Supplementary Materials for the Submission: Topology-Matching Normalizing Flows for Out-of-Distribution Detection in Robot Learning

Anonymous Author(s)
Affiliation
Address
email

1 Background

1.1 Resampled Base Distribution

Learned Accept/Reject Sampling (LARS) approximates a d-dimensional distribution $q(z)$ by reweighting a proposal distribution. The reweightings are learned through a parametrized acceptance function. The parameter is ϕ, which determines the shape of the acceptance function. If a sample of π is given, it is accepted with a certain probability, otherwise, it is rejected, and a new sample is drawn until one of the proposed samples is accepted. This means:

$$p_\infty(z) = \frac{\pi(z)a_\phi(z)}{Z} \quad \text{with} \quad Z := \int_\pi (z)a_\phi(z)dz. \tag{1}$$

One can also reduce the reject rates by a truncation parameter. The trick here is to accept the T-th sample, if the first T-1 samples are rejected. This is independent of the value provided by LARS. With this, the sampling distribution becomes, given $\alpha_T := (1 - Z)^{T-1}$:

$$p_T(z) = (1 - \alpha_T)\frac{a_\phi(z)\pi(z)}{Z} + \alpha_T\pi(z) \quad \text{where} \quad Z \approx \frac{1}{S}\sum_{s=1}^{S} a_\phi(z_s). \tag{2}$$

We note that reducing rejection rates are desirable in order to reduce the computational overhead. Z is often approximated due to intractability. As parameters of the acceptance function cause changes in Z, z_s needs to be recomputed in every training iteration.

The resample base distributions (RSB) relies on LARS. LARS can be used as a proposal to avoid topological mismatches of the flows. Then, density could become the aforementioned sampling distribution. The log-likelihood using such distributions can be derived as:

$$\log p(x) = \log\pi(z) + \log\left(\alpha_T + (1 - \alpha_T\frac{a_\phi(z)}{Z})\right) - \log|\det J_{F_\theta}(z)|. \tag{3}$$

F_θ is the flow transformation. The intuition is that by learning the parameters of the base distribution while keeping it computationally feasible with LARS, the base distribution is modified so that the topology mismatch problem can be addressed [1].

1.2 Information Bottleneck for Normalizing Flows

The Information Bottleneck (IB) for Normalizing Flows (NFs) with class conditional base distribution, and its derivation, closely follows Ardizzone et al. [2]. They originally derived this quantity for the class-conditional Gaussian Mixture Models (GMMs).

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

Generally, the IB [3] is defined as

$$\mathcal{L}_{\text{IB}} = I(U, Z) - \beta I(Z, Y) \tag{4}$$

with Mutual Information (MI) I and trade-off parameter β. To derive the learning objective presented in the main part two key steps are required: (1) we substitute MI I with the so-called *mutual cross-information* $\hat{I}$, and (2) we inject Gaussian noise in the otherwise lossless transformation between U and Z.

The *mutual cross-information* $\hat{I}$ [2] is defined as

$$\hat{I}(A, B) = \mathbb{E}_{a,b\sim p(A,B)} \left[\log \frac{q(a, b)}{q(a)q(b)} \right] \tag{5}$$

where q represents the approximative distribution of the true density p. We then replace both occurrences of I in Equation (4) with $\hat{I}$. Moreover, for simplicity, we assume that $q(Y) = p(Y)$, and in our experiments, we model $p(Y)$ uniformly.

The bijection T_ϕ is loss-less by design and, thus, the joint distributions $p(U, Z)$ and $p(U, Z)$ are not valid Radon-Nikodym densities rendering I and $\hat{I}$ undefined. To circumvent the issue and as indicated above (2), we artificially introduce noise $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ and model $Z_\epsilon = T_\phi^{-1}(U + \epsilon)$ instead. Then, dropping all terms that are independent of the model or vanish for $\sigma \to 0$, we obtain (see Ardizzone et al. [2] for full details)

$$\hat{I}(U, Z_\epsilon) = \mathbb{E}_{p(\mathbf{u}),p(\epsilon)} \left[-\log \sum_{y\prime} (p_\psi(\mathbf{z}|y\prime)) - \log |\det(J_{T_\phi^{-1}}(\mathbf{u} + \epsilon))| \right], \tag{6}$$

Similarly, $\hat{I}(Y, Z_\epsilon)$ resolves to

$$\begin{aligned} \hat{I}(Y, Z_\epsilon) &= \underbrace{\mathbb{E}_{p(y)} \left[-\log p(y) \right]}_{=\text{const.}} + \mathbb{E}_{p(Z,Y),p(\epsilon)} \left[\log \frac{p_\psi(T_\phi^{-1}(\mathbf{u} + \epsilon)|y)p(y)}{\sum_{y\prime}(p_\psi(T_\phi^{-1}(\mathbf{u} + \epsilon)|y\prime)p(y\prime))} \right] \\ &= \mathbb{E}_{p(Z,Y),p(\epsilon)} \left[\log \frac{p_\psi(\mathbf{z}|y)p(y)}{\sum_{y\prime}(p_\psi(\mathbf{z}|y\prime)p(y\prime))} \right] + \text{const.} \end{aligned} \tag{7}$$

with $\mathbf{z} = T_\phi^{-1}(\mathbf{u} + \epsilon)$. The overall objective combines Equation (6) and Equation (7) to

$$\mathcal{L}_{\text{IBNF}} = \hat{I}(U, Z) - \beta \hat{I}(Z, Y) \tag{8}$$

In summary, with $\hat{I}(U, Z)$ we minimize the marginalized density $p(\mathbf{u})$ subject to the noise term ϵ. On the other hand, $\mathcal{L}_{\text{IBNF}}$ optimizes for predictive performance, again subject to noise ϵ.

