Appendices

A. ACFlow for Categorical Data

In the main text, we mainly focus on real-valued data, but our model is also applicable with data that contains categorical dimensions. Consider a d-dimensional vector x with both real-valued and categorical features. The conditional distribution $p(x_u|x_o)$ can be factorized as $p(x_u^c|x_o)p(x_u^r|x_o, x_u^c)$, where x_u^r and x_u^c represent the real-valued and categorical components in x_u respectively. Since the conditioning inputs in Eq. (4) can be either real-valued or categorical, $p(x_u^r|x_o, x_u^c)$ can be directly modeled by ACFlow. However, ACFlow is not directly applicable to $p(x_u^c|x_o)$, because the change of variable theorem, Eq. (1), cannot be applied to categorical covariates. But we can use an autoregressive model, i.e., a special ACFlow without any transformations, to get likelihoods for categorical components. Hence, our proposed method can be applied to both real-valued and categorical data.

B. Synthetic Datasets

To validate the effectiveness of our model, we conduct experiments on synthetic 2-dimensional datasets (Behrmann et al., 2018), i.e., $x = (x_1, x_2) \in \mathbb{R}^2$. The joint distributions are plotted in Fig. 2. Here, the conditional distributions are highly multi-modal and vary significantly when conditioned on different observations. We train arbitrary conditional models on 100,000 samples from the joint distribution. The masks are generated by dropping out one dimension at random.

We compare our model against VAEAC, the current *state-of-the-art* model trained purely by likelihood. We closely follow the released code for VAEAC[1] to construct the proposal, prior, and generative networks. Specifically, we use fully connected layers and skip connections as is in their official implementation. We also use short-cut connections between the prior network and the generative network. We search over different combinations of the number of layers, the number of hidden units of fully connected layers, and the dimension of the latent code. Validation likelihoods are used to select the best model.

ACFlow is constructed by stacking multiple conditional transformations and an autoregressive likelihood. One conditional transformation layer (shown in Fig. B.1) contains one linear transformation, followed by one leaky relu transformation, and then one RNN coupling transformation. The DNN in the linear transformation is a 2-layer fully connected network with 256 units. RNN coupling is implemented as a 2-layer GRU with 256 hidden units. We stack four conditional transformation layers and use reverse transformations in between. The autoregressive likelihood model is a 2-layer GRU with 256 hidden units. The base distribution is a Gaussian Mixture Model with 40 components.

[1] https://github.com/tigvarts/vaeac/blob/master/imputation_networks.py

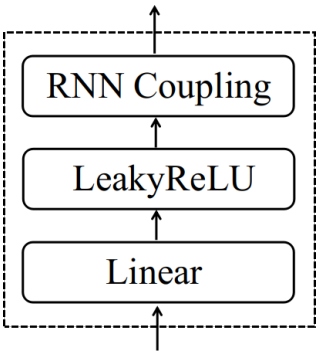


Figure B.1. One conditional transformation layer.

Fig. 2 shows imputation results from our model and VAEAC. We show imputations from both $p(x_1 \mid x_2, b)$ and $p(x_2 \mid x_1, b)$ by generating 10 samples conditioned on each observed covariate. Observed covariates are sampled from the marginal distribution $p(x_2)$ and $p(x_1)$ respectively. We see that ACFlow is capable of learning multi-modal distributions, while VAEAC tends to merge multiple modes into one single mode.

C. Image Datasets

C.1. Datasets and Masks

We compare ACFlow against baselines using three common image datasets, MNIST, Omniglot, and CelebA. MNIST contains grayscale images of size 28×28. Omniglot data is also resized to 28×28 and augmented with rotations. For CelebA dataset, we take a central crop of 128×128 and then resize it to 64×64. Since a flow model requires continuous inputs, we dequantize the images by adding independent uniform noise to them.

In order to train the conditional model $p(x_u \mid x_0, b)$, we manually generate the binary masks form some predefined distributions. We train ACFlow and baselines using a mixture of different masks. For MNIST and Omniglot, we use MCAR masks, rectangular masks, and half masks. For CelebA, other than the three masks described above, we also use a random pattern mask proposed in (Pathak et al., 2016) and the masks used in (Li et al., 2017). We shall describe the meaning of different masks below.

MCAR mask MCAR mask utilizes a pixelwise independent Bernoulli distribution with $p = 0.2$ to construct masks. That is, on average, we randomly drop out 80% of the pixels.

Rectangular mask We randomly generate a rectangle inside which pixels are marked as unobserved. The mask is generated by sampling a point to be the upper-left corner and randomly generating the height and width of the rectangle, although the area is required to be at least 10% of the full image area.

