NF 2: Autoregressive Flows

CS 598: Deep Generative and Dynamical Models

Instructor: Arindam Banerjee

October 5, 2021

Variational Inference with Normalizing Flows

- Goal is to optimize the ELBO

$$\mathcal{L}(\mathrm{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})}[\log p_\theta(\mathrm{x},\mathrm{z}) - \log q_\phi(\mathrm{z}|\mathrm{x})]$$

- Create flexible $q(\mathrm{z}|\mathrm{x})$ in two steps
  - Inference network to get $q(\mathrm{z}_0|\mathrm{x})$

$$q(\mathrm{z}_0|\mathrm{x}) \sim \mathcal{N}(\mathrm{z}_0|\mu(\mathrm{x}),\sigma^2(\mathrm{x}))$$

  - Normalizing flow to get $\mathrm{z}_t = f_t(\mathrm{z}_{t-1};\mathrm{x}))) \;, \quad t = 1,\ldots,T$

$$\log q(\mathrm{z}_T|\mathrm{x}) = \log q(\mathrm{z}_0|\mathrm{x}) - \sum_{t=1}^{T} \log\det\left|\frac{dz_t}{dz_{t-1}}\right|$$

Linear Vector Autoregressive (VAR) Models

- Classical order k linear VAR model

$$y_t = A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_k y_{t-k} + c + e_t$$

  - $y_\tau, c, e_\tau \in \mathbb{R}^d$
  - $A_\tau \in \mathbb{R}^{d \times d}$

- The error terms e_t are unbiased and uncorrelated
  - Unbiased: $\mathbb{E}[e_t] = 0$
  - Covariance structure: $\mathbb{E}[e_t e_t^T] = \Sigma$
  - Uncorrelated: $\mathbb{E}[e_t e_\tau^T] = 0, \ \tau < t$

- For Gaussian errors, uncorrelated implies independent

Non-linear (Deep) Autoregressive Models

- Sample a noise vector $\epsilon \sim \mathcal{N}(0, \mathbb{I})$
- Autoregressive model with non-linear mean and variances

$$y_0 = \mu_0 + \sigma_0 \cdot \epsilon_0$$
$$y_i = \mu_i(y_{1:i-1}) + \sigma_i(y_{1:i-1}) \cdot \epsilon_i$$

- μ_i, σ_i can be non-linear (deep) functions
- Sampling is sequential, scales with dimensionality
  - Not suitable for posterior sampling
  - Variational inference needs sampling, not suitable for VI

Inverse Autoregressive Transformations

- Inverse of autoregressive transformation

$$\epsilon_i = \frac{y_i - \mu_i(\mathrm{y}_{1:i-1})}{\sigma_i(\mathrm{y}_{1:i-1})}$$

  - Maps 'signal' $y_i, \mathrm{y}_{1:i-1}$ to 'noise' ϵ_i

- Inverse autoregressive transformation as normalizing flows
  - Given y, ϵ can be computed directly, non-sequentially

$$\epsilon = \frac{\mathrm{y} - \mu(\mathrm{y})}{\sigma(\mathrm{y})}$$

  - Has a simple Jacobian determinant

$$\log\det\left|\frac{d\epsilon}{d\mathrm{y}}\right| = \sum_{i=1}^{D} -\log\sigma_i(\mathrm{y})$$

Inverse Autoregressive Flows (IAF)

- Initial encoder network gives $\mu_0(\mathrm{x}), \sigma_0(\mathrm{x})$, extra output h
- With $\epsilon \sim \mathcal{N}(0, \mathrm{I})$, initial latent variable

$$\mathrm{z}_0 = \mu_0 + \sigma_0 \odot \epsilon$$

- Flow consists of a chain of transformations

$$\mathrm{z}_t = \mu_t + \sigma_t \odot \mathrm{z}_{t-1}$$

  - Autoregressive networks $\mu_t(\mathrm{z}_{1:t-1}, \mathrm{h})$ and $\sigma_t(\mathrm{z}_{1:t-1}, \mathrm{h})$

- Overall model autoregressive w.r.t. $\mathrm{z}_{1:t-1}$
  - Jacobians $\frac{d\mu_t}{d\mathrm{z}_{t-1}}, \frac{d\sigma_t}{d\mathrm{z}_{t-1}}$ are triangular with 0 diagonals