2 Implementation Details

2.1 Synthetic Density Estimation

We follow a resource constraint setup to allow for conclusions about practical mobile applications. Specifically, we use a Real NVP [4] with four layers for the cRSB and five layers for the MoG with randomly initialized trainable mean and variance. The difference in layers accounts for the extra compute that is required for the cRSB. For the NSF we use two layers each since the cRSB overhead is negligible here. In the MoG's we model a class-conditional Gaussian $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c\mathbf{I})$ with diagonal variance matrix.

2.2 Out-of-Distribution (OOD) detection for 2D Object Detection

For object detection, we implement our method with the Glow [5] architecture. Besides, we also ablate on this with different ones such as Neural Spline Flows (NSF) [6]. On object detection task, we train our flows based on logits generated by the pre-trained Faster-RCNN [7] object detectors

Table 1: Implementation Details for density estimation.

hyper-params	cRSB	MoG
IB loss	$\beta(0.1); \sigma(0.01)$	$\beta(0.1); \sigma(0.01)$
learning rate	$1e-3$	$1e-3$
training epoch	$10,000$	$25,000$
optimizer	Adamax	Adamax
batch_size	1024	1024

provided by Miller et al. [8] for fair comparison [1]. We summarize the training details for the results presented in the main paper in Table 2 and Table 3. Regarding the trainability for the base distribution in Mixture of Gaussians (MoG), we found minor difference between setting it trainable and untrainable and the major impacting factor for performance improvement and training stability is the distance between means for initialization. Therefore we seek to tune this hyper-parameter and leave the base distribution untrainable.

Table 2: Implementation Details for Glow on Pascal-VOC-OS.

hyper-params	cRSB_IB	GMM_IB	cRSB_CLS	GMM_CLS	RSB	GMM	Gaussian
base	Dropout(0.1); $T(100)$; $\epsilon(0.05)$; $a(\cdot): 3\times128$;	distance-scale for means initialization:10	Dropout(0.1); $T(100)$; $\epsilon(0.05)$; $a(\cdot): 3\times128$;	distance-scale for means initialization:10	Dropout(0.1); $T(100)$; $\epsilon(0.01)$; $a(\cdot): 3\times128$;	distance-scale for means initialization:10	—
flow arch	$16\times[4\times64]$	$16\times[4\times64]$	$16\times[4\times64]$	$16\times[4\times64]$	$16\times[4\times64]$	$16\times[4\times64]$	$16\times[4\times64]$
IB loss	$\beta(30.0)$; $\sigma(0.5)$	$\beta(50.0)$; $\sigma(0.5)$	—	—	—	—	—
learning rate	$1e-4$	$1e-4$	$1e-4$	$1e-4$	$1e-4$	$1e-4$	$1e-4$
training epoch	400	400	200	400	200	400	400
optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam
batch_size	1024	1024	1024	1024	1024	1024	1024

Table 3: Implementation Details for Glow on MS-COCO-OS.

hyper-params	cRSB_IB	GMM_IB	cRSB_CLS	GMM_CLS	RSB	GMM	Gaussian
base	Dropout(0.1); $T(100)$; $\epsilon(0.01)$; $a(\cdot): 3\times128$;	distance-scale for means initialization:10	Dropout(0.1); $T(100)$; $\epsilon(0.05)$; $a(\cdot): 3\times128$;	distance-scale for means initialization:10	Dropout(0.1); $T(100)$; $\epsilon(0.01)$; $a(\cdot): 3\times128$;	distance-scale for means initialization:10	—
flow arch	$8\times[4\times128]$	$8\times[4\times128]$	$8\times[4\times128]$	$8\times[4\times128]$	$8\times[4\times128]$	$8\times[4\times128]$	$8\times[4\times128]$
IB loss	$\beta(30.0)$; $\sigma(0.1)$	$\beta(50.0)$; $\sigma(0.5)$	—	—	—	—	—
learning rate	$1e-4$	$1e-4$	$1e-4$	$1e-4$	$1e-4$	$1e-4$	$1e-4$
training epoch	200	200	200	200	200	200	200
optimizer	Adamax	Adamax	Adamax	Adamax	Adamax	Adamax	Adamax
batch_size	512	1024	1024	1024	1024	1024	1024

2.3 Real Robot Deployment

We used the open-sourced implementation [2] for training and testing the object detector yolov7 [9] based on datasets described in Section 3.3. Specifically, we trained the architecture of yolov7-e6e with a learning rate $1e^{-2}$, weight decay $5e^{-4}$, batch size 2, image size 720×720 and SGD optimizer for 50 epochs. The detector was then deployed on the embedded computing module NVIDIA Jetson Orin on our aerial manipulation robot.

