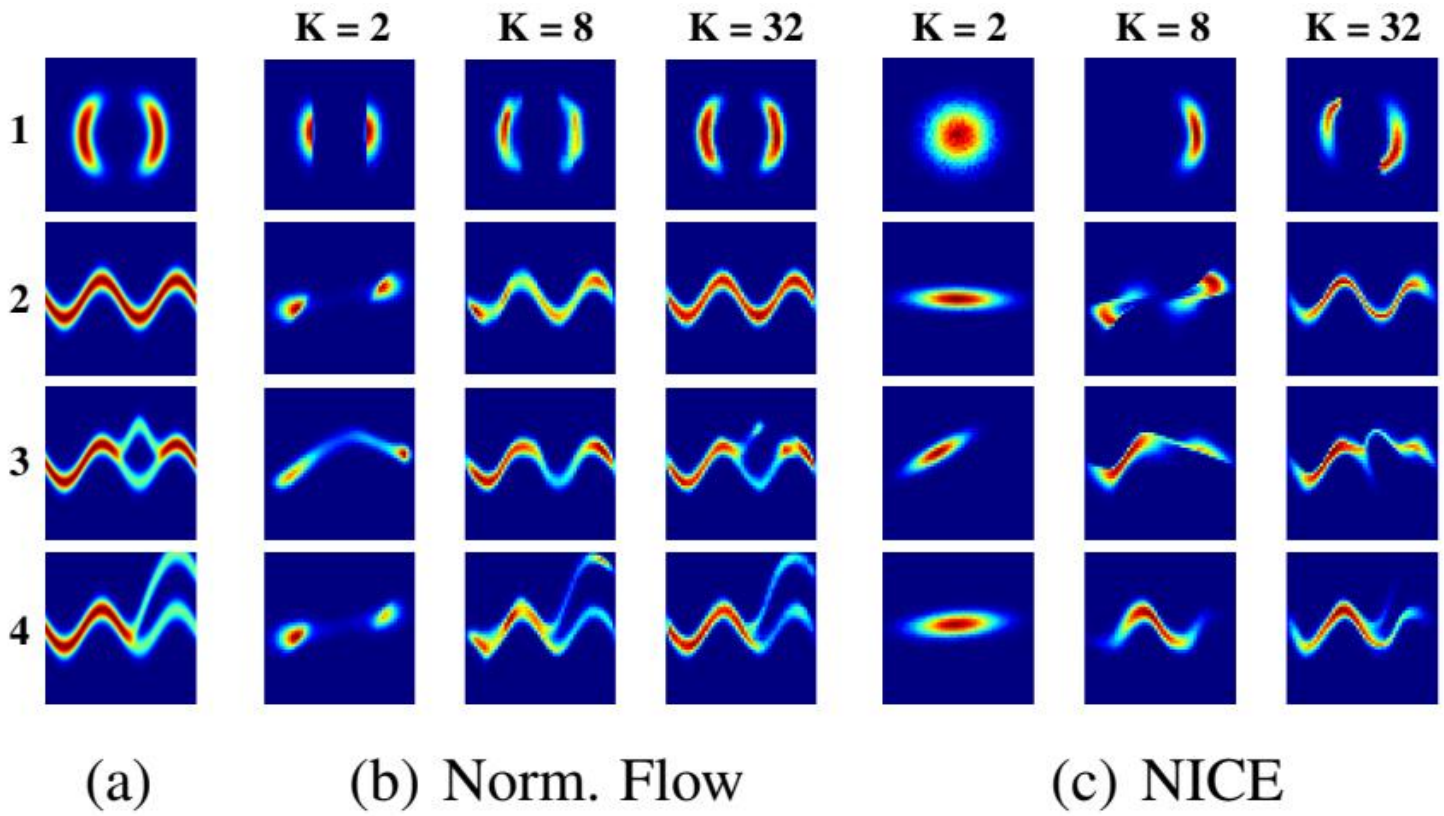
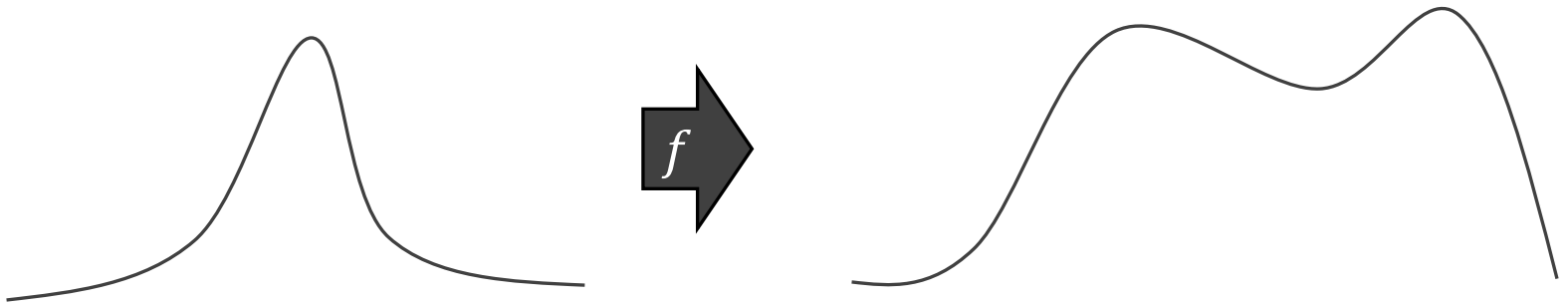
Normalizing flows

