BRUNO: A Deep Recurrent Model for Exchangeable Data

Iryna Korshunova

Ghent University (visiting Gatsby Unit, now at Twitter)

Jonas Degrave

Ghent University (now at DeepMind)

Ferenc Huszár

Twitter

Yarin Gal

University of Oxford

Arthur Gretton

Gatsby Unit, UCL

Joni Dambre

Ghent University

Exchangeability

A stochastic process $x_1, x_2, x_3 \ldots$ is exchangeable if for all n and all permutations π:

$$p(x_1, \ldots, x_n) = p\left(x_{\pi(1)}, \ldots, x_{\pi(n)}\right)$$

Exchangeability

A stochastic process $x_1, x_2, x_3 \ldots$ is exchangeable if for all n and all permutations π:

$$p(x_1, \ldots, x_n) = p\left(x_{\pi(1)}, \ldots, x_{\pi(n)}\right)$$

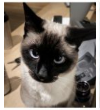
P( )= P( )

Exchangeability

A stochastic process $x_1, x_2, x_3 \ldots$ is exchangeable if for all n and all permutations π:

$$p(x_1, \ldots, x_n) = p\left(x_{\pi(1)}, \ldots, x_{\pi(n)}\right)$$

P( )= P( )

P(  )=P(  )=P(  ) = ...

...

Exchangeability. IID example

A stochastic process $x_1, x_2, x_3 \ldots$ is exchangeable if for all n and all permutations π:

$$p(x_1, \ldots, x_n) = p\left(x_{\pi(1)}, \ldots, x_{\pi(n)}\right)$$

IID random variables are exchangeable:

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i)$$

Exchangeability. Non-IID example

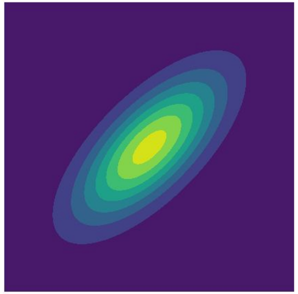
$$p(x_1, \dots, x_n) = \mathcal{N}_n(\mu, \mathbf{\Sigma})$$

$$\mu = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix} \qquad \mathbf{\Sigma} = \begin{bmatrix} v & \rho & \rho & \dots & \rho \\ \rho & v & \rho & \dots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \dots & v \end{bmatrix} \quad 0 \le \rho < v$$

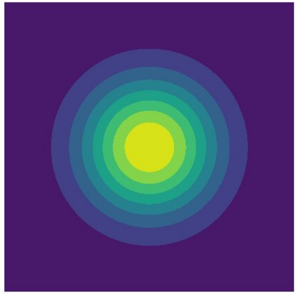
compound symmetry
covariance structure

$v = 1 \quad \rho = 0.7$

exchangeable

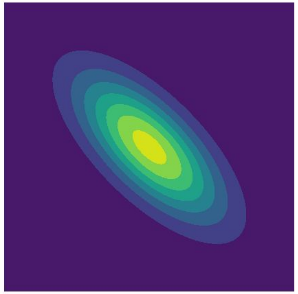


i.i.d



$v = 1 \quad \rho = 0$

not exchangeable



$v = 1 \quad \rho = -0.7$

Exchangeability and Bayesian computations

De Finetti's theorem says that every exchangeable process is a mixture of i.i.d. processes:

$$p(x_1, \ldots, x_n) = \int p(\theta) \prod_{i=1}^{n} p(x_i|\theta) d\theta$$

where θ is some parameter conditioned on which the data is i.i.d.

De Finetti's theorem. Example

Assume we have a process, where $x_1, \ldots, x_n \sim \mathcal{N}_n(0, \mathbf{\Sigma})$

with an exchangeable covariance structure:

$$\mathbf{\Sigma} = \begin{bmatrix} v & \rho & \rho & \ldots & \rho \\ \rho & v & \rho & \ldots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \ldots & v \end{bmatrix} \quad 0 \leq \rho < v$$

De Finetti's theorem. Example

Assume we have a process, where $x_1, \ldots, x_n \sim \mathcal{N}_n(0, \mathbf{\Sigma})$

with an exchangeable covariance structure:

$$\mathbf{\Sigma} = \begin{bmatrix} v & \rho & \rho & \ldots & \rho \\ \rho & v & \rho & \ldots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \ldots & v \end{bmatrix} \quad 0 \leq \rho < v$$

Then $x_1, \ldots x_n$ are i.i.d

with $x_i \sim \mathcal{N}(\theta, v - \rho)$ conditionally on $\theta \sim \mathcal{N}(0, \rho)$

De Finetti's theorem. Why?

Bayesian

$$x_1, x_2, x_3, \ldots$$

successive coin tosses

De Finetti's theorem. Why?

Bayesian

$x_1, x_2, x_3, \ldots$

successive coin tosses

If we assume $x_1, x_2, x_3, \ldots$ are iid:

De Finetti's theorem. Why?

Bayesian

$$x_1, x_2, x_3, \ldots$$

successive coin tosses

If we assume $x_1, x_2, x_3, \ldots$ are iid:

$$p(x_n | x_{1:n-1}) = p(x_n)$$

=> results of the first *n-1* tosses do not change the uncertainty about the result of *n*-th tosses

Exchangeability and Bayesian computations #2

$$p(x_1, \ldots, x_n) = \int p(\theta) \prod_{i=1}^{n} p(x_i|\theta) d\theta$$

rewrite in terms of predictive distributions

$$p(x_n|x_{1:n-1}) = \int \underbrace{p(x_n|\theta)}_{\text{likelihood}} \underbrace{p(\theta|x_{1:n})}_{\text{posterior}} d\theta$$

Exchangeability and Bayesian computations #3

$$p(x_n|x_{1:n-1}) = \int \underbrace{p(x_n|\theta)}_{\text{likelihood}} \underbrace{p(\theta|x_{1:n})}_{\text{posterior}} d\theta$$

Gives 2 ways for defining models of exchangeable sequences

Exchangeability and Bayesian computations #3

$$p(x_n|x_{1:n-1}) = \int \underbrace{p(x_n|\theta)}_{\text{likelihood}} \underbrace{p(\theta|x_{1:n})}_{\text{posterior}} d\theta$$

Gives 2 ways for defining models of exchangeable sequences:

1. via explicit Bayesian modelling => **VAE-based models**

Exchangeability and Bayesian computations #3

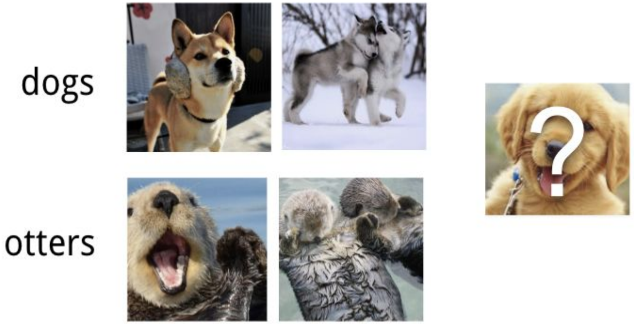
$$p(x_n|x_{1:n-1}) = \int \underbrace{p(x_n|\theta)}_{\text{likelihood}} \underbrace{p(\theta|x_{1:n})}_{\text{posterior}} d\theta$$

Gives 2 ways for defining models of exchangeable sequences:

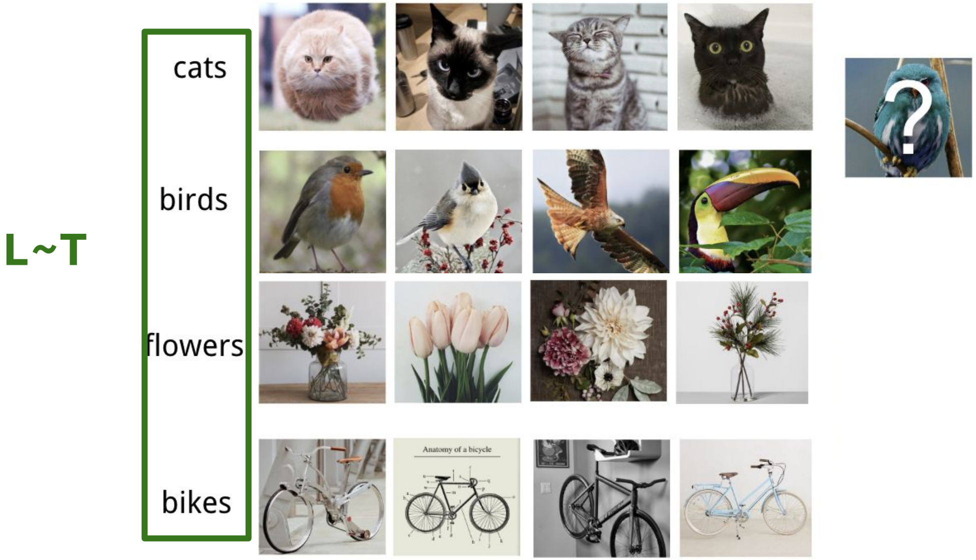
1. via explicit Bayesian modelling => **VAE-based models**