An aerial manipulation system is composed of a mobile platform, capable of moving in the 3D world. Carrying a robotic manipulator, such systems extend the mobility of robotic manipulators. Several applications are envisioned. Amongst them, in this paper, we ground our method in applications of robotic inspection and maintenance. The perception system has to understand its surroundings semantically, and here, deep learning-based methods are the current golden standards. Unfortunately, learning-based methods often assume that the test samples are generated from the same distribution as the training data. This assumption is routinely violated in the real world, and out-of-distribution detectors aim to identify such failure cases of learning-based methods.

[1] https://github.com/dimitymiller/openset_detection

[2] https://github.com/WongKinYiu/yolov7

For the implementation details, our in-house developed robot consists of one stereo camera, one monocular camera, one RBG-D camera, and a LiDAR sensor. In this work, since the semantics of the scenes may rely on vision as its main modalities, we utilize one monocular camera to detect the objects of interest, which are an industrial valve, and an inspection robotic crawler for oil and gas pipes in refineries. For computing, the robot is equipped with two NVIDIA Jetson Orin. In the experiments adapted, the real images were captured in a mock-up facility, and tested with the NVIDIA Jetson ORIN on the robot. The experimental data were collected with 30W mode with JETPACK 5.1.1. Auvidea carrier board is used.

3 Datasets

In this section, we provide more details on the datasets used in the experiments.

3.1 2-D Density Estimation

In each training epoch, we sample a new batch according to the subsequent unnormalized log densities following Stimper et al. [1]. For **Two Moons**, we use

$$-\frac{(\|\mathbf{u}\|-1)^2}{0.08}-\frac{\left(|u_0|-2\right)^2}{0.18}+\log\left(1+\mathrm{e}^{-\frac{4u_0}{0.09}}\right) \tag{9}$$

and define the class assignments in $y \in \{0,1\}$ via $p(y|\mathbf{u}) = u_1 > 0$. In words, the class is 1 for positive y-values and 0 otherwise. The **Two Rings** distribution is defined as

$$\log\left[\sum_{i=1}^{2}\left(\frac{32}{\pi}\exp\left\{-32(\|\mathbf{u}\|-i-1)^2\right\}\right)\right] \tag{10}$$

where $p(y|\mathbf{u}) = u_0 > 0$. Here we split classes along the y-axis. Last, the **Circle of Gaussians** is given by

$$\log\left[\sum_{i=1}^{8}\left(\frac{9}{2\pi(2-\sqrt{2})}\exp\left\{-\frac{9\left(\left(u_0-2\sin\left(\frac{2\pi}{8}i\right)\right)^2+\left(u_1-2\cos\left(\frac{2\pi}{8}i\right)\right)^2\right)}{4-2\sqrt{2}}\right\}\right)\right] \tag{11}$$

where we assign each Gaussian plot in an alternating scheme to the classes $y \in \{0,1\}$. Concretely, we split the classes according to $\sin(4\cdot(\mathrm{atan2}(u_1,u_0)+\frac{\pi}{8}))>0$. For this, we use the $\mathrm{atan2}(u_1,u_0)$ function that is available in many programming environments. We follow the IEEE convention for value combinations like $u_0 = u_1 = 0$.

3.2 OOD detection for 2D Object Detection

Following the experimental protocol used in [8], we first construct the open-set object detection data set as they did for Pascal-VOC [10] and MS-COCO [11], dubbed as Pascal-VOC-OS and MS-COCO-OS . The idea behind is to exclude the effects of the unknown objects in the backgrounds for object detector training, since the object detector is inherently trained to ignore the background objects seen during training. We summarize the detailed information regarding the training, validation and test set of the adapted object detection datasets in Section 3.2. It contains the sizes of correct and false predictions from the detector, i.e. True Positives (TP) and False Positives (FP). The FP here are from the OOD data which should not be detected by the detector.

For convenience in training NFs, we further extract the features (logits in our case) from all the detections predicted from the object detector. We then filter out the detections with low quality and keep the meaningful detections by setting thresholds for the Intersection-Over-Union (0.5) and confidence score (0.2). The purpose behind is to simplify the problem and focus on only meaningful False Positive (FP) predictions predicted by the detector.

Table 4: Information of Pascal-VOC-OS and MS-COCO-OS

	Pascal-VOC-OS	MS-COCO-OS
ID train set	first 15 classes from Pascal-Voc2007&2012 train	first 50 classes from MS-COCO2017 train
ID val set	first 15 classes from Pascal-Voc2007&2012 val	first 50 classes from MS-COCO2017 val
Test set	20 classes from Pascal-Voc2007 test	80 classes from MS-COCO2017 test
#TP in training set	18318	251202
#TP in val set	7601	55593
#ID FP in val set	348	2287
#TP in test set	8288	16148
#ID FP in test set	213	784
#OOD FP in test set	660	1068