IAF Properties

- Jacobians $\frac{dz_t}{dz_{t-1}}$ is triangular with σ_t on diagonals
- Determinant is $\prod_{i=1}^{D} \sigma_{t,i}$
- Final iterate is given by

$$\log q(z_T|x) = -\sum_{i=1}^{D}\left(\frac{1}{2}\epsilon_i^2 + \frac{1}{2}\log 2\pi + \sum_{t=0}^{T}\log \sigma_{t,i}\right)$$

- Numerically stable version

$$[m_t, s_t] = \text{ARNN}[t](z_{t-1}, h; \theta)$$
$$\sigma_t = \text{sigmoid}(s_t)$$
$$z_t = \sigma_t \odot z_{t-1} + (1 - \sigma_t) \odot m_t$$

- Reverse the ordering of variables after each step

IAF Model

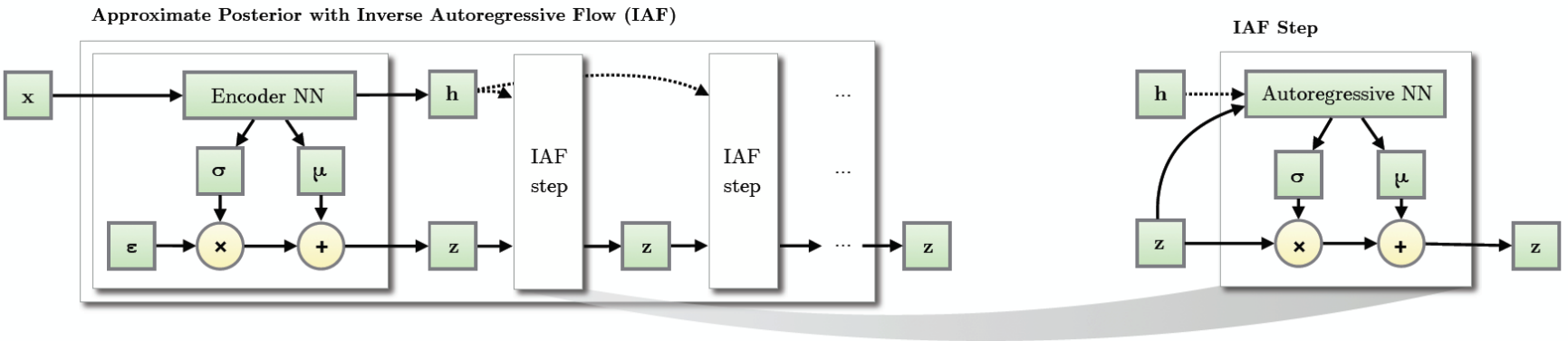


Figure 2: Like other normalizing flows, drawing samples from an approximate posterior with Inverse Autoregressive Flow (IAF) consists of an initial sample $\mathbf{z}$ drawn from a simple distribution, such as a Gaussian with diagonal covariance, followed by a chain of nonlinear invertible transformations of $\mathbf{z}$, each with a simple Jacobian determinants.

IAF Algorithm

Algorithm 1: Pseudo-code of an approximate posterior with Inverse Autoregressive Flow (IAF)

Data:

$\mathbf{x}$: a datapoint, and optionally other conditioning information
$\boldsymbol{\theta}$: neural network parameters
$\texttt{EncoderNN}(\mathbf{x}; \boldsymbol{\theta})$: encoder neural network, with additional output $\mathbf{h}$
$\texttt{AutoregressiveNN}[*](\mathbf{z}, \mathbf{h}; \boldsymbol{\theta})$: autoregressive neural networks, with additional input $\mathbf{h}$
$\texttt{sum}(.)$: sum over vector elements
$\texttt{sigmoid}(.)$: element-wise sigmoid function

Result:

$\mathbf{z}$: a random sample from $q(\mathbf{z}|\mathbf{x})$, the approximate posterior distribution
l: the scalar value of $\log q(\mathbf{z}|\mathbf{x})$, evaluated at sample '$\mathbf{z}$'