Half mask Half mask means either the upper, lower, left or right half of the image is unobserved.

Random pattern mask Random pattern means we mask out a random region with an arbitrary shape. We take the implementation from VAEAC[2].

Masks in (Li et al., 2017) They proposed six different masks to mask out different parts of a face, like eyes, mouth and checks.

C.2. Models

In this experiment, we compare to VAEAC (Ivanov et al., 2018) and an autoregressive model, PixelCNN (Salimans et al., 2017).

VAEAC released their implementation for the 128×128 CelebA dataset and we modify their code by removing the first pooling layer to suit 64×64 images. For MNIST and Omniglot, we build similar networks by using fewer convolution and pooling layers. Specifically, we pad the images to 32×32 and use 5 pooling layers to get a latent vector. We use 32 latent variables for MNIST and Omniglot and 256 latent variables for CelebA. We also use short-cut connections between the prior network and the generative network as in (Ivanov et al., 2018).

PixelCNN is originally proposed to model the joint likelihoods. However, since it decomposes the joint likelihood into a sequence of conditionals, it can be easily extended to model the arbitrary conditional distributions:

$$p(x_u \mid x_o, b) = \prod_{i=1}^{|u|} p(x_u^i \mid x_u^{<i}, x_o, b), \tag{14}$$

where x_u^i represents the ith dimension of x_u. In order to capture the dependencies over x_o and b, we apply an embedding network to get a vector embedding of them. Note that different from the original PixelCNN implementation, where they use discrete base distributions, we use mixture of Gaussian base distribution here to make the comparison fair.

ACFlow is implemented by replacing affine coupling in RealNVP (Dinh et al., 2016) with our proposed conditional affine coupling transformation. The DNN in affine coupling is implemented as a ResNet (He et al., 2016). For MNIST and Omniglot, we use 2 residual blocks. For CelebA, we use 4 blocks. We also use the multi scale architecture via the squeeze operation proposed in (Dinh et al., 2016). For MNIST and Omniglot, we apply 3 squeeze operations. For CelebA, we apply 4 squeeze operations. Note that we also need to squeeze the binary mask b in the same way to make sure it corresponds to x_o. The arbitrary conditional likelihood is chosen as either a Gaussian base likelihood model or an autoregressive one with the same formulation as the PixelCNN model described above.

For models trained with our proposed "best guess" penalty, we set the hyperparameter λ to 1.0. During preliminary experiments, we find our model is quite robust to different λ values.

C.3. Joint Likelihood

For a model trained with arbitrary conditional likelihood $p(x_u \mid x_o, b)$, we evaluate the joint likelihood by setting the binary mask b to zeros. We can of course sample from this joint distribution as well. We show some samples in Fig. C.2. We see that the model generate decent samples although it never observes the complete data during training.

C.4. Gibbs Sampling

Due to space limitation, we only show part of the mixing steps in the Gibbs sampling procedure. In Figure. C.3, we show all the 50 steps. Please zoom in for better visualization.

C.5. Additional Inpaintings

We show additional inpainting results from ACFlow in Fig. C.5. We also show some "best guess" inpaintings obtained by a model trained with the auxiliary objective in Fig. C.4. As we expected, the "best guess" imputations tend to be blurry due to the MSE penalty. However, the samples from the same model are still diverse and coherent as can be seen from Fig. C.6.