2. via exchangeable processes => **BRUNO**

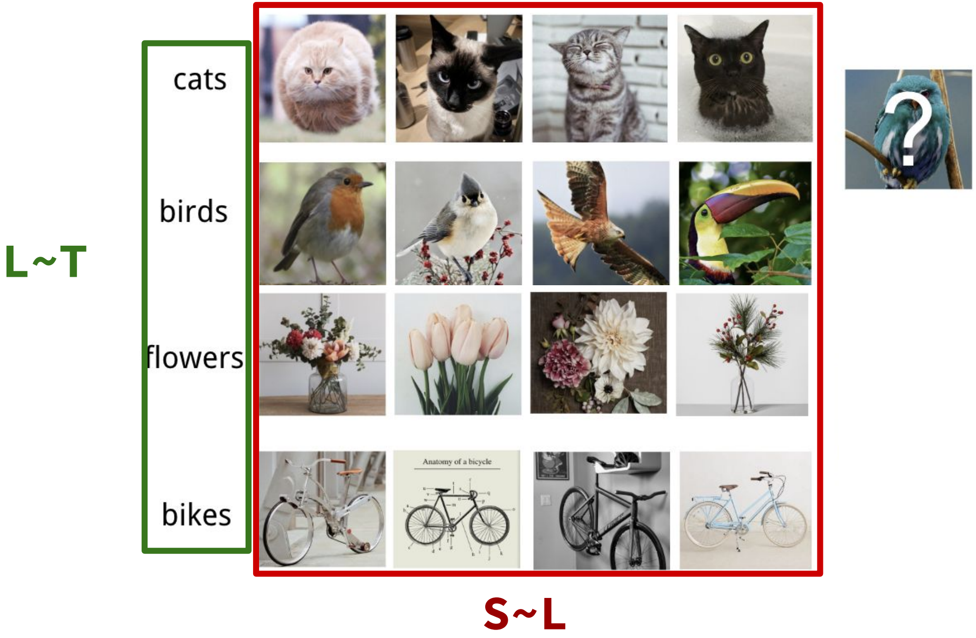
A killer application: meta-learning



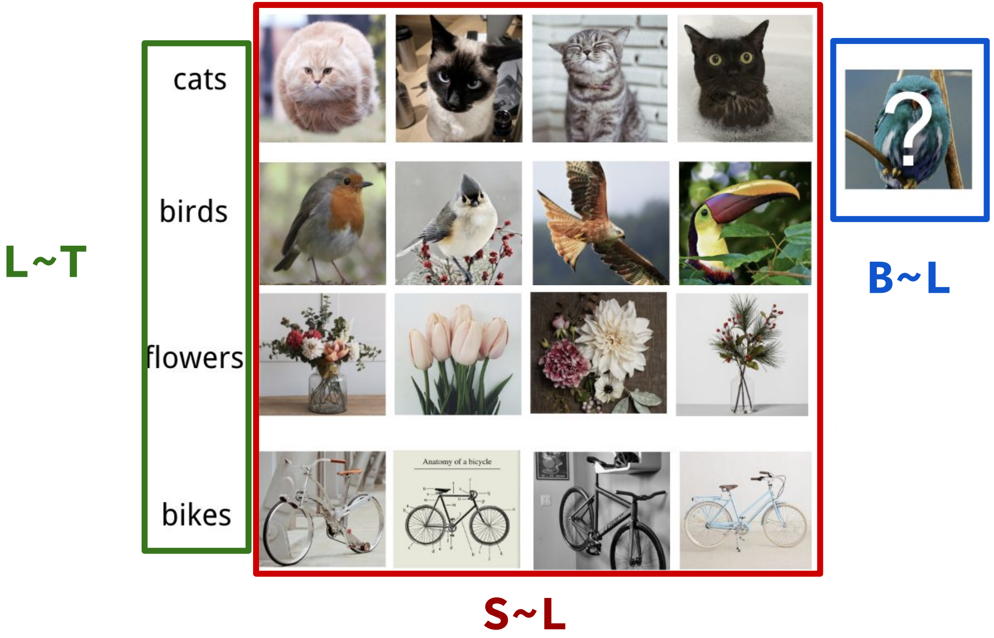
Meta-learning. Episode



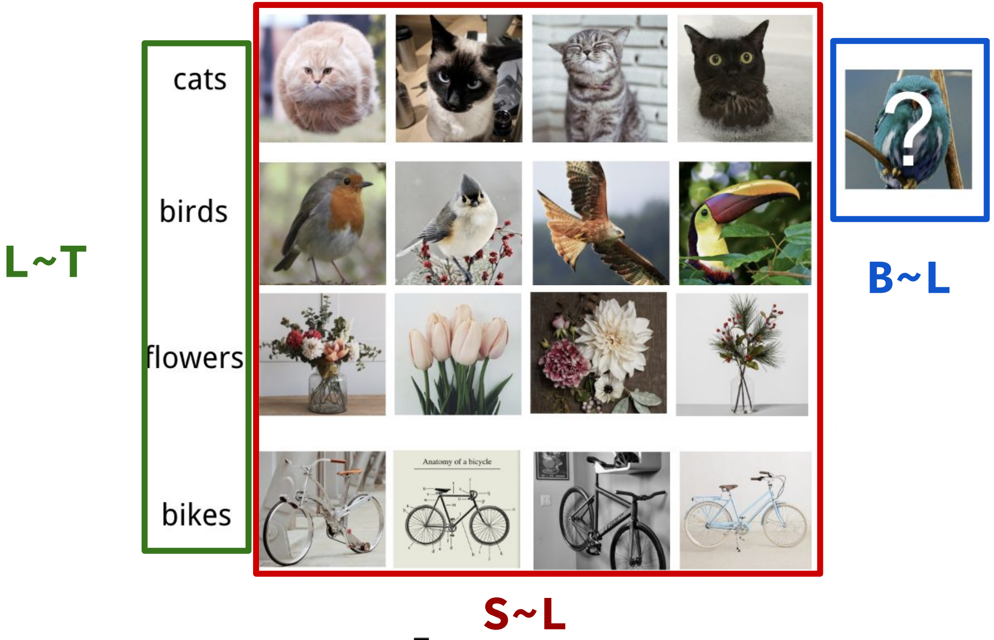
Meta-learning. Episode



Meta-learning. Episode

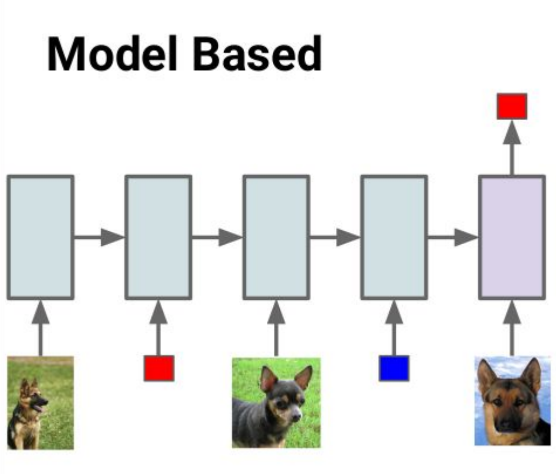


Meta-learning. Episode



$$\theta^* = \arg\max_\theta \mathbb{E}_{L\sim T}\left[\mathbb{E}_{S\sim L, B\sim L}\left[\sum_{(x,y)\in B} \log p_\theta(y|x, S)\right]\right]$$