$[\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{h}] \leftarrow \texttt{EncoderNN}(\mathbf{x}; \boldsymbol{\theta})$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
$\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}$
$l \leftarrow -\texttt{sum}(\log \boldsymbol{\sigma} + \frac{1}{2}\boldsymbol{\epsilon}^2 + \frac{1}{2}\log(2\pi))$
for $t \leftarrow 1$ **to** T **do**
  $[\mathbf{m}, \mathbf{s}] \leftarrow \texttt{AutoregressiveNN}[t](\mathbf{z}, \mathbf{h}; \boldsymbol{\theta})$
  $\boldsymbol{\sigma} \leftarrow \texttt{sigmoid}(\mathbf{s})$
  $\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \mathbf{z} + (1 - \boldsymbol{\sigma}) \odot \mathbf{m}$
  $l \leftarrow l - \texttt{sum}(\log \boldsymbol{\sigma})$
end

ResNet VAE with Bidirectional Inference

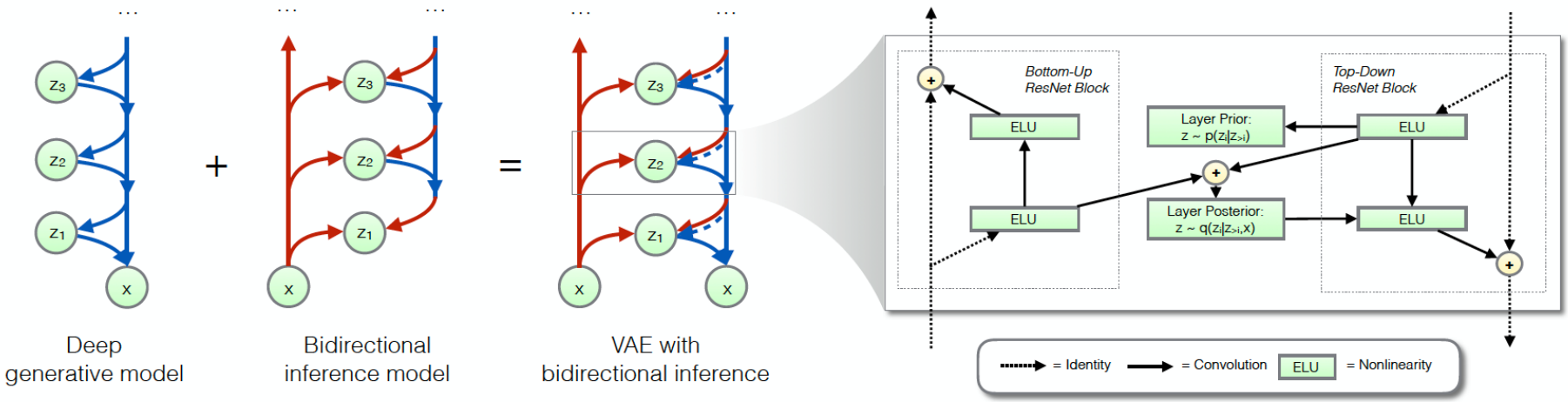


Figure 3: Overview of our ResNet VAE with bidirectional inference. The posterior of each layer is parameterized by its own IAF.

Results: Likelihoods, MNIST

Table 1: Generative modeling results on the dynamically sampled binarized MNIST version used in previous publications (Burda et al., 2015). Shown are averages; the number between brackets are standard deviations across 5 optimization runs. The right column shows an importance sampled estimate of the marginal likelihood for each model with 128 samples. Best previous results are reproduced in the first segment: [1]: (Salimans et al., 2014) [2]: (Burda et al., 2015) [3]: (Kaae Sønderby et al., 2016) [4]: (Tran et al., 2015)

Model	VLB	$\log p(\mathbf{x}) \approx$
Convolutional VAE + HVI [1]	-83.49	-81.94
DLGM 2hl + IWAE [2]		-82.90
LVAE [3]		-81.74
DRAW + VGP [4]	-79.88	
Diagonal covariance	-84.08 ($\pm$ 0.10)	-81.08 ($\pm$ 0.08)
IAF (Depth $= 2$, Width $= 320$)	-82.02 ($\pm$ 0.08)	-79.77 ($\pm$ 0.06)
IAF (Depth $= 2$, Width $= 1920$)	-81.17 ($\pm$ 0.08)	-79.30 ($\pm$ 0.08)
IAF (Depth $= 4$, Width $= 1920$)	-80.93 ($\pm$ 0.09)	-79.17 ($\pm$ 0.08)
IAF (Depth $= 8$, Width $= 1920$)	-80.80 ($\pm$ 0.07)	**-79.10** ($\pm$ 0.07)

Results: Bits/dim, Cifar-10

Table 2: Our results with ResNet VAEs on CIFAR-10 images, compared to earlier results, in *average number of bits per data dimension* on the test set. The number for convolutional DRAW is an upper bound, while the ResNet VAE log-likelihood was estimated using importance sampling.