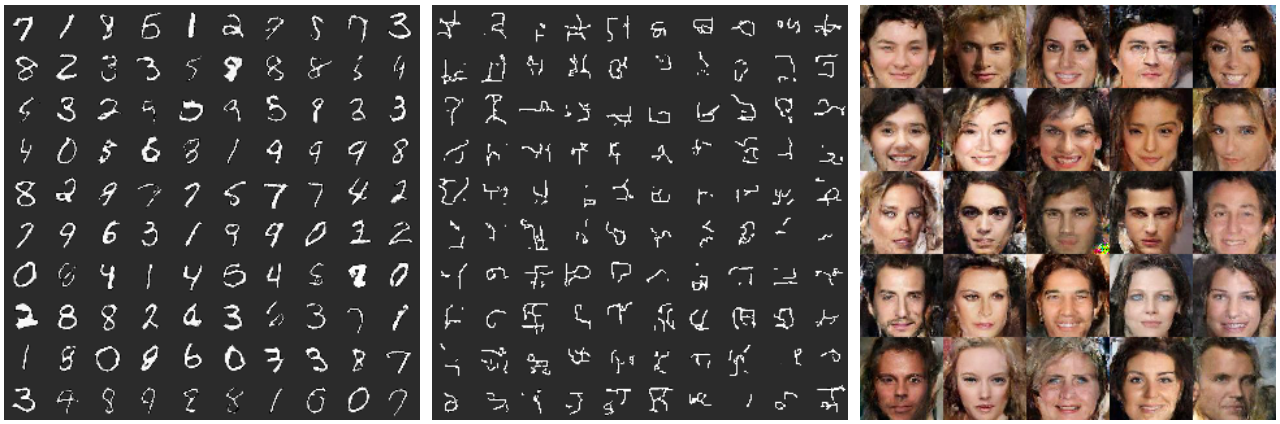
D. Real-valued Datasets

D.1. Models

In this experiment, we compare to VAEAC and an autoregressive model in term of the likelihood. We also compare to a GAN based method, GAIN, and two classic imputation methods, MICE and MisForest in terms of the imputation performance.

We modify the official VAEAC implementation for this experiment. In their original implementation, they restricted the generative network to output a distribution with variance

[2] https://github.com/tigvarts/vaeac/blob/master/mask_generators.py#L100



(a) MNIST (b) Omniglot (c) CelebA

Figure C.2. Joint sampling from a model trained with $p(x_u \mid x_o, b)$ by setting b to zeros.

Figure C.3. Gibbs sampling using a pretrained ACFlow. We iteratively sample the upper and lower half for 50 steps conditioned on the remaining part.

equal to $\mathbf{1}$. We found that learning the variance can improve VAEAC's likelihood significantly and gives comparable NRMSE scores. Note that during sampling, we still use the mean of the outputs from the generative network, because it gives slightly better NRMSE scores.

The autoregressive baseline is implemented as a 4-layer GRU netowrk with 256 hidden units. We did not observe further improvements when adding more layers.

We build ACFlow by combining 6 conditional transformation layers (shown in Fig. B.1) with a 4-layer GRU autoregerssive likelihood model. The base distribution is a Gaussian Mixture with 40 components. "ACFlow+BG" uses the same network architecture but trained with $\lambda = 1.0$. We search over 0.1, 1.0, and 10.0 for λ and find it gives comparable likelihood for all datasets. Hence, we only report results when $\lambda = 1.0$ in the main text.

For GAIN, we use the variant the VAEAC authors proposed. They observed consistent improvement over the original one on UCI datasets. When we have complete training data, we add another MSE loss over the unobserved covariates to train the generator.

MICE and MissForest are trained using default parameters in the R packages.

The autoencoder baseline is implemented as a 6-layer fully connected network with 256 hidden units. ReLU is used after each fully connected layer except the last one. We use standard Gaussian as the base distribution.

D.2. Marginal Likelihood

In the main text, we compare the marginal likelihood to TANs that are trained specifically for the corresponding marginal distributions. In Figure. D.7, we qualitatively compare them by showing the scatter plots of the samples. We draw 1024 samples from both ACFlow and TANs, and also sample 1024 real data.

D.3. Imputation Results

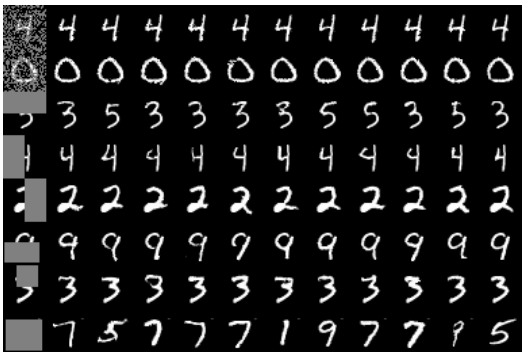
We list all imputation results in Table. D.1. We report mean and standard deviation by generating 5 different masks for each test data point.

Table D.1. Missing feature imputation results. Lower is better for both NRMSE and NLL. Numbers inside a parentheses are standard deviation for 5 randomly generated binary mask.