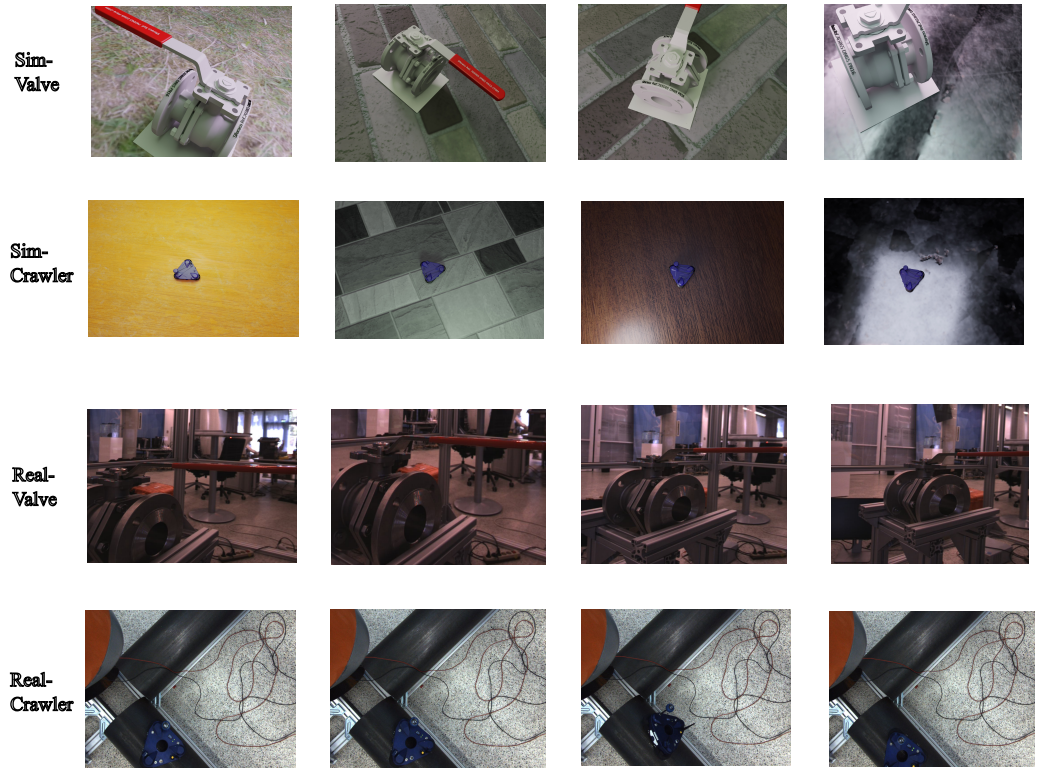


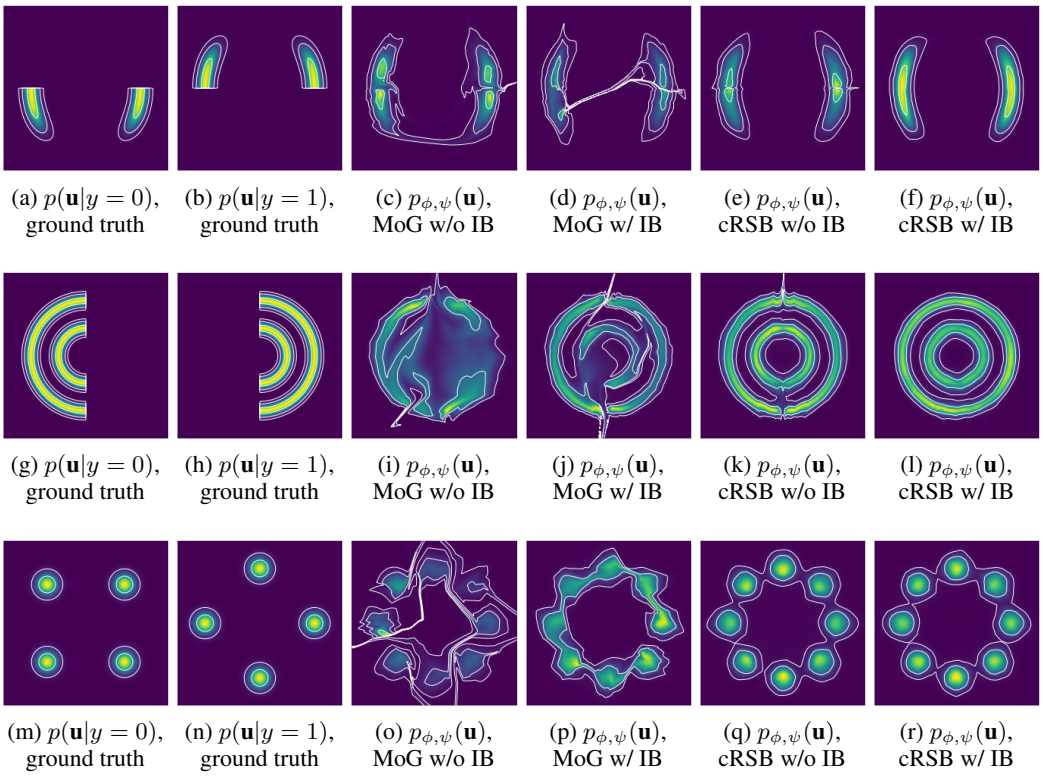
Figure 1: Exemplar images from training and test set used in the real robot experiment.

3.3 OOD detection on Real Robot

The synthetic images used for training in this part were generated by BlenderProc [12]. The test set contains real images collected in our lab. Some examplar images are shown in Figure 1. We summarize the sizes of the used training and test data in Section 3.3.