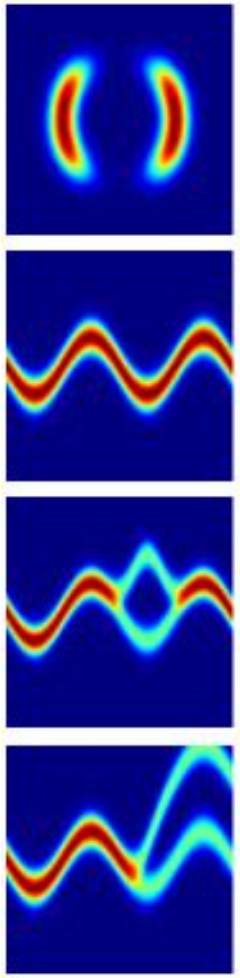


Intuition

- Often our posterior approximation is not enough
  - Imagine data generated by two modes

  → approximating with a standard Gaussian would be problematic
- If we applying a transformation to a simple density input
  - *e.g.*, a Gaussian
  - We can morph it into a more complicated density function
- Doing it many times → model any complex density





Change of variables

- For 1-d variables we know that

$$\int f(g(x))g'(x)dx = \int f(u)du \text{, where } u = g(x)$$

  - This is called change of variables (or integration by substitution)
  - The density is $du = g'(x)dx$

- For multivariate cases

$$\int f(g(\boldsymbol{x}))\left|\det\frac{dg}{d\boldsymbol{x}}\right|d\boldsymbol{x} = \int f(\boldsymbol{u})d\boldsymbol{u} \text{, where } \boldsymbol{u} = g(\boldsymbol{x})$$

And $\boldsymbol{u}$ and $\boldsymbol{x}$ have the same dimensionality

Normalizing flows

- Our model is an encoder $f: \boldsymbol{x} \rightarrow \boldsymbol{z}$
  - It maps the input $\boldsymbol{x}$ with density $p(\boldsymbol{x})$ to the latent $\boldsymbol{z}$ with density $p(\boldsymbol{z})$
- The inverse model is the decoder $f^{-1}: \boldsymbol{z} \rightarrow \boldsymbol{x}$
  - It maps the latent $\boldsymbol{z}$ back to the input $\boldsymbol{x}$
- For our forward model we have

$$p(\boldsymbol{z}) \left| \det \frac{df}{d\boldsymbol{x}} \right| = p(\boldsymbol{x}) \Leftrightarrow \log p(\boldsymbol{z}) + \log \left| \det \left(\frac{df}{d\boldsymbol{x}} \right) \right| = \log p(\boldsymbol{x})$$