Meta-learning models. Taxonomy

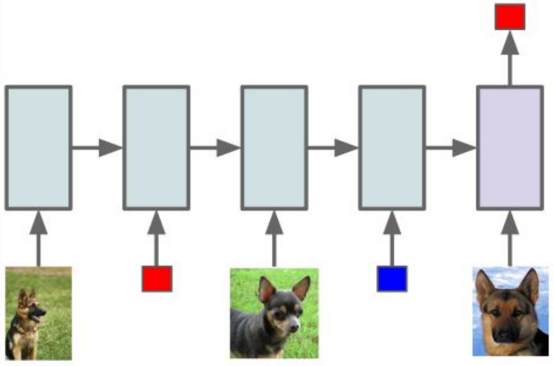


$$p_\theta(y|x, S) = f_\theta(x, S)$$

Memory-Augmented Neural Network
Neural Processes
BRUNO

Meta-learning models. Taxonomy

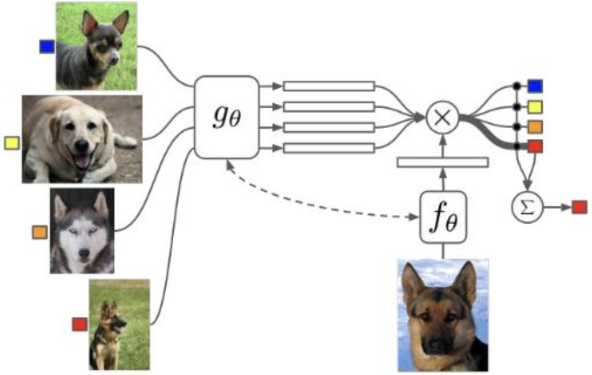
Model Based



$$p_\theta(y|x,S) = f_\theta(x,S)$$

Memory-Augmented Neural Network
Neural Processes
BRUNO

Metric Based

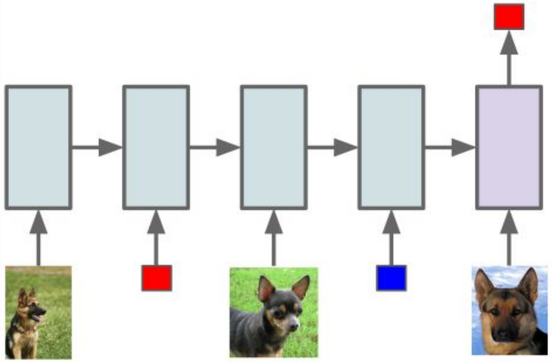


$$p_\theta(y|x,S) = \sum_{(x_i,y_i)\in S} k(x_i,x)y_i$$

Siamese Neural Networks
Matching Networks
Relation Network
Prototypical Networks

Meta-learning models. Taxonomy

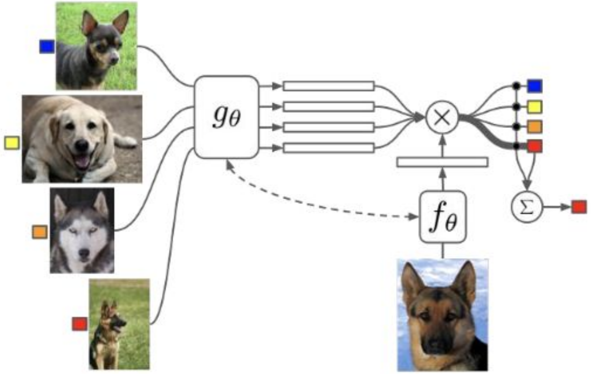
Model Based



$$p_\theta(y|x,S) = f_\theta(x,S)$$

Memory-Augmented Neural Network
Neural Processes
BRUNO

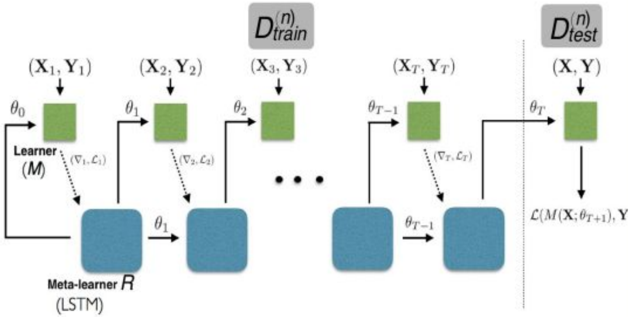
Metric Based



$$p_\theta(y|x,S) = \sum_{(x_i,y_i)\in S} k(x_i,x)y_i$$

Siamese Neural Networks
Matching Networks
Relation Network
Prototypical Networks

Optimization Based



$$p_\theta(y|x,S) = f_{\theta(S)}(x,S)$$
$$\theta(S) = g_\phi(\theta_0, \{\nabla_{\theta_0} L(x_i,y_i)\}_{(x_i,y_i\in S)})$$

LSTM meta-learner
Model-agnostic meta-learning (MAML)

O. Vinyals Meta-learning symposium talk @ NIPS'17

Exchangeability and meta-learning

1. Order-invariance

Exchangeability and meta-learning

1. Order-invariance

2. Correlation

$\{D_1, D_2, ..., D_K\}$ are i.i.d. $\{x_1, x_2, ..., x_n\}$ are correlated

Simple

$$p(x_1, \ldots, x_n) = \mathcal{N}_n(\mu, \mathbf{\Sigma})$$

$$\mu = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix} \quad \mathbf{\Sigma} = \begin{bmatrix} v & \rho & \rho & \ldots & \rho \\ \rho & v & \rho & \ldots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \ldots & v \end{bmatrix} \quad 0 \leq \rho < v$$

Difficult

x_1 x_2 x_3 x_4

$$p(x_1, \ldots, x_n) = ?$$

Simple

Difficult

bijection

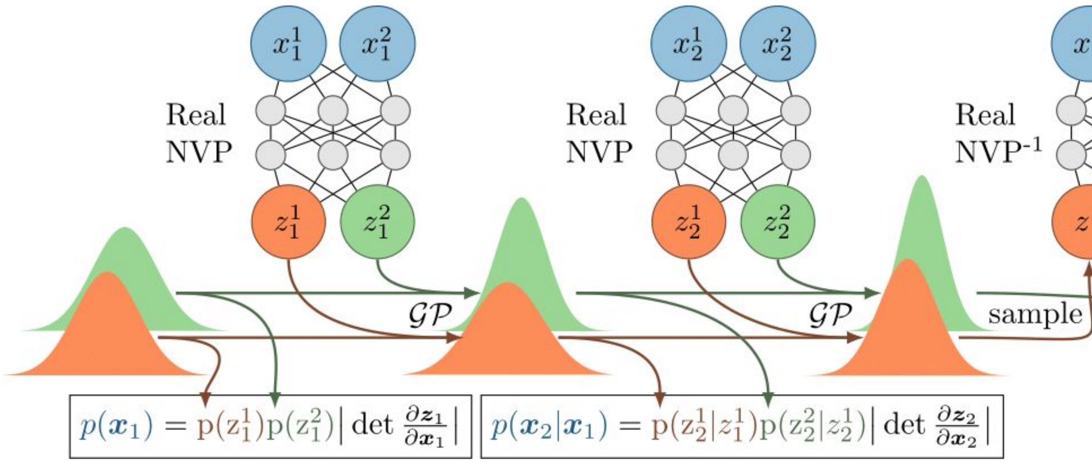
x_1 x_2 x_3 x_4

$$p(x_1, \ldots, x_n) = \mathcal{N}_n(\mu, \mathbf{\Sigma})$$

$$\mu = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix} \quad \mathbf{\Sigma} = \begin{bmatrix} v & \rho & \rho & \ldots & \rho \\ \rho & v & \rho & \ldots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \ldots & v \end{bmatrix} \quad 0 \leq \rho < v$$

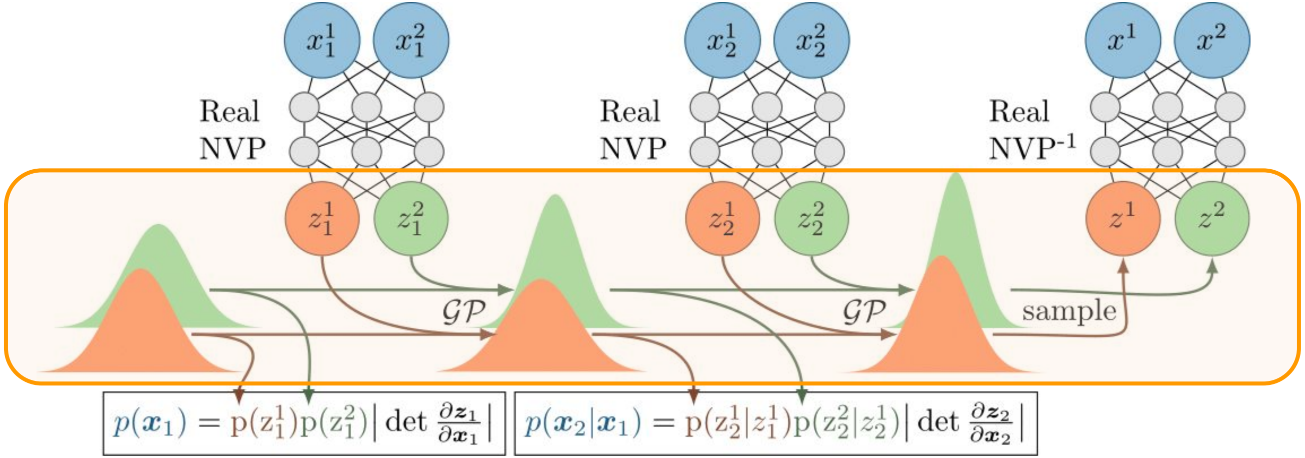
$$p(x_1, \ldots, x_n) = ?$$

BRUNO: Bayesian RecUrrent Neural mOdel



BRUNO: Bayesian RecUrrent Neural mOdel

Process in the latent space



Exchangeability

A stochastic process $x_1, x_2, x_3 \ldots$ is exchangeable if for all n and all permutations π:

$$p(x_1, \ldots, x_n) = p\left(x_{\pi(1)}, \ldots, x_{\pi(n)}\right)$$

Exchangeability

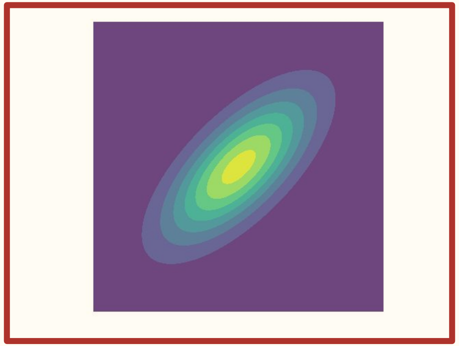
A stochastic process $x_1, x_2, x_3 \ldots$ is exchangeable if for all n and all permutations π:

$$p(x_1, \ldots, x_n) = p\left(x_{\pi(1)}, \ldots, x_{\pi(n)}\right)$$

$$p(x_1, \ldots, x_n) = \mathcal{N}_n(\mu, \boldsymbol{\Sigma})$$

$$\mu = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} v & \rho & \rho & \ldots & \rho \\ \rho & v & \rho & \ldots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \ldots & v \end{bmatrix} \quad 0 \leq \rho < v$$

$v = 1 \quad \rho = 0.7$



Exchangeability

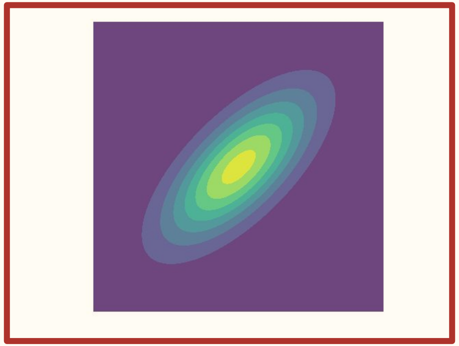
A stochastic process $x_1, x_2, x_3 \ldots$ is exchangeable if for all n and all permutations π:

$$p(x_1, \ldots, x_n) = p\left(x_{\pi(1)}, \ldots, x_{\pi(n)}\right)$$

$$p(x_1, \ldots, x_n) = \mathcal{N}_n(\mu, \boldsymbol{\Sigma})$$

$$\mu = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} v & \rho & \rho & \ldots & \rho \\ \rho & v & \rho & \ldots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \ldots & v \end{bmatrix} \quad 0 \leq \rho < v$$

$v = 1 \quad \rho = 0.7$



Defines an exchangeable Gaussian process

Gaussian processes

Definition. f is a Gaussian process on $\mathcal{X}$ with

mean function $\Phi : \mathcal{X} \mapsto \mathbb{R}$

kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$

if any finite collection of function values have a joint multivariate Gaussian distribution, i.e. $(f(x_1), ..., f(x_n)) \sim \mathcal{N}_n(\mu, \mathbf{\Sigma})$ where

$\mu \in \mathbb{R}^n$ with $\mu_i = \Phi(x_i)$

$\mathbf{\Sigma} \in \Pi(n)$ with $\Sigma_{ij} = k(x_i, x_j)$

Exchangeable Gaussian processes

Definition. f is a Gaussian process on $\mathcal{X}$ with

mean function $\Phi : \mathcal{X} \mapsto \mathbb{R}$

kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$

if any finite collection of function values have a joint multivariate Gaussian distribution, i.e. $(f(x_1), ..., f(x_n)) \sim \mathcal{N}_n(\mu, \mathbf{\Sigma})$ where

$\mu \in \mathbb{R}^n$ with $\mu_i = \Phi\cancel{(x_i)}$

$\mathbf{\Sigma} \in \Pi(n)$ with $\Sigma_{ij} = k(\cancel{x_i, x_j})$

$$\mu = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix} \qquad \mathbf{\Sigma} = \begin{bmatrix} v & \rho & \rho & \dots & \rho \\ \rho & v & \rho & \dots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \dots & v \end{bmatrix} \quad 0 \leq \rho < v$$

Exchangeable Gaussian processes

$$(z_1, \dots z_n) \sim \mathcal{N}_n(\mu, \mathbf{\Sigma}) \qquad \mu = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix} \qquad \mathbf{\Sigma} = \begin{bmatrix} v & \rho & \rho & \dots & \rho \\ \rho & v & \rho & \dots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \dots & v \end{bmatrix} \quad 0 \leq \rho < v$$

Predictive distribution?

$$p(z_{n+1} | z_{1:n}) = \mathcal{N}(\mu_{n+1}, v_{n+1})$$

Exchangeable Gaussian processes

$$(z_1, \ldots z_n) \sim \mathcal{N}_n(\mu, \mathbf{\Sigma}) \qquad \mu = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix} \qquad \mathbf{\Sigma} = \begin{bmatrix} v & \rho & \rho & \ldots & \rho \\ \rho & v & \rho & \ldots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \ldots & v \end{bmatrix}_{0 \leq \rho < v}$$

Predictive distribution?

$$p(z_{n+1}|z_{1:n}) = \mathcal{N}(\mu_{n+1}, v_{n+1})$$

Exchangeable Gaussian processes

$$(z_1, \dots z_n) \sim \mathcal{N}_n(\mu, \mathbf{\Sigma}) \qquad \mu = \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix} \qquad \mathbf{\Sigma} = \begin{bmatrix} v & \rho & \rho & \dots & \rho \\ \rho & v & \rho & \dots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \dots & v \end{bmatrix}_{0 \le \rho < v}$$

Predictive distribution?

$$p(z_{n+1} | z_{1:n}) = \mathcal{N}(\mu_{n+1}, v_{n+1})$$

$$\mu_{n+1} = (1 - d_n)\mu_n + d_n z_n$$
$$v_{n+1} = (1 - d_n)v_n + d_n(v - \rho)$$