Table 5: Training and test data for real robot experiments

#images in Sim train set	907
#images in Sim test set	855
#images in Real test set	2078
#ID True Positives in test set	1326
#OOD False Positives in test set	1898



(a) $p(\mathbf{u}|y=0)$, ground truth (b) $p(\mathbf{u}|y=1)$, ground truth (c) $p_{\phi,\psi}(\mathbf{u})$, MoG w/o IB (d) $p_{\phi,\psi}(\mathbf{u})$, MoG w/ IB (e) $p_{\phi,\psi}(\mathbf{u})$, cRSB w/o IB (f) $p_{\phi,\psi}(\mathbf{u})$, cRSB w/ IB

(g) $p(\mathbf{u}|y=0)$, ground truth (h) $p(\mathbf{u}|y=1)$, ground truth (i) $p_{\phi,\psi}(\mathbf{u})$, MoG w/o IB (j) $p_{\phi,\psi}(\mathbf{u})$, MoG w/ IB (k) $p_{\phi,\psi}(\mathbf{u})$, cRSB w/o IB (l) $p_{\phi,\psi}(\mathbf{u})$, cRSB w/ IB

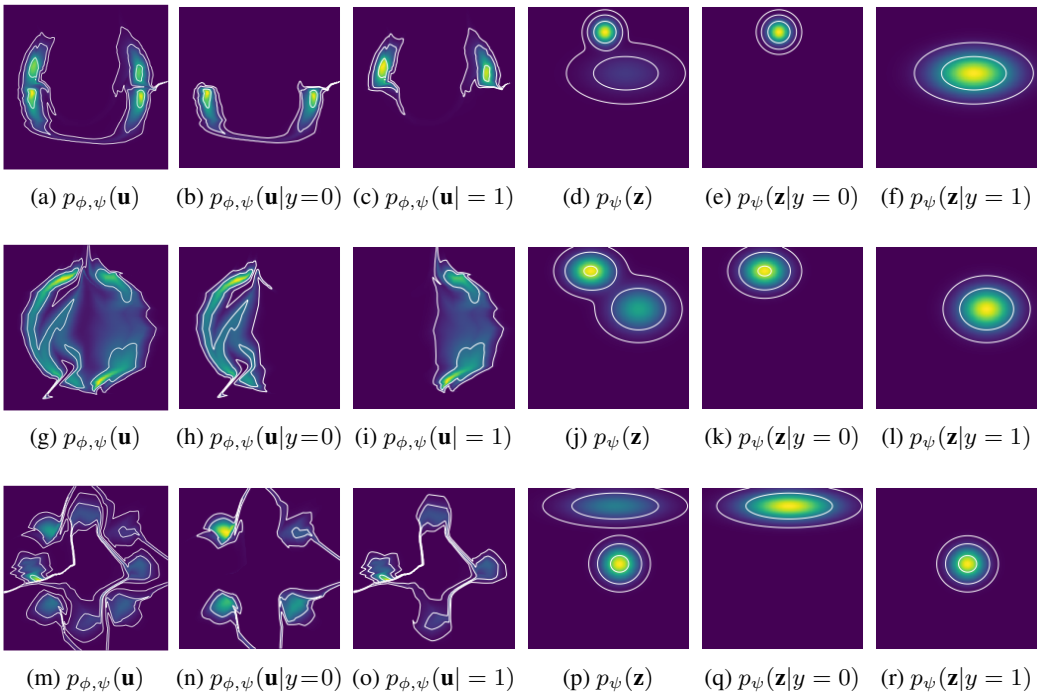
(m) $p(\mathbf{u}|y=0)$, ground truth (n) $p(\mathbf{u}|y=1)$, ground truth (o) $p_{\phi,\psi}(\mathbf{u})$, MoG w/o IB (p) $p_{\phi,\psi}(\mathbf{u})$, MoG w/ IB (q) $p_{\phi,\psi}(\mathbf{u})$, cRSB w/o IB (r) $p_{\phi,\psi}(\mathbf{u})$, cRSB w/ IB

Figure 2: Comparison of generative modeling capabilities $p_{\phi,\psi}(\mathbf{u})$ including training w/o and w/ IB.

4 Density Estimation

In Figure 2, we provide a comparison of the (marginal) density estimation capabilities for the three density estimation datasets of NFs with MoG as well as Conditional Resampled Base Distributions (cRSB) base distributions, trained w/o as well as w/ IB. In Figure 3-6, we illustrate the marginal and class-conditional feature as well as base distributions.

The main conclusions are (1) training with IB objective results in better modeling capabilities of the marginal distribution $p_{\phi,\psi}(\mathbf{u})$; (2) training w/o IB objective not only results in more filaments between the modes but also causes spiky tendrils at areas where the class conditional distributions are touching; (3) results with our cRSB base distributions are essentially filament free; (4) training with IB allows for slight overlaps in the class conditional densities $p_{\phi,\psi}(\mathbf{u}|y)$ as well as $p_{\phi,\psi}(\mathbf{z}|y)$. In other words, here we improve the modeling capabilities of the marginal distribution $p_{\phi,\psi}(\mathbf{u})$ at the cost of imperfect modeling of the class-conditional structure.

