on three outlier detection datasets (Shuttle, Mammography, and ForestCover) from ODDS database (`http://odds.cs.stonybrook.edu/`). Each dataset is split into training, validation and test set (details of data description can be found in Appendix D). Besides the neural density estimators, we also introduced two baselines One-class SVM [27] and Isolation Forest [14]. The results were shown in Table 2. Roundtrip achieves the best or comparable results in different outlier detection tasks. Especially in the last dataset ForestCover, in which the outlier percetage is only 0.9%, Roundtrip still achieves a precision of 17.7% while the precision of other neural density estimators is less than 6%.

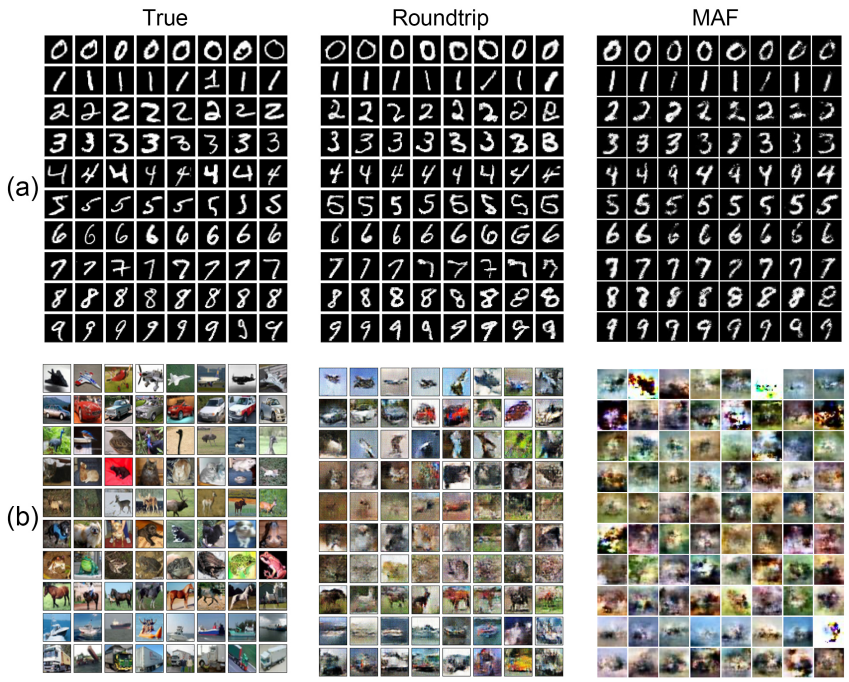

Figure 3: (a) True and generated images of MNIST. (b) True and generated images of CIFAR-10. Note that images generated by Roundtrip and MAF were sorted by decreased likelihood for each class. The results of MAF were directly collected from its original paper.

4 Discussion

We proposed Roundtrip as a novel neural density estimator based on deep generative models. Unlike prior studies modeling the invertible transformation from a base density, of which the parameters are learned by neural networks, Roundtrip directly learns the joint distribution of data based on deep generative models. Roundtrip outperforms previous neural density estimators in a variety of density estimation tasks, including simulation/real data studies and an outlier detection application. We also demonstrated the high flexibility in Roundtrip as it can be either used for estimating density in vector-valued data and tensor-values data (e.g., images).

Acknowledgements

This work is supported by NSF grants DMS1721550, DMS1811920, National Key Research and Development Program of China No. 2018YFC0910404, the National Natural Science Foundation of China Nos. 61873141, 61721003, 61573207, and the Tsinghua-Fuzhou Institute for Data Technology.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. Density modeling of images using a generalized normalization transformation. *arXiv preprint arXiv:1511.06281*, 2015.

[3] Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

[4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[5] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[6] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[9] Mahdi Karami, Laurent Dinh, Daniel Duckworth, Jascha Sohl-Dickstein, and Dale Schuurmans. Generative convolutional flow for density estimation. In *Workshop on Bayesian Deep Learning NeurIPS 2018*, 2018.

[10] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[11] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[12] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[13] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[14] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[15] Luo Lu, Hui Jiang, and Wing H Wong. Multivariate density estimation by bayesian sequential partitioning. *Journal of the American Statistical Association*, 108(504):1402–1410, 2013.

[16] Gábor Lugosi, Andrew Nobel, et al. Consistency of data-driven histogram methods for density estimation and classification. *The Annals of Statistics*, 24(2):687–706, 1996.

[17] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[18] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[19] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[20] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[21] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[22] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[25] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[26] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, pages 832–837, 1956.

[27] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[28] David W Scott. On optimal and data-based histograms. *Biometrika*, 66(3):605–610, 1979.

[29] David W Scott. Multivariate density estimation: theory, practice, and visualization. 1992.

[30] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[31] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

[32] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857, 2017.

[33] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.