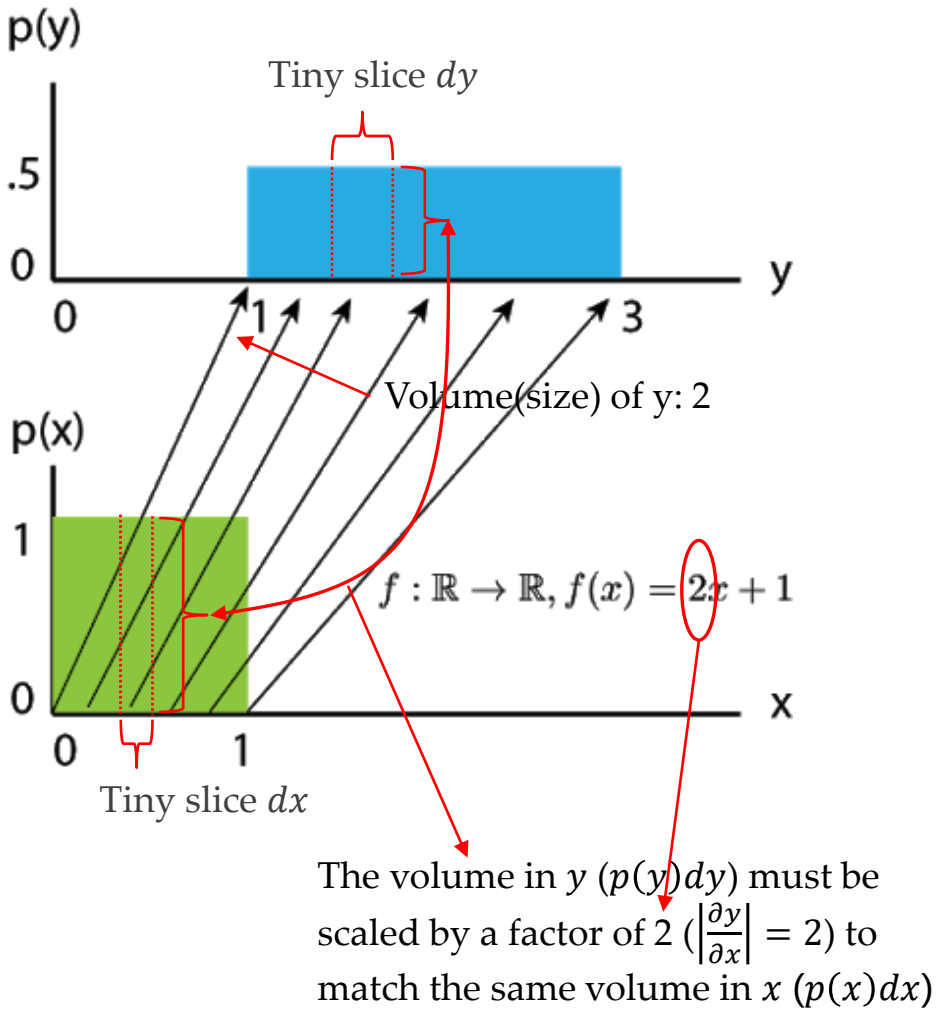
Rezende and Mohamed, Variational Inference with Normalizing Flows

Normalizing flows geometrically

- The determinant shows the change in the volume of the input and output probability spaces
- 1-d case

$$p(y)dy = p(x)dx \Rightarrow p(y)\left|\frac{\partial y}{\partial x}\right| = p(x)$$