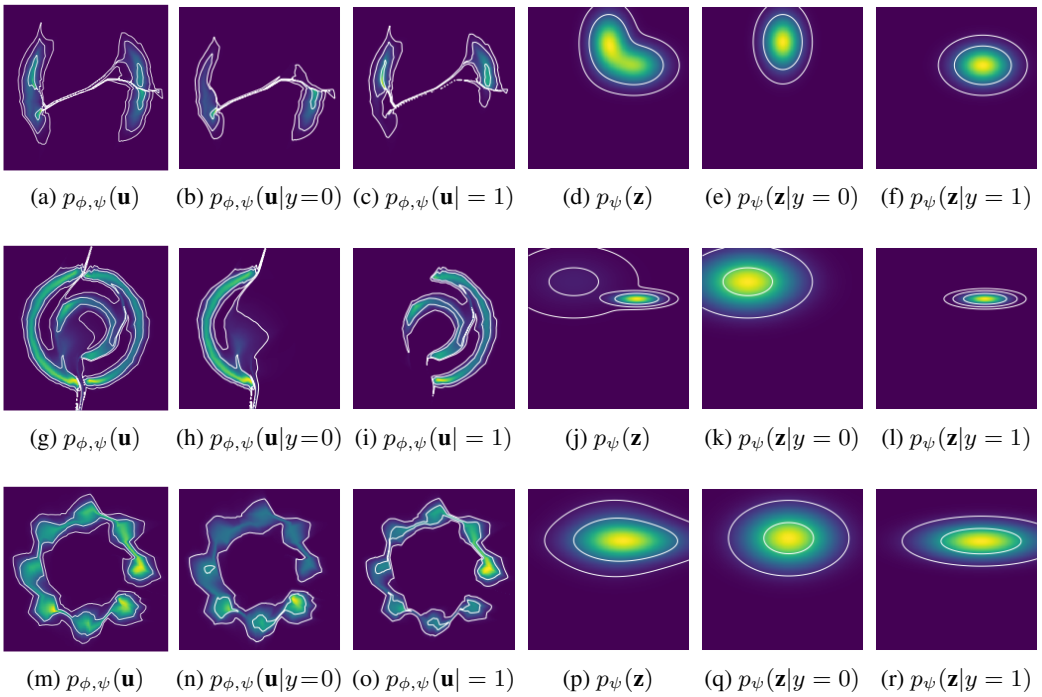


(a) $p_{\phi,\psi}(\mathbf{u})$ (b) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (c) $p_{\phi,\psi}(\mathbf{u}|=1)$ (d) $p_\psi(\mathbf{z})$ (e) $p_\psi(\mathbf{z}|y=0)$ (f) $p_\psi(\mathbf{z}|y=1)$

(g) $p_{\phi,\psi}(\mathbf{u})$ (h) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (i) $p_{\phi,\psi}(\mathbf{u}|=1)$ (j) $p_\psi(\mathbf{z})$ (k) $p_\psi(\mathbf{z}|y=0)$ (l) $p_\psi(\mathbf{z}|y=1)$

(m) $p_{\phi,\psi}(\mathbf{u})$ (n) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (o) $p_{\phi,\psi}(\mathbf{u}|=1)$ (p) $p_\psi(\mathbf{z})$ (q) $p_\psi(\mathbf{z}|y=0)$ (r) $p_\psi(\mathbf{z}|y=1)$

Figure 3: Marginalized and class-conditional feature as well as base distributions for MoG trained w/o IB.

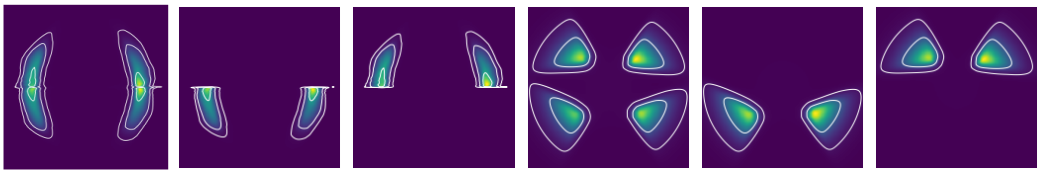


(a) $p_{\phi,\psi}(\mathbf{u})$ (b) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (c) $p_{\phi,\psi}(\mathbf{u}|=1)$ (d) $p_\psi(\mathbf{z})$ (e) $p_\psi(\mathbf{z}|y=0)$ (f) $p_\psi(\mathbf{z}|y=1)$

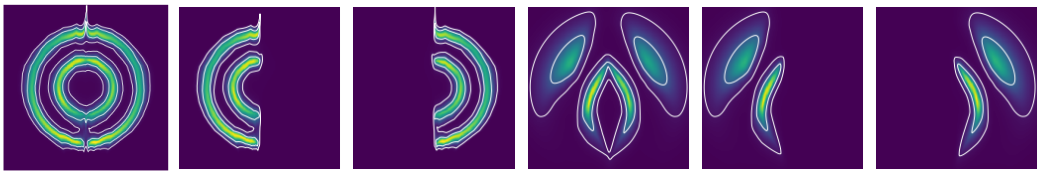
(g) $p_{\phi,\psi}(\mathbf{u})$ (h) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (i) $p_{\phi,\psi}(\mathbf{u}|=1)$ (j) $p_\psi(\mathbf{z})$ (k) $p_\psi(\mathbf{z}|y=0)$ (l) $p_\psi(\mathbf{z}|y=1)$