Method	bits/dim $\leq$
Results with tractable likelihood models:	
Uniform distribution (van den Oord et al., 2016b)	8.00
Multivariate Gaussian (van den Oord et al., 2016b)	4.70
NICE (Dinh et al., 2014)	4.48
Deep GMMs (van den Oord and Schrauwen, 2014)	4.00
Real NVP (Dinh et al., 2016)	3.49
PixelRNN (van den Oord et al., 2016b)	**3.00**
Gated PixelCNN (van den Oord et al., 2016c)	**3.03**
Results with variationally trained latent-variable models:	
Deep Diffusion (Sohl-Dickstein et al., 2015)	5.40
Convolutional DRAW (Gregor et al., 2016)	3.58
ResNet VAE with IAF (Ours)	**3.11**

Neural Density Estimation

- Two faimilies: AR models and NF models
- Normalizing Flows
  - Sequence of invertible transformations of a base distribution
  - "... not well suited for density estimation ..."
- Autoregressive density estimation
  - Chain rule $p(\mathrm{x}) = \prod_i p(x_i|\mathrm{x}_{1:i-1})$
  - Sequential and sensitive to variable order

Autoregression as Normalizing Flows

- Autoregression with Gaussians
  $p(x_i|x_{1:i-1}) = \mathcal{N}(x_i|\mu_i, (\exp \alpha_i)^2)$, $\mu_i = f_{\mu_i}(x_{1:i-1}), \alpha_i = f_{\alpha_i}(x_{1:i-1})$
- Generating samples using $u_i \sim \mathcal{N}(0, 1)$
  $$x_i = u_i \exp \alpha_i + \mu_i \ , \quad \mu_i = f_{\mu_i}(x_{1:i-1}), \alpha_i = f_{\alpha_i}(x_{1:i-1})$$
- With $x = f(u), u \sim \mathcal{N}(0, \mathbb{I})$, consider f^{-1}
  - Given x, $u = f^{-1}(x)$ is easy to compute
    $$u_i = (x_i - \mu_i) \exp(-\alpha_i) \ , \quad \mu_i = f_{\mu_i}(x_{1:i-1}), \alpha_i = f_{\alpha_i}(x_{1:i-1})$$
  - Jacobian of f^{-1} is traingular by design with diagonal $\exp(-\alpha_i)$
    $$\left| \det \frac{\partial f^{-1}}{\partial x} \right| = \exp \left(- \sum_i \alpha_i \right) \ , \quad \alpha_i = f_{\alpha_i}(x_{1:i-1})$$

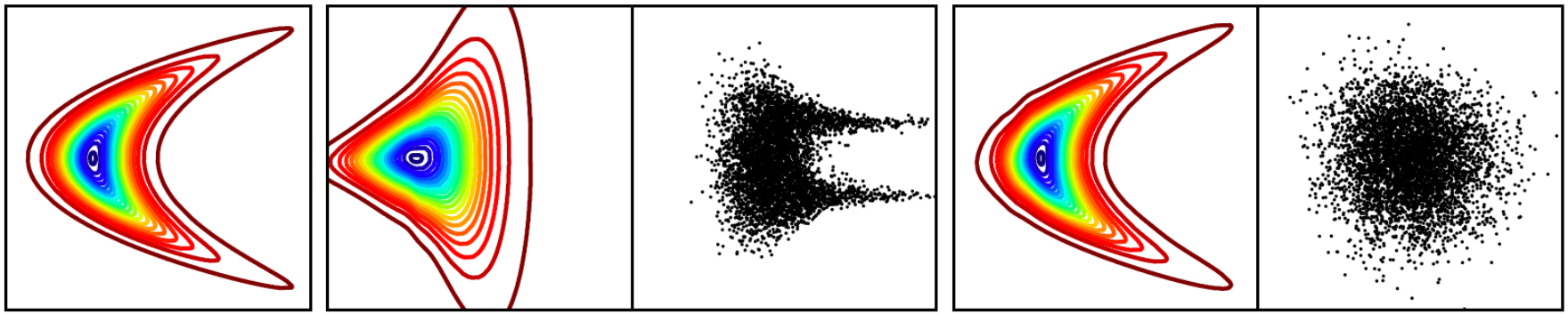
Masked Autoregressive Flows (MAF)