  - $\left|\frac{dy}{dx}\right|$ indicates how much I must rescale a similarly tiny slice in y so the two densities are the same
  - The volumes (sizes) rescale so that $\int p(\boldsymbol{z}) = 1$ and $\int p(\boldsymbol{x}) = 1$
- Normalizing flows expand or contract the density



Stacking normalizing flows

- The change of variables can be applied recursively

$$\boldsymbol{x} = \boldsymbol{z}_K = f_K^{-1} \circ \cdots \circ f_2^{-1} \circ f_1^{-1}(\boldsymbol{z}_0)$$

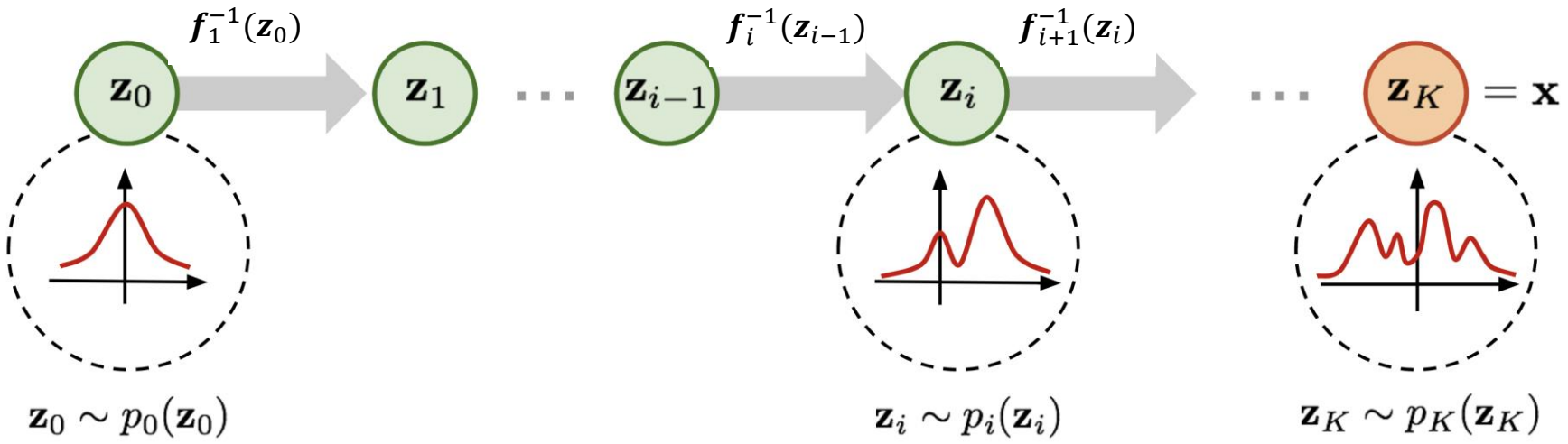
$$(e.g., \boldsymbol{z}_1 = f_1^{-1}(\boldsymbol{z}_0) \Leftrightarrow \boldsymbol{z}_0 = f_1(\boldsymbol{z}_1))$$

- The log density of our data is

$$\log p(\boldsymbol{x}) = \log p_K(\boldsymbol{z}_K) = \log p_0\,(\boldsymbol{z}_0) - \sum_{k=1}^{K} \log \det \left(\frac{df_k}{d\boldsymbol{x}_k} \right)$$

- Optimize with maximum likelihood

Stacking normalizing flows



What transformations?

$$\log p_K(\boldsymbol{z}_K) = \log p_0(\boldsymbol{z}_0) - \sum_{k=1}^{K} \log \det \left(\frac{df}{d\boldsymbol{x}} \right)$$

- We want smooth, differentiable transformations f_k
  - For which it with easy to compute inverse f_k^{-1}
  - and determinant of the Jacobian $\det \frac{df_k}{dz_k}$
- Example transformations
  - Planar flows
  - Radial flows
  - Coupling layers