(m) $p_{\phi,\psi}(\mathbf{u})$ (n) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (o) $p_{\phi,\psi}(\mathbf{u}|=1)$ (p) $p_\psi(\mathbf{z})$ (q) $p_\psi(\mathbf{z}|y=0)$ (r) $p_\psi(\mathbf{z}|y=1)$

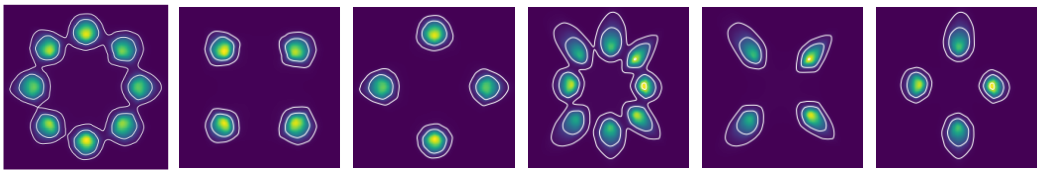
Figure 4: Marginalized and class-conditional feature as well as base distributions for MoG trained w/ IB.



(a) $p_{\phi,\psi}(\mathbf{u})$ (b) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (c) $p_{\phi,\psi}(\mathbf{u}|=1)$ (d) $p_\psi(\mathbf{z})$ (e) $p_\psi(\mathbf{z}|y=0)$ (f) $p_\psi(\mathbf{z}|y=1)$

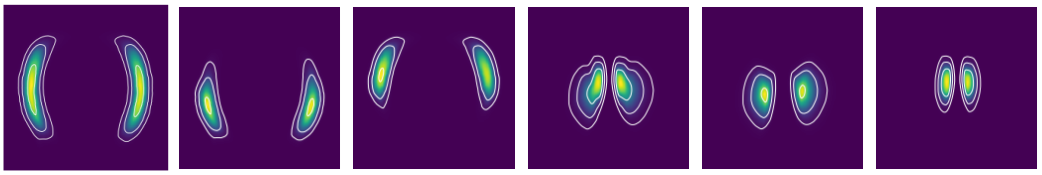


(g) $p_{\phi,\psi}(\mathbf{u})$ (h) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (i) $p_{\phi,\psi}(\mathbf{u}|=1)$ (j) $p_\psi(\mathbf{z})$ (k) $p_\psi(\mathbf{z}|y=0)$ (l) $p_\psi(\mathbf{z}|y=1)$

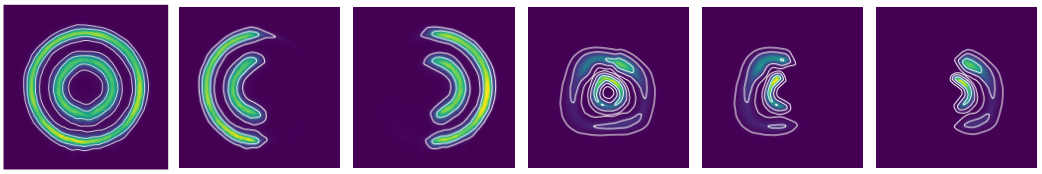


(m) $p_{\phi,\psi}(\mathbf{u})$ (n) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (o) $p_{\phi,\psi}(\mathbf{u}|=1)$ (p) $p_\psi(\mathbf{z})$ (q) $p_\psi(\mathbf{z}|y=0)$ (r) $p_\psi(\mathbf{z}|y=1)$

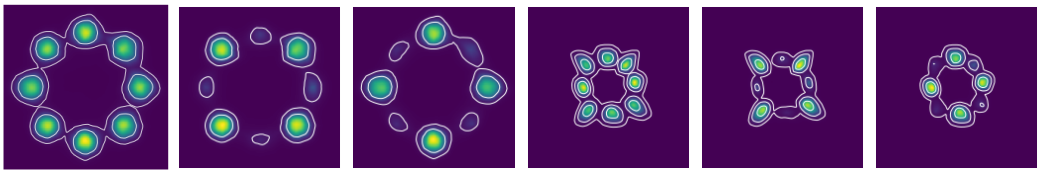
Figure 5: Marginalized and class-conditional feature as well as base distributions for cRSB trained w/o IB.



(a) $p_{\phi,\psi}(\mathbf{u})$ (b) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (c) $p_{\phi,\psi}(\mathbf{u}|=1)$ (d) $p_\psi(\mathbf{z})$ (e) $p_\psi(\mathbf{z}|y=0)$ (f) $p_\psi(\mathbf{z}|y=1)$