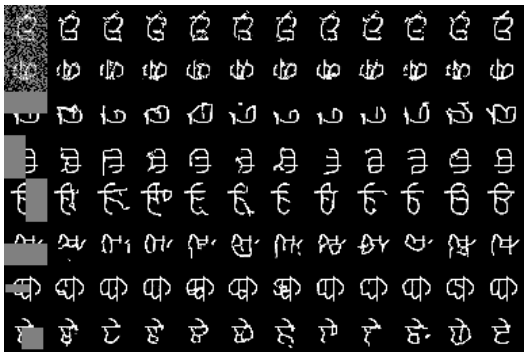
Missing Rate	Method		bsds	gas	hepmass	miniboone	power
p=0	GAIN	NRMSE	0.895±0.151	0.715±0.041	0.948±0.006	0.620±0.002	0.949±0.017
	AutoEncoder	NRMSE	0.635±0.000	1.016±0.178	0.930±0.001	0.483±0.003	0.887±0.002
		NLL	3.502±0.014	-2.543±0.115	12.707±0.018	6.861±0.114	2.488±0.039
	VAEAC	NRMSE	0.615±0.000	0.574±0.033	0.896±0.001	0.462±0.002	0.880±0.001
		NLL	1.708±0.005	-2.418±0.006	10.082±0.010	3.452±0.067	0.042±0.002
	AutoRegressive	NRMSE	0.652±0.000	0.457±0.011	0.903±0.001	0.460±0.005	0.877±0.002
		NLL	-1.73±0.010	-7.646±0.009	6.428±0.006	-0.057±0.022	-0.399±0.003
	ACFlow	NRMSE	0.603±0.000	0.567±0.050	0.909±0.000	0.478±0.004	0.877±0.001
		NLL	-5.269±0.007	-8.086±0.010	8.197±0.008	0.972±0.022	-0.561±0.003
	ACFlow+BG	NRMSE	0.572±0.000	0.369±0.016	0.861±0.001	0.442±0.001	0.833±0.002
		NLL	-4.841±0.008	-7.593±0.011	6.833±0.006	1.098±0.032	-0.528±0.003
p=0.1	MICE	NRMSE	0.631±0.003	0.518±0.004	0.964±0.004	0.605±0.004	0.911±0.008
	MissForest	NRMSE	0.665±0.002	0.418±0.005	0.985±0.002	0.561±0.003	0.991±0.019
	GAIN	NRMSE	0.749±0.128	0.502±0.070	1.024±0.023	0.615±0.017	1.074±0.038
	AutoEncoder	NRMSE	0.648±0.001	0.761±0.095	0.936±0.001	0.498±0.002	0.887±0.002
		NLL	4.300±0.038	-2.266±0.169	12.851±0.012	7.305±0.043	2.521±0.028
	VAEAC	NRMSE	0.620±0.000	0.558±0.047	0.899±0.000	0.467±0.004	0.881±0.003
		NLL	2.245±0.015	-2.823±0.009	10.389±0.005	4.242±0.071	0.103±0.005
	AutoRegressive	NRMSE	0.752±0.000	0.472±0.011	0.915±0.000	0.539±0.004	0.876±0.001
		NLL	3.233±0.015	-7.536±0.009	7.824±0.006	5.409±0.066	-0.466±0.003
	ACFlow	NRMSE	0.610±0.000	0.588±0.025	0.908±0.001	0.533±0.005	0.877±0.002
		NLL	-4.225±0.018	-7.568±0.005	7.784±0.006	5.150±0.053	-0.557±0.003
	ACFlow+BG	NRMSE	0.586±0.001	0.384±0.018	0.863±0.001	0.468±0.003	0.836±0.002
		NLL	-3.187±0.017	-7.212±0.008	9.670±0.007	3.577±0.057	-0.510±0.003
p=0.5	MICE	NRMSE	0.628±0.001	0.539±0.005	0.969±0.002	0.615±0.002	0.916±0.005
	MissForest	NRMSE	0.662±0.001	0.436±0.003	0.990±0.002	0.573±0.005	0.990±0.012
	GAIN	NRMSE	0.929±0.123	1.152±0.180	1.143±0.035	0.800±0.042	1.101±0.044
	AutoEncoder	NRMSE	0.739±0.001	0.618±0.056	0.962±0.001	0.567±0.005	0.905±0.002
		NLL	10.078±0.021	0.990±0.097	13.482±0.012	10.775±0.091	2.858±0.047
	VAEAC	NRMSE	0.666±0.001	0.531±0.036	0.915±0.001	0.513±0.004	0.892±0.002
		NLL	9.930±0.029	-1.952±0.023	11.415±0.012	9.051±0.079	0.343±0.004
	AutoRegressive	NRMSE	0.879±0.001	0.483±0.021	0.937±0.000	0.644±0.002	0.882±0.002
		NLL	11.348±0.010	-5.723±0.004	9.760±0.007	11.024±0.069	-0.363±0.003
	ACFlow	NRMSE	0.667±0.001	0.488±0.030	0.938±0.000	0.614±0.004	0.890±0.000
		NLL	1.508±0.010	-5.405±0.008	10.538±0.006	9.892±0.084	-0.458±0.005
	ACFlow+BG	NRMSE	0.645±0.000	0.421±0.016	0.890±0.000	0.582±0.007	0.843±0.001
		NLL	3.497±0.015	-4.818±0.009	10.975±0.006	10.849±0.105	-0.417±0.005

(a) MNIST (b) Omniglot (c) CelebA

Figure C.4. Single imputation from our "best guess". Left: inputs. Middle: best guess imputation. Right: groundtruth.