Planar flow

- The transformation is

$$f(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}h(\boldsymbol{w}^T\boldsymbol{z} + b)$$

  - $\boldsymbol{u}, \boldsymbol{w}, b$ are free parameters
  - h is an element-wise non-linearity (element-wise so that it is easy to invert)
  - The log-determinant of the Jacobian is

$$\psi(\boldsymbol{z}) = h'\left(\boldsymbol{w}^T\boldsymbol{z} + b\right)\boldsymbol{w}$$

$$\det\left(\frac{df}{d\boldsymbol{z}}\right) = |1 + u^T\psi(\boldsymbol{z})|$$

Radial flow

- The transformation is

$$f(\boldsymbol{z}) = \boldsymbol{z} + \beta h(\alpha, r)(\boldsymbol{z} - \boldsymbol{z}_0)$$

Where $h(\alpha, r) = 1/(\alpha + r)$

- The log-determinant of the Jacobian is

$$\det\left(\frac{df}{d\boldsymbol{z}}\right) = [1 + \beta h(\alpha, r)]^{d-1}[1 + \beta h(\alpha, r) + h'(\alpha, r)r]$$

VAE with Normalizing Flows

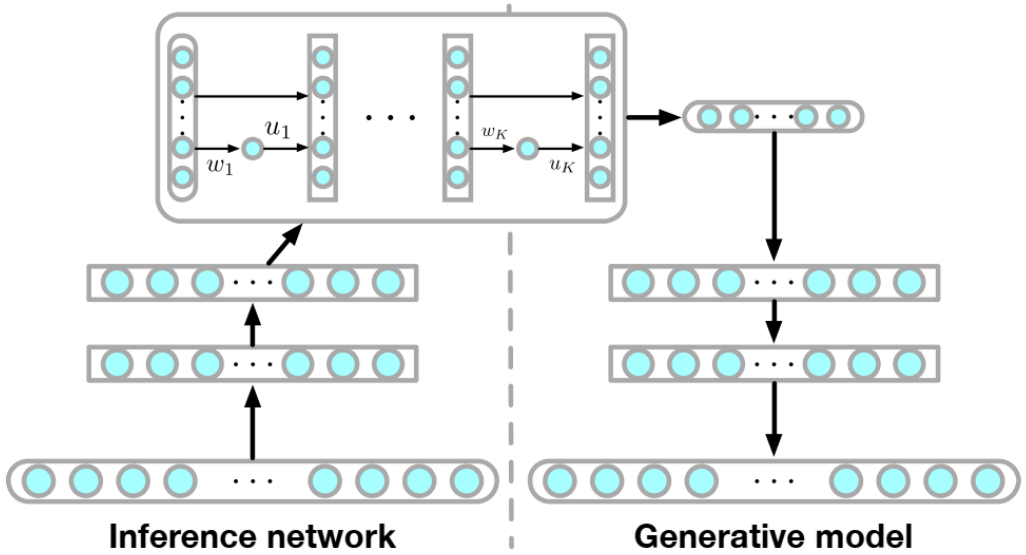


Figure 2. Inference and generative models. Left: Inference network maps the observations to the parameters of the flow; Right: generative model which receives the posterior samples from the inference network during training time. Round containers represent layers of stochastic variables whereas square containers represent deterministic layers.

Learning better posteriors with variational inference

- Again: the evidence lower bound is

$$\text{ELBO}_{\theta,\varphi}(\boldsymbol{x}) = \log p(\boldsymbol{x}) - \text{KL}(q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x}))$$