- Use autoregressive model as a flow
- Assess generative model by checking if $\mathrm{u} = f^{-1}(\mathrm{x})$ is Gaussian

$$u_i = (x_i - \mu_i)\exp(-\alpha_i) \ , \quad \mu_i = f_{\mu_i}(\mathrm{x}_{1:i-1}), \alpha_i = f_{\alpha_i}(\mathrm{x}_{1:i-1})$$

- Stack a set of AR models $M_1, \ldots, M_k$
- Implement $\{f_{\mu_i}, f_{\alpha_i}\}$ with masking, avoid sequential computation
  - Use MADE, autoregressive property implemented by masking
- MAF is stacked MADE, each layer is a MADE

Example: MADE vs MAF



(a) Target density (b) MADE with Gaussian conditionals (c) MAF with 5 layers

Figure 1: **(a)** The density to be learnt, defined as $p(x_1, x_2) = \mathcal{N}(x_2 \,|\, 0, 4)\mathcal{N}\left(x_1 \,|\, \frac{1}{4}x_2^2, 1\right)$. **(b)** The density learnt by a MADE with order (x_1, x_2) and Gaussian conditionals. Scatter plot shows the train data transformed into random numbers $\mathbf{u}$; the non-Gaussian distribution indicates that the model is a poor fit. **(c)** Learnt density and transformed train data of a 5 layer MAF with the same order (x_1, x_2).

MAF and IAF

- Both use flows based autoregression
- IAF generates (μ_i, α_i) from past random numbers

$$x_i = u_i \exp \alpha_i + \mu_i \ , \quad \mu_i = f_{\mu_i}(u_{1:i-1}), \alpha_i = f_{\alpha_i}(u_{1:i-1})$$

  - Generate samples and compute their likelihood in one pass
  - Likelihood of new x: sequentially generate u, needs D passes

- MAF generates (μ_i, α_i) from past data variables

$$x_i = u_i \exp \alpha_i + \mu_i \ , \quad \mu_i = f_{\mu_i}(x_{1:i-1}), \alpha_i = f_{\alpha_i}(x_{1:i-1})$$

  - Likelihood of any x in one pass
  - Sampling x is sequential, needs D passes

MAF and Real NVP

- Real NVP uses coupling layers

$$\mathrm{x}_{1:d} = \mathrm{u}_{1:d}$$

$$\mathrm{x}_{d+1:D} = \mathrm{u}_{d+1:D} \odot \exp \alpha + \mu \ , \quad \mu = f_\mu(\mathrm{u}_{1:d}), \alpha = f_\alpha(\mathrm{u}_{1:d})$$

  - Stack coupling layers by permuting variables
  - Special case $\alpha = 0$ corresponds to NICE (NF1, first paper)

- Special cases of MAF (and IAF), which have fine-grained control

$$x_i = u_i \exp \alpha_i + \mu_i \ , \quad \mu_i = f_{\mu_i}(\mathrm{x}_{1:i-1}), \alpha_i = f_{\alpha_i}(\mathrm{x}_{1:i-1})$$

- Generating samples, i.e., draw $x \sim p(\mathrm{x})$
  - One pass for Real NVP and IAF, D passes for MAF

- Likelihood computation, i.e., given x, compute $p(\mathrm{x})$
  - One pass for Real NVP and MAF, D passes for IAF

Results: Likelihood

Table 1: Average test log likelihood (in nats) for unconditional density estimation. The best performing model for each dataset is shown in bold (multiple models are highlighted if the difference is not statistically significant according to a paired t-test). Error bars correspond to 2 standard deviations.

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
Gaussian	-7.74 ± 0.02	-3.58 ± 0.75	-27.93 ± 0.02	-37.24 ± 1.07	96.67 ± 0.25
MADE	-3.08 ± 0.03	3.56 ± 0.04	-20.98 ± 0.02	-15.59 ± 0.50	148.85 ± 0.28
MADE MoG	$\mathbf{0.40 \pm 0.01}$	8.47 ± 0.02	$\mathbf{-15.15 \pm 0.02}$	-12.27 ± 0.47	153.71 ± 0.28
Real NVP (5)	-0.02 ± 0.01	4.78 ± 1.80	-19.62 ± 0.02	-13.55 ± 0.49	152.97 ± 0.28
Real NVP (10)	0.17 ± 0.01	8.33 ± 0.14	-18.71 ± 0.02	-13.84 ± 0.52	153.28 ± 1.78
MAF (5)	0.14 ± 0.01	9.07 ± 0.02	-17.70 ± 0.02	$\mathbf{-11.75 \pm 0.44}$	155.69 ± 0.28
MAF (10)	0.24 ± 0.01	$\mathbf{10.08 \pm 0.02}$	-17.73 ± 0.02	-12.24 ± 0.45	154.93 ± 0.28
MAF MoG (5)	0.30 ± 0.01	9.59 ± 0.02	-17.39 ± 0.02	$\mathbf{-11.68 \pm 0.44}$	$\mathbf{156.36 \pm 0.28}$

Results: Likelihood, Cifar-10

Table 2: Average test log likelihood (in nats) for conditional density estimation. The best performing model for each dataset is shown in bold. Error bars correspond to 2 standard deviations.

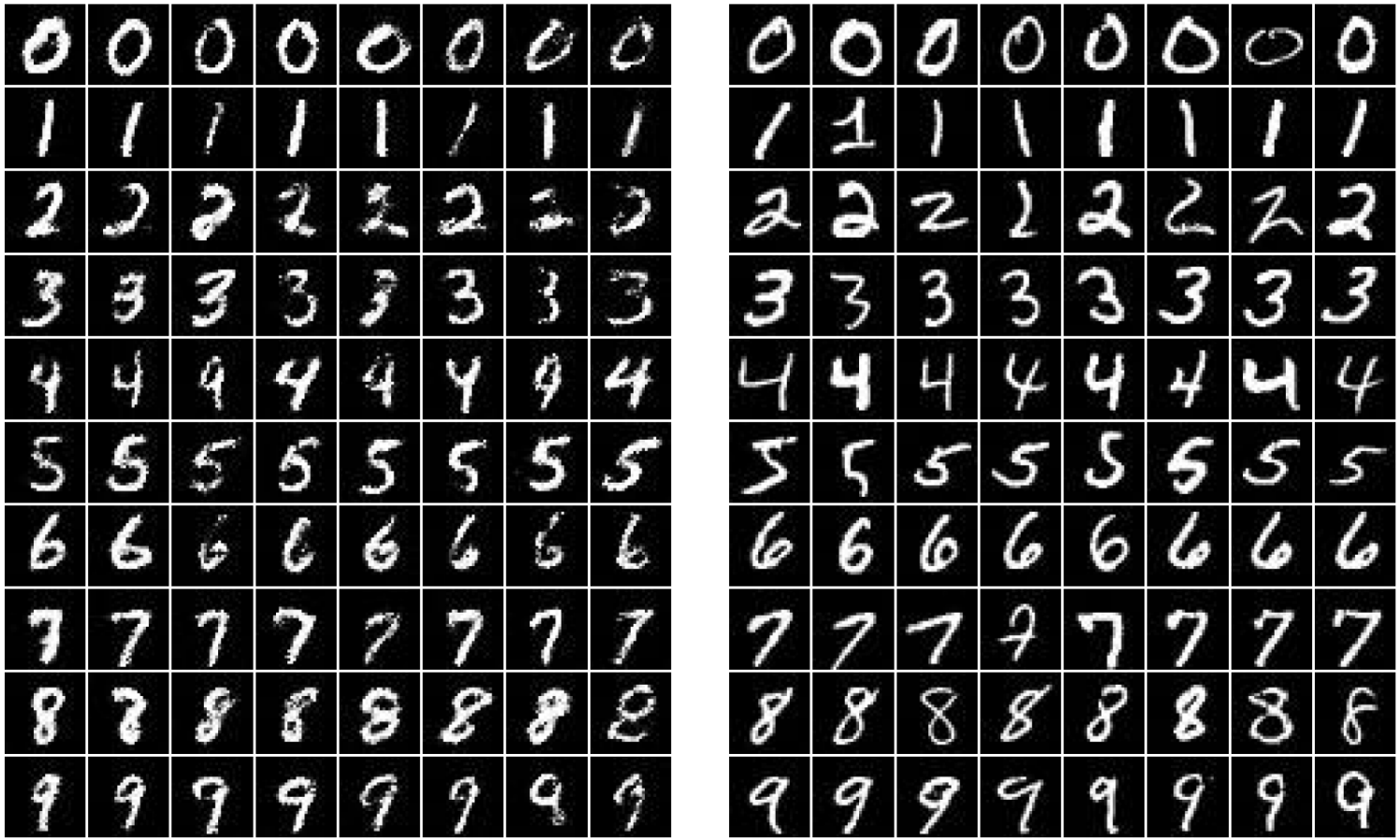
	MNIST		CIFAR-10	
	unconditional	conditional	unconditional	conditional
Gaussian	-1366.9 ± 1.4	-1344.7 ± 1.8	2367 ± 29	2030 ± 41
MADE	-1380.8 ± 4.8	-1361.9 ± 1.9	147 ± 20	187 ± 20
MADE MoG	$\mathbf{-1038.5 \pm 1.8}$	$\mathbf{-1030.3 \pm 1.7}$	-397 ± 21	-119 ± 20
Real NVP (5)	-1323.2 ± 6.6	-1326.3 ± 5.8	2576 ± 27	2642 ± 26
Real NVP (10)	-1370.7 ± 10.1	-1371.3 ± 43.9	2568 ± 26	2475 ± 25
MAF (5)	-1300.5 ± 1.7	-1302.9 ± 1.7*	2936 ± 27	2983 ± 26*
MAF (10)	-1313.1 ± 2.0	-1316.8 ± 1.8*	$\mathbf{3049 \pm 26}$	$\mathbf{3058 \pm 26}$*
MAF MoG (5)	-1100.3 ± 1.6	-1092.3 ± 1.7	2911 ± 26	2936 ± 26