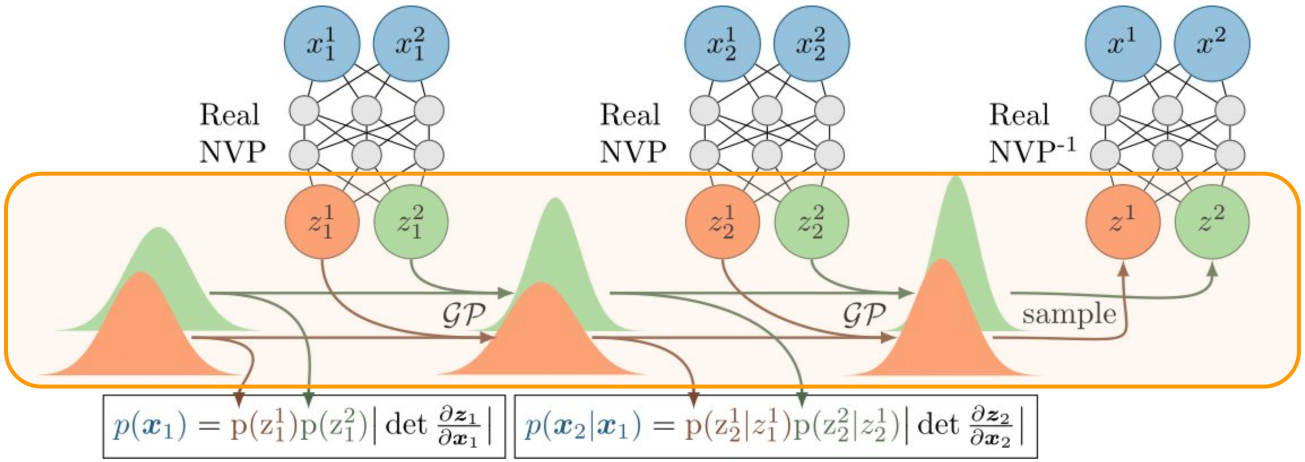
$$d_n = \frac{\rho}{v + \rho(n-1)}$$

$$\mu_1 = \mu, \quad v_1 = v$$

recurrence

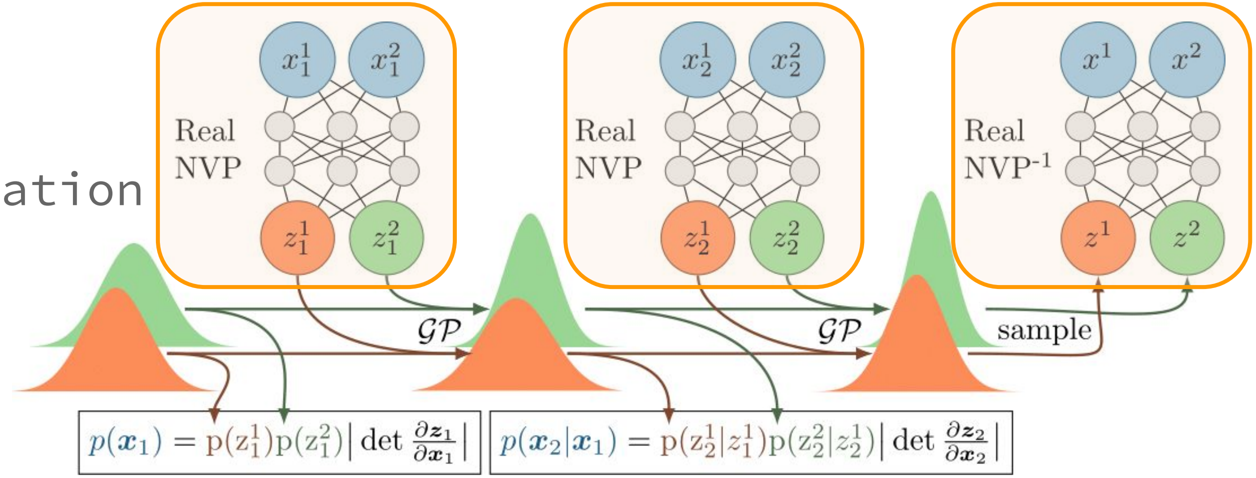
BRUNO

Process in the latent space



BRUNO

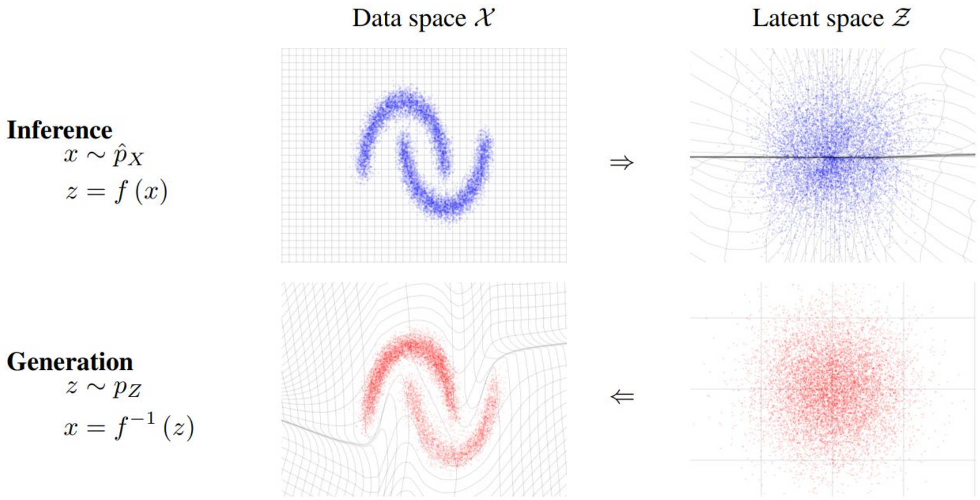
Powerful bijective transformation



Real NVP

$$f : \mathcal{X} \mapsto \mathcal{Z} \text{ with } \mathcal{X} = \mathbb{R}^D$$

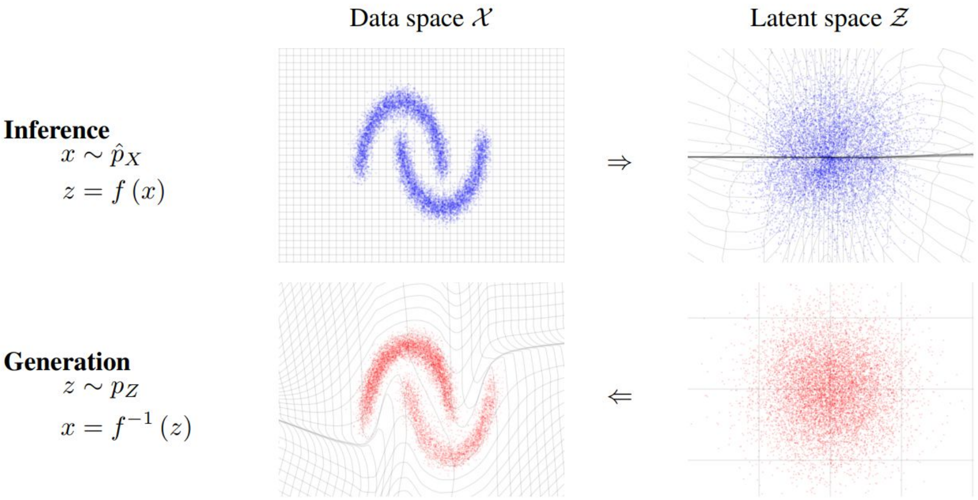
- bijective
- forward and the inverse mappings are equally expensive
- computing the log determinant of the Jacobian is $O(D)$



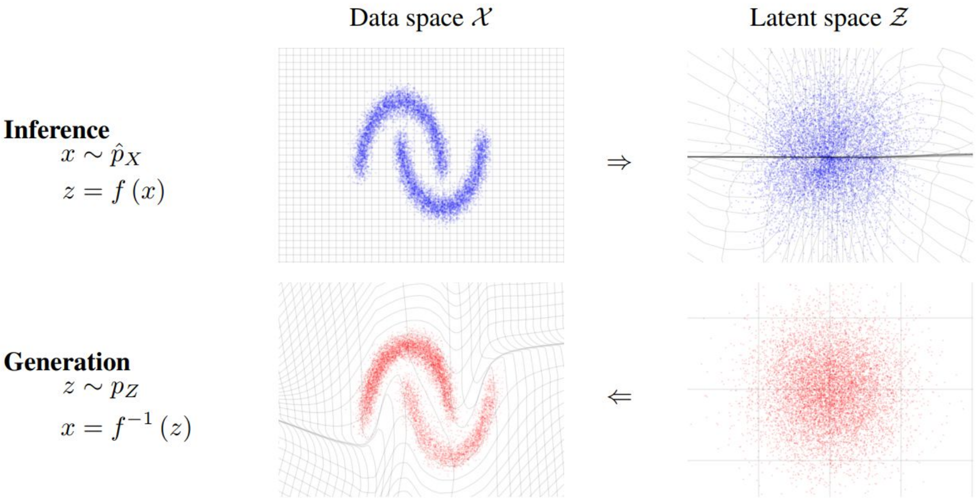
Real NVP

$$f : \mathcal{X} \mapsto \mathcal{Z} \text{ with } \mathcal{X} = \mathbb{R}^D$$

- bijective
- forward and the inverse mappings are equally expensive
- computing the log determinant of the Jacobian is *O(D)*



Real NVP. Change of Variables

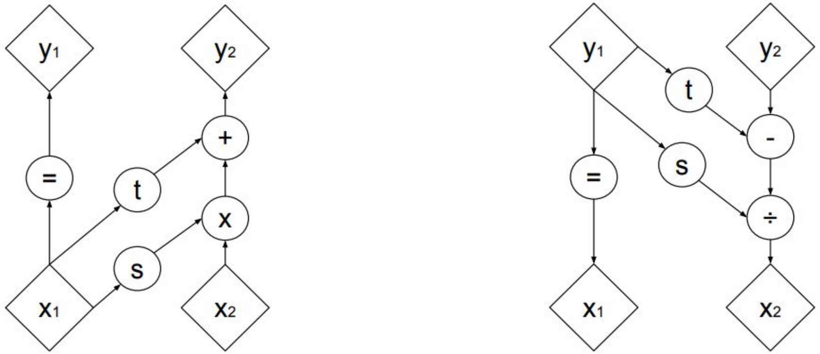


Likelihood evaluation:

$$p(\mathbf{x}) = p(\mathbf{z}) \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

Real NVP's coupling layer

$$
\begin{aligned}
y_{1:d} &= x_{1:d} \\
y_{d+1:D} &= x_{d+1:D} \odot \exp\big(s(x_{1:d})\big) + t(x_{1:d})
\end{aligned}
$$



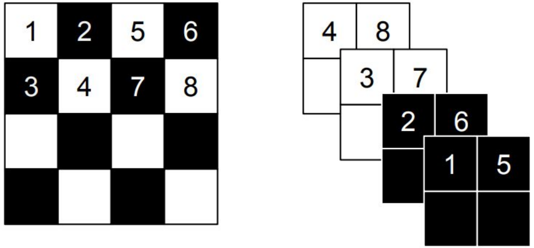
(a) Forward propagation (b) Inverse propagation

Jacobian:

$$
\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \operatorname{diag}\big(\exp\left[s\left(x_{1:d}\right)\right]\big) \end{bmatrix}
$$

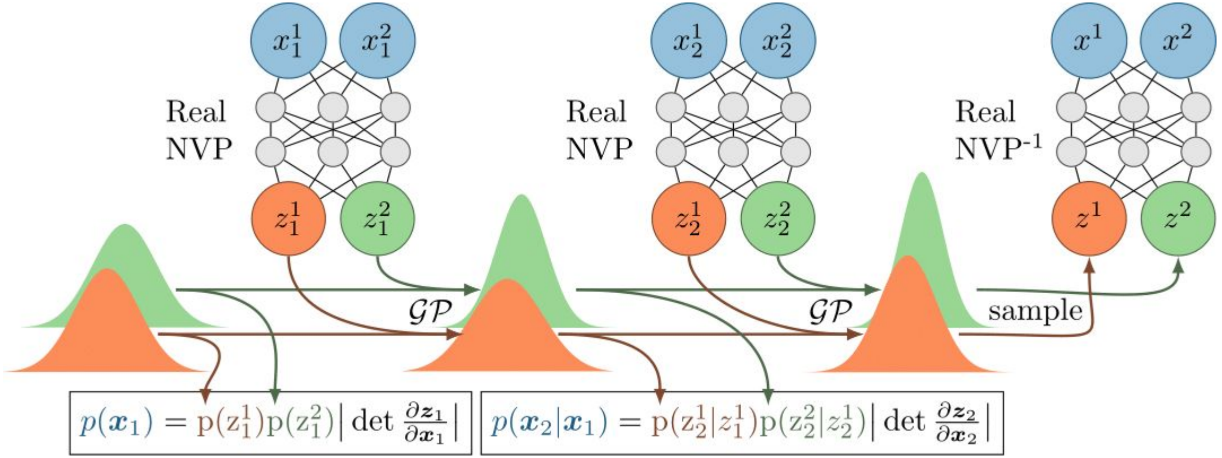
Convolutional Real NVP

Schemes to partition the dimensions when using convolutions



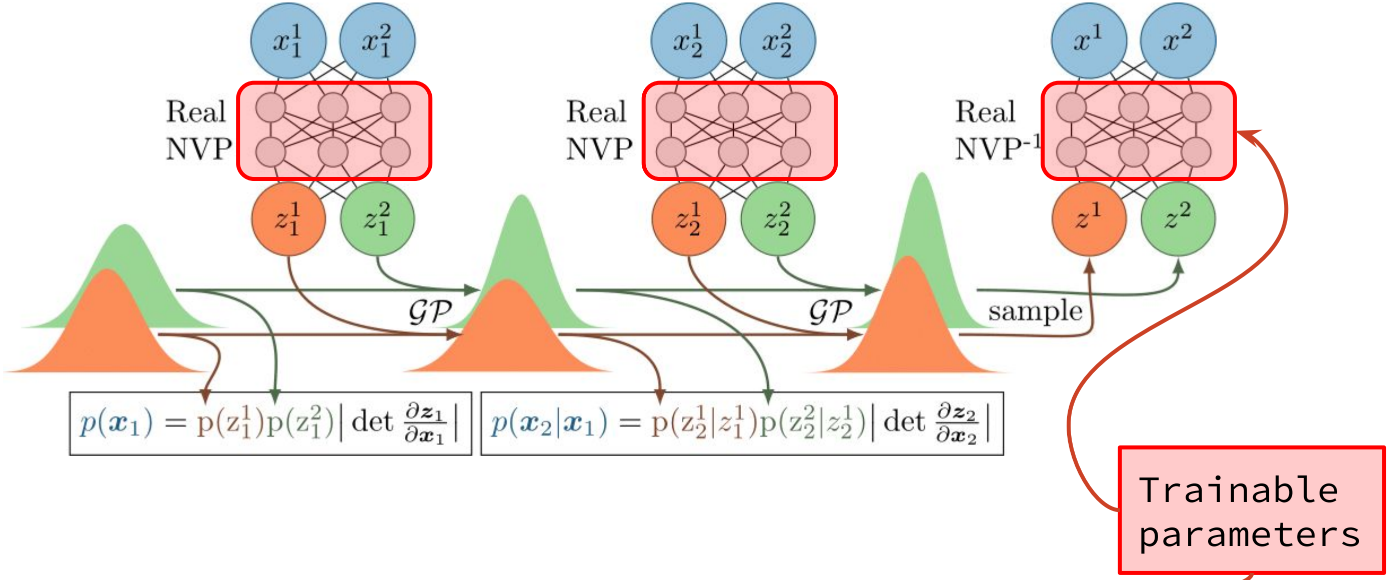
Samples from Real NVP trained on CelebA and LSUN

BRUNO



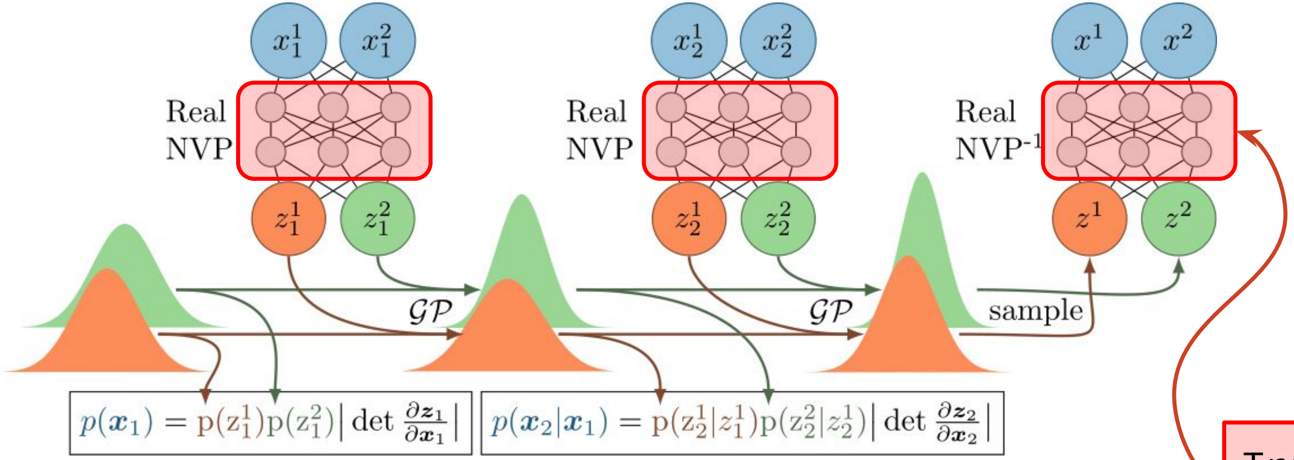
* Latent dimensions are independent, so $p(\mathbf{z}) = \prod_{d=1}^{D} p(z^d)$
* For every latent dimension *d:* $(z_1^d, \ldots, z_n^d) \sim \mathcal{N}_n(\mu^d \mathbf{1}, \mathbf{K}^d)$ with an exchangeable $\mathbf{K}^{\mathrm{d}}$

Training BRUNO



- $p(\mathbf{z}) = \prod_{d=1}^{D} p(z^d)$
- $(z_1^d, \ldots, z_n^d) \sim \mathcal{N}_n(\mu^d \mathbf{1}, \mathbf{K}^d)$ with $\mathbf{K}_{ij}^d = \begin{cases} v^d, & i = j \\ \rho^d, & i \neq j \end{cases}$

Training BRUNO



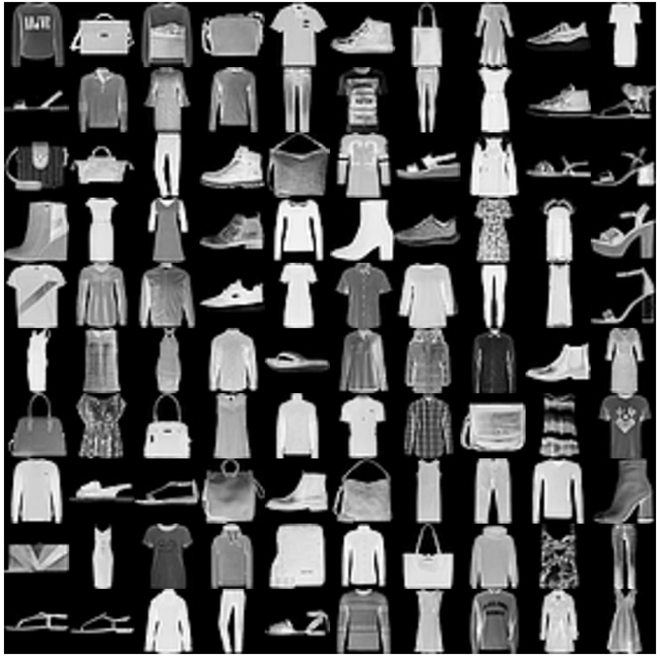
Trainable parameters

$$\mathbf{K}_{ij}^{d} = \begin{cases} v^d, & i = j \\ \rho^d, & i \neq j \end{cases}$$

Classical RNN objective:

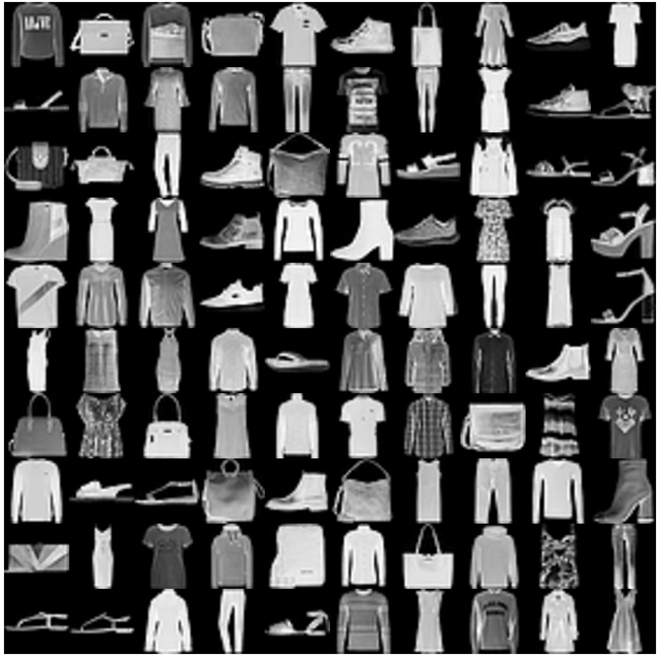
$$\mathcal{L} = \sum_{n=0}^{N-1} \log p(\mathbf{x}_{n+1}|\mathbf{x}_{1:n})$$

Experiments: Fashion MNIST

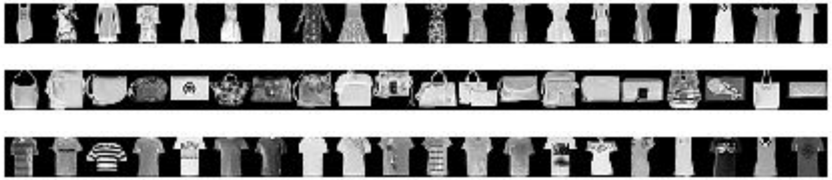


Random samples from the dataset

Experiments: Fashion MNIST

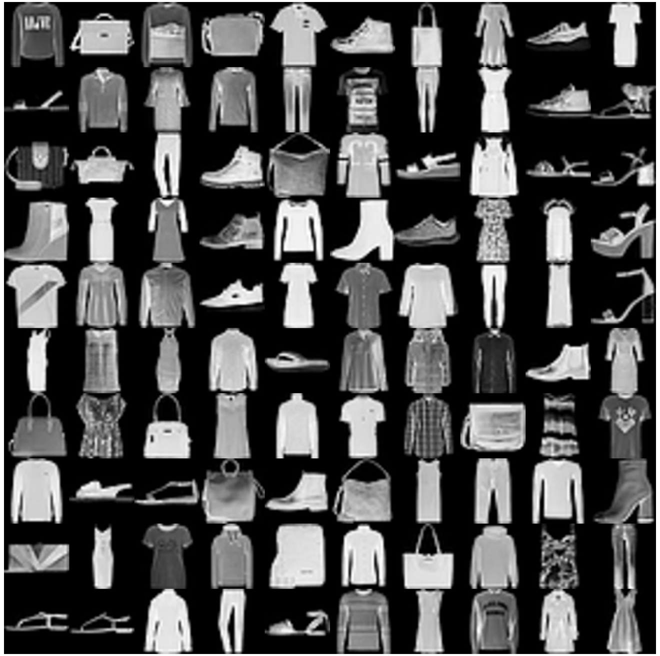


Random samples from the dataset

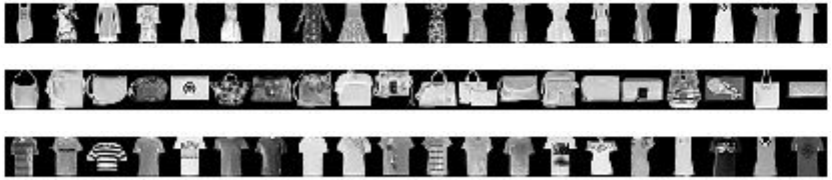


Training sequences

Experiments: Fashion MNIST



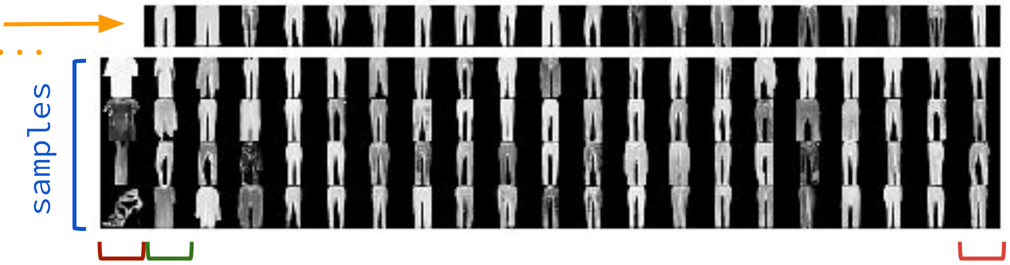
Random samples from the dataset

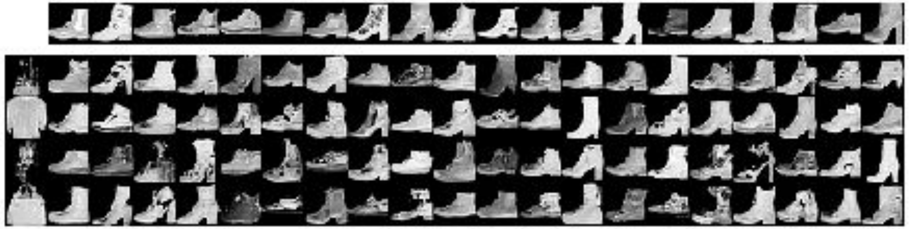


Training sequences

BRUNO samples from
the prior p(x)

Experiments: Fashion MNIST





Learning to correlate

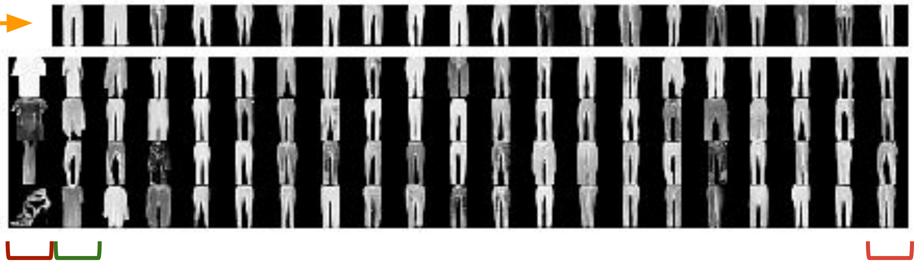
28x28 inputs -> 784 latent dimensions
with own variances and covariances:

$$\Sigma = \begin{bmatrix} v & \rho & \rho & \dots & \rho \\ \rho & v & \rho & \dots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \dots & v \end{bmatrix} \quad 0 \leq \rho < v$$

Learning to correlate

conditioning inputs $x_1, x_2, \ldots$

samples



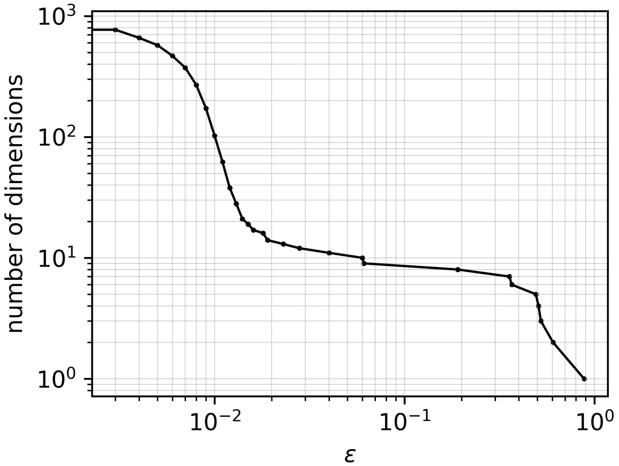
samples from the prior $p(x)$

samples from $p(x|x_1)$

samples from $p(x|x_{1,\ldots,}x_{20})$

28x28 inputs -> 784 latent dimensions with own variances and covariances:

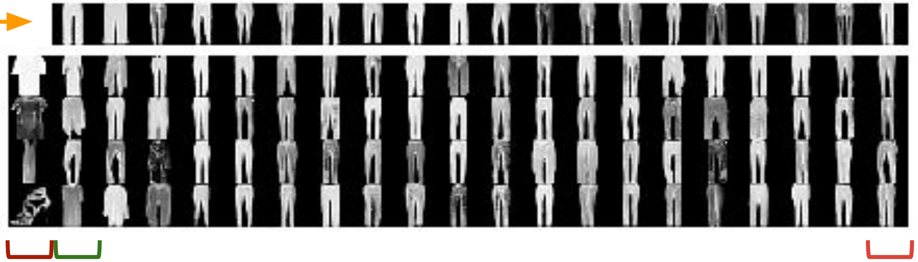
$$\Sigma = \begin{bmatrix} v & \rho & \rho & \ldots & \rho \\ \rho & v & \rho & \ldots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \ldots & v \end{bmatrix} \quad 0 \leq \rho < v$$



Learning to correlate

conditioning inputs $x_1, x_2, \ldots$

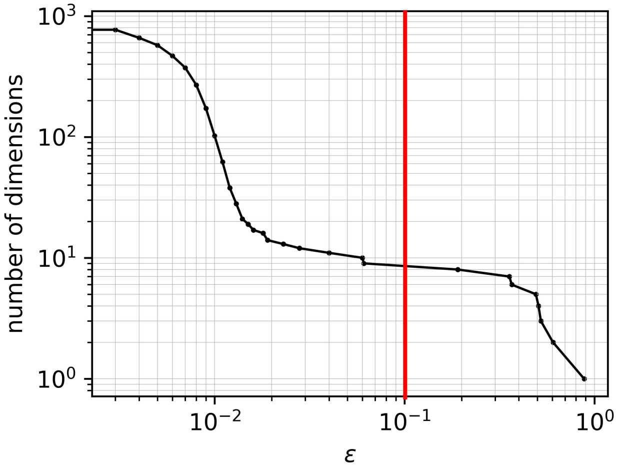
samples



samples from the prior $p(x)$

samples from $p(x|x_1)$

samples from $p(x|x_{1,\ldots,}x_{20})$



28x28 inputs -> 784 latent dimensions with own variances and covariances:

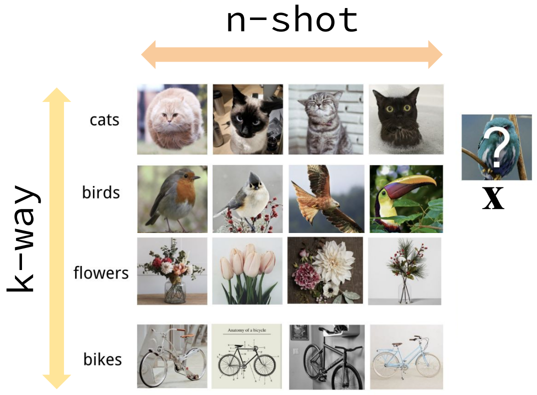
$$\Sigma = \begin{bmatrix} v & \rho & \rho & \ldots & \rho \\ \rho & v & \rho & \ldots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \ldots & v \end{bmatrix}_{0 \leq \rho < v}$$

Experiments: OMNIGLOT few-shot Generation

Omniglot: 1623 different handwritten characters from 50 different alphabets.
Each of the characters was drawn by 20 different people.

Experiments: OMNIGLOT few-shot classification

Model	5-way 1-shot	5-way 5-shot	20-way 1-shot	20-way 5-shot
Baseline Classifier*	80.0	95.0	69.5	89.1
Matching Nets*	98.1	98.9	93.8	98.5
BRUNO	86.3	95.6	69.2	87.7
BRUNO (discriminative fine-tuning)	97.1	99.4	91.3	97.8



episode

Test time

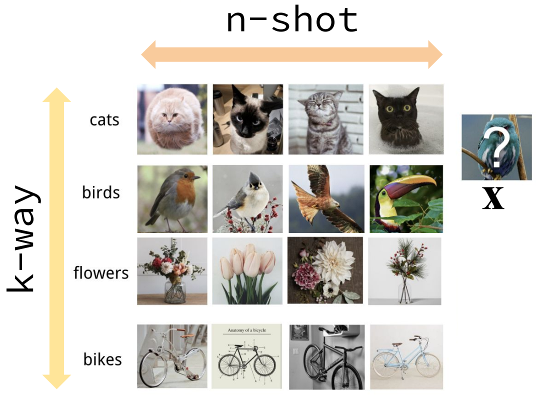
max

$$p(\mathbf{x}|\mathrm{cat}_{1:4})$$
$$p(\mathbf{x}|\mathrm{bird}_{1:4})$$
$$p(\mathbf{x}|\mathrm{flower}_{1:4})$$
$$p(\mathbf{x}|\mathrm{bike}_{1:4})$$

*O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, D. Wierstra. Matching Networks for One Shot Learning. NIPS'16

Experiments: OMNIGLOT few-shot classification

Model	5-way 1-shot	5-way 5-shot	20-way 1-shot	20-way 5-shot
Baseline Classifier*	80.0	95.0	69.5	89.1
Matching Nets*	98.1	98.9	93.8	98.5
BRUNO	86.3	95.6	69.2	87.7
BRUNO (discriminative fine-tuning)	97.1	99.4	91.3	97.8



episode

Train time

softmax

$$p(\mathbf{x}|\mathrm{cat}_{1:4})$$
$$p(\mathbf{x}|\mathrm{bird}_{1:4})$$
$$p(\mathbf{x}|\mathrm{flower}_{1:4})$$
$$p(\mathbf{x}|\mathrm{bike}_{1:4})$$

Fine-tune with a discriminative objective:

$$p(y = \text{‘bird’}|x, S)$$

*O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, D. Wierstra. Matching Networks for One Shot Learning. NIPS'16

BRUNO

* Likelihoods $p(\mathbf{x}_{n+1}|\mathbf{x}_{1:n})$
* Easy sampling from $p(\mathbf{x}_{n+1}|\mathbf{x}_{1:n})$

BRUNO

* Likelihoods $p(\mathbf{x}_{n+1}|\mathbf{x}_{1:n})$
* Easy sampling from $p(\mathbf{x}_{n+1}|\mathbf{x}_{1:n})$

Conditional BRUNO

Can we model $p(\mathbf{x}_{n+1}|\mathbf{h}_{n+1}, \mathbf{x}_{1:n}, \mathbf{h}_{1:n})$?

BRUNO

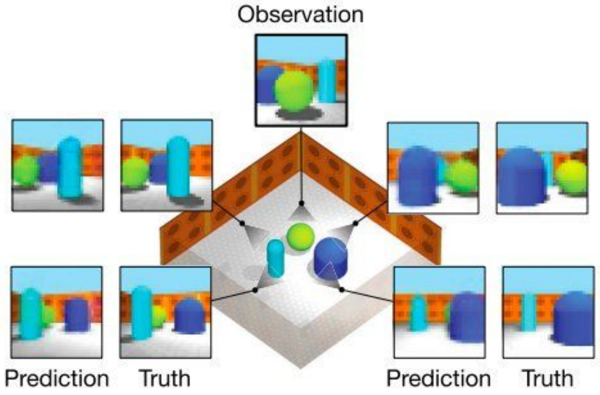
* Likelihoods $p(\mathbf{x}_{n+1}|\mathbf{x}_{1:n})$
* Easy sampling from $p(\mathbf{x}_{n+1}|\mathbf{x}_{1:n})$

Conditional BRUNO

observations