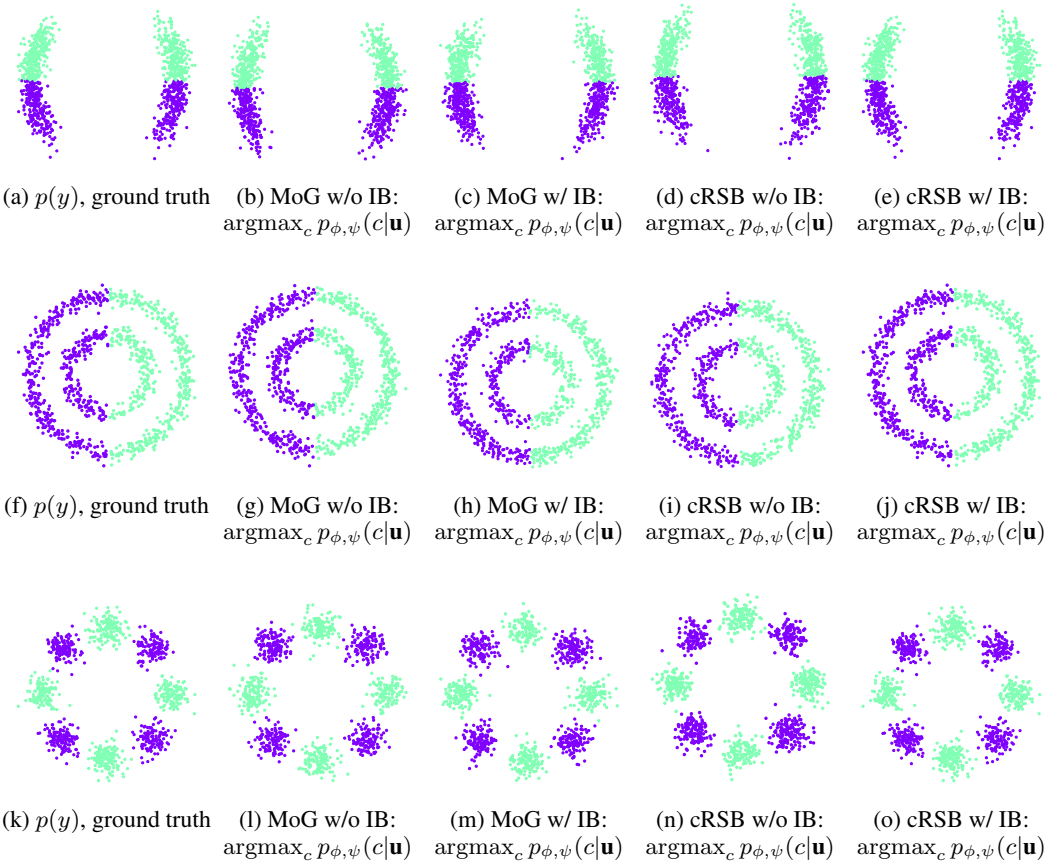


(g) $p_{\phi,\psi}(\mathbf{u})$ (h) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (i) $p_{\phi,\psi}(\mathbf{u}|=1)$ (j) $p_\psi(\mathbf{z})$ (k) $p_\psi(\mathbf{z}|y=0)$ (l) $p_\psi(\mathbf{z}|y=1)$



(m) $p_{\phi,\psi}(\mathbf{u})$ (n) $p_{\phi,\psi}(\mathbf{u}|y{=}0)$ (o) $p_{\phi,\psi}(\mathbf{u}|=1)$ (p) $p_\psi(\mathbf{z})$ (q) $p_\psi(\mathbf{z}|y=0)$ (r) $p_\psi(\mathbf{z}|y=1)$

Figure 6: Marginalized and class-conditional feature as well as base distributions for cRSB trained w/ IB.



(a) $p(y)$, ground truth (b) MoG w/o IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ (c) MoG w/ IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ (d) cRSB w/o IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ (e) cRSB w/ IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$

(f) $p(y)$, ground truth (g) MoG w/o IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ (h) MoG w/ IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ (i) cRSB w/o IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ (j) cRSB w/ IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$

(k) $p(y)$, ground truth (l) MoG w/o IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ (m) MoG w/ IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ (n) cRSB w/o IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ (o) cRSB w/ IB: $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$

Figure 7: Comparison of predictive capabilities $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ including training w/o and w/ IB for a sample of size 1,024 that was drawn from the target distribution.

5 Generative Classification

In Figure 7, we plot the ground truth assignments for a sample drawn from the respective data distribution along with the most likely class assignment $\operatorname{argmax}_c p_{\phi,\psi}(c|\mathbf{u})$ over varying base distribution and training objectives. Despite the perceptually different results in the density estimation (Section 4), all models classify the input data remarkably well. This highlights the importance of the IB objective since it allows to govern the tradeoff between the quality of density estimation and predictive capabilities.

References

[1] V. Stimper, B. Schölkopf, and J. M. Hernández-Lobato. Resampling base distributions of normalizing flows. In *International Conference on Artificial Intelligence and Statistics*, pages 4915–4936. PMLR, 2022.

[2] L. Ardizzone, R. Mackowiak, C. Rother, and U. Köthe. Training normalizing flows with the information bottleneck for competitive generative classification. *Advances in Neural Information Processing Systems*, 33:7828–7840, 2020.

[3] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[4] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[5] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[6] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

[7] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[8] D. Miller, N. Sünderhauf, M. Milford, and F. Dayoub. Uncertainty for identifying open-set errors in visual object detection. *IEEE Robotics and Automation Letters*, 7(1):215–222, 2021.

[9] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475, 2023.

[10] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010. URL http://dblp.uni-trier.de/db/journals/ijcv/ijcv88.html#EveringhamGWWZ10.

[11] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2014. URL http://arxiv.org/abs/1405.0312. cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list.

[12] M. Denninger, D. Winkelbauer, M. Sundermeyer, W. Boerdijk, M. W. Knauer, K. H. Strobl, M. Humt, and R. Triebel. Blenderproc2: A procedural pipeline for photorealistic rendering. *Journal of Open Source Software*, 8(82):4901, 2023.