Results: Bits/dim, Conditional Models

Table 7: Bits per pixel for conditional density estimation (lower is better). The best performing model for each dataset is shown in bold. Error bars correspond to 2 standard deviations.

	MNIST		CIFAR-10	
	unconditional	conditional	unconditional	conditional
Gaussian	2.01 ± 0.01	1.97 ± 0.01	4.63 ± 0.01	4.79 ± 0.02
MADE	2.04 ± 0.01	2.00 ± 0.01	5.67 ± 0.01	5.65 ± 0.01
MADE MoG	$\mathbf{1.41 \pm 0.01}$	$\mathbf{1.39 \pm 0.01}$	5.93 ± 0.01	5.80 ± 0.01
Real NVP (5)	1.93 ± 0.01	1.94 ± 0.01	4.53 ± 0.01	4.50 ± 0.01
Real NVP (10)	2.02 ± 0.02	2.02 ± 0.08	4.54 ± 0.01	4.58 ± 0.01
MAF (5)	1.89 ± 0.01	1.89 ± 0.01*	4.36 ± 0.01	4.34 ± 0.01*
MAF (10)	1.91 ± 0.01	1.92 ± 0.01*	$\mathbf{4.31 \pm 0.01}$	$\mathbf{4.30 \pm 0.01}$*
MAF MoG (5)	1.52 ± 0.01	1.51 ± 0.01	4.37 ± 0.01	4.36 ± 0.01

Results: Samples, MNIST

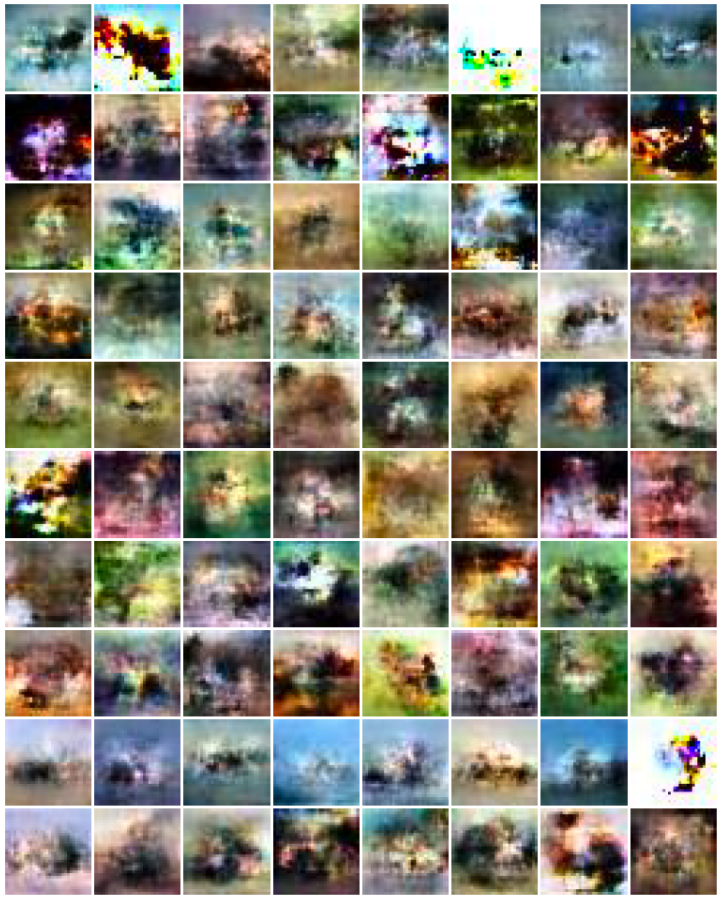


(a) Generated images

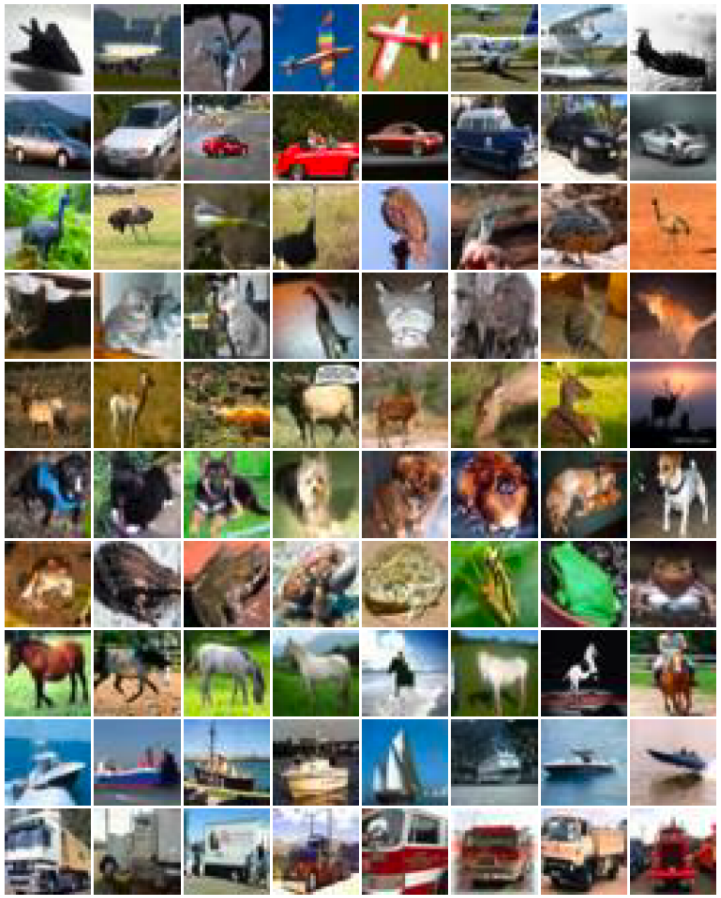
(b) Real images

Figure 3: Class-conditional generated and real images from MNIST. Rows are different classes. Generated images are sorted by decreasing log likelihood from left to right.

Results: Samples, CIFAR-10



(a) Generated images



(b) Real images

Figure 4: Class-conditional generated and real images from CIFAR-10. Rows are different classes. Generated images are sorted by decreasing log likelihood from left to right.

References

- D. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, M. Welling. Improved Variational Inference with Inverse Autoregressive Flow. NeurIPS, 2016.
- G. Papamakarios, T. Pavlakou, I. Murray. Masked Autoregressive Flow for Density Estimation. NeurIPS, 2017.