- Replace the simple approximate posterior by normalizing flows

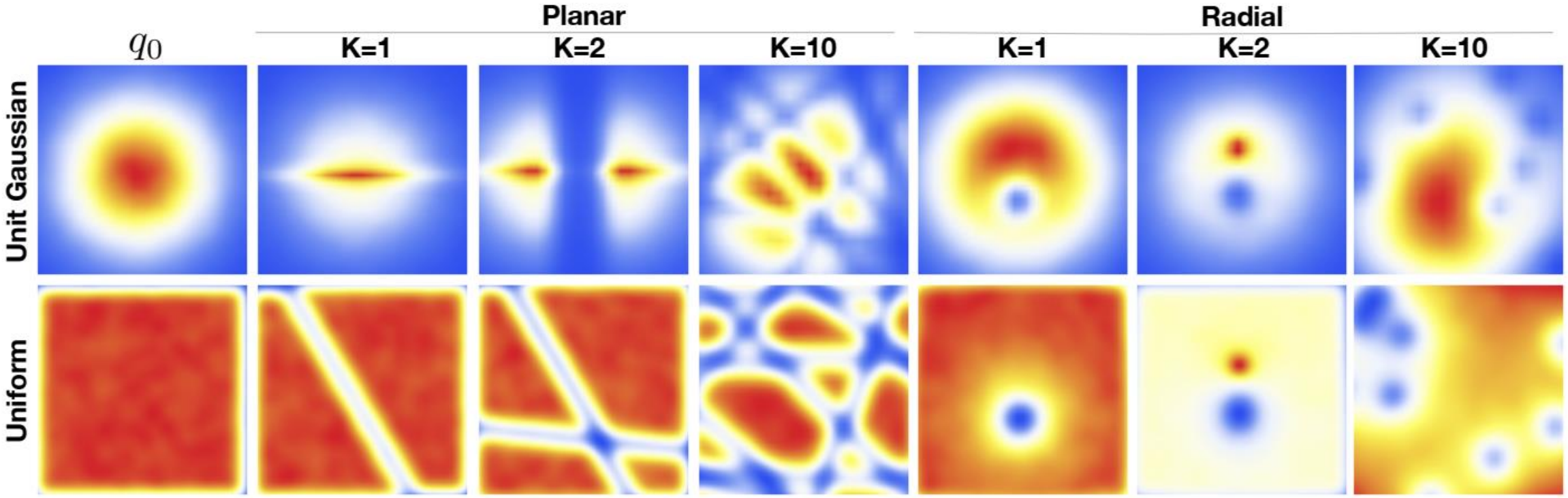
$$\mathbb{E}_{q_0(\boldsymbol{z}_0|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_K)] - \text{KL}(q_0(\boldsymbol{z}_0|\boldsymbol{x}) \parallel p(\boldsymbol{z})) + \mathbb{E}_{q_0(\boldsymbol{z}_0|\boldsymbol{x})}\left[\sum_{k=1}^{K} \log \left|\det \frac{df_k}{d\boldsymbol{z}_k}\right|^{-1}\right]$$

Algorithm 1 Variational Inf. with Normalizing Flows

Parameters: ϕ variational, $\boldsymbol{\theta}$ generative
while not converged **do**
  $\mathbf{x} \leftarrow$ {Get mini-batch}
  $\mathbf{z}_0 \sim q_0(\bullet|\mathbf{x})$
  $\mathbf{z}_K \leftarrow f_K \circ f_{K-1} \circ \ldots \circ f_1(\mathbf{z}_0)$
  $\mathcal{F}(\mathbf{x}) \approx \mathcal{F}(\mathbf{x}, \mathbf{z}_K)$
  $\Delta\boldsymbol{\theta} \propto -\nabla_{\theta}\mathcal{F}(\mathbf{x})$
  $\Delta\boldsymbol{\phi} \propto -\nabla_{\phi}\mathcal{F}(\mathbf{x})$
end while

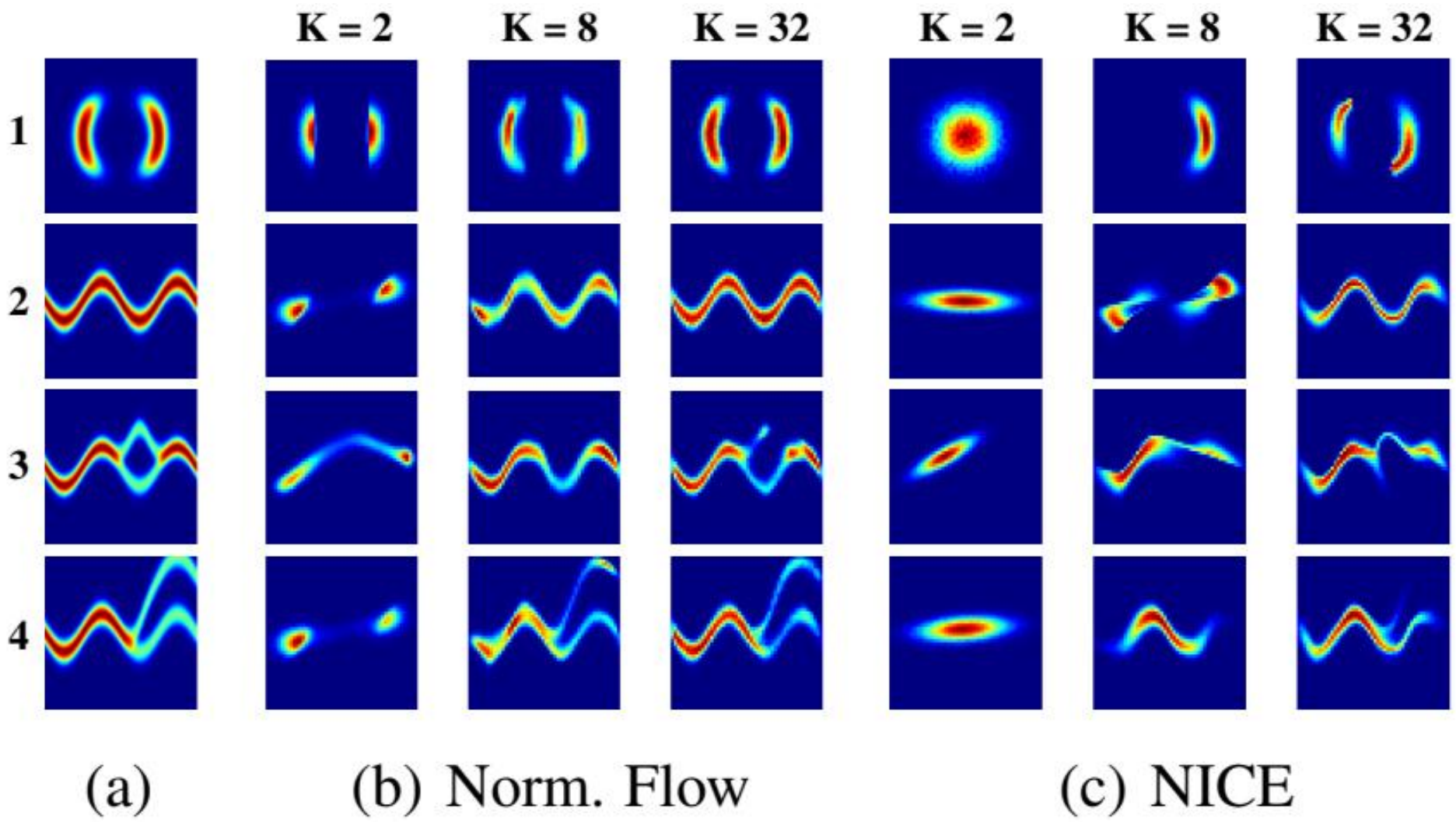
Rezende and Mohamed, Variational Inference with Normalizing Flows

The effect of number of transformations/flows



Rezende and Mohamed, Variational Inference with Normalizing Flows

Some results



Rezende and Mohamed, Variational Inference with Normalizing Flows