(a) MNIST

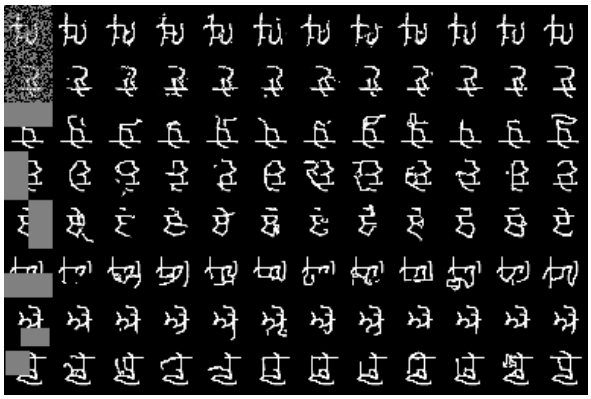


(b) Omniglot

(c) CelebA

Figure C.5. Additional inpaintings from ACFlow. Left: inputs. Middle: Samples. Right: groundtruth.

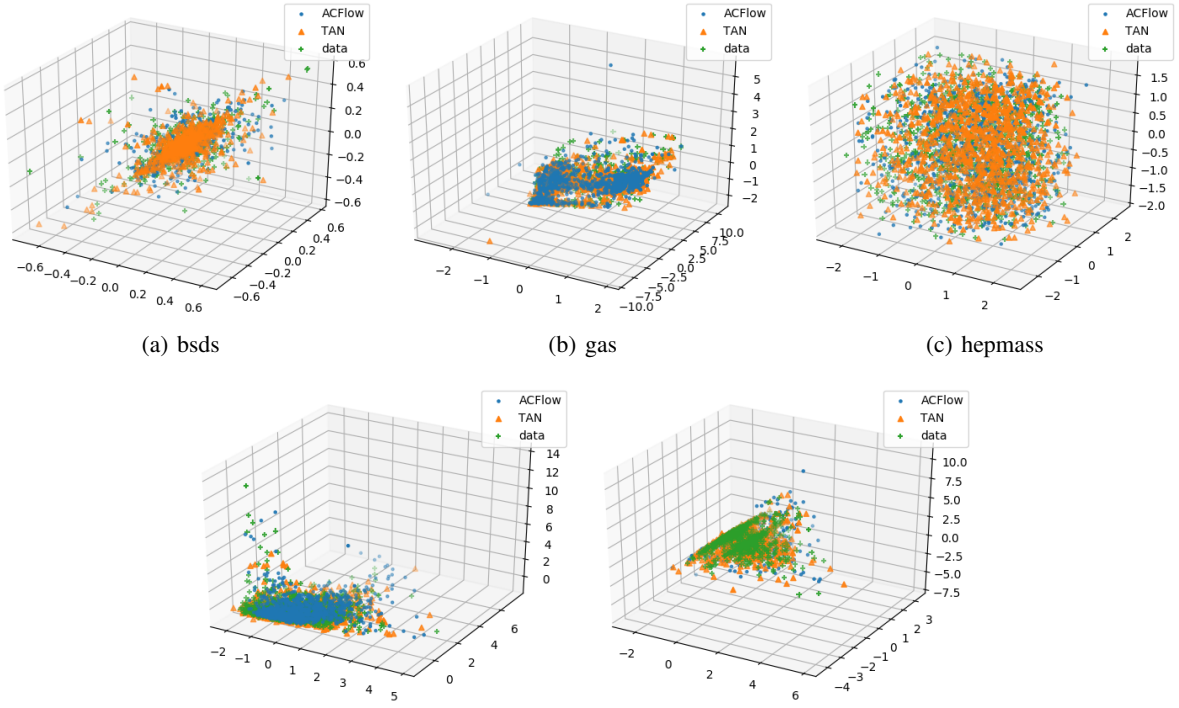
(a) MNIST



(b) Omniglot

(c) CelebA

Figure C.6. Multiple imputations from ACFlow+BG. Left:inputs. Middle: Samples. Right: groundtruth.



(a) bsds (b) gas (c) hepmass

(d) miniboone (e) power

Figure D.7. sample the first 3 dimensions from the learned marginal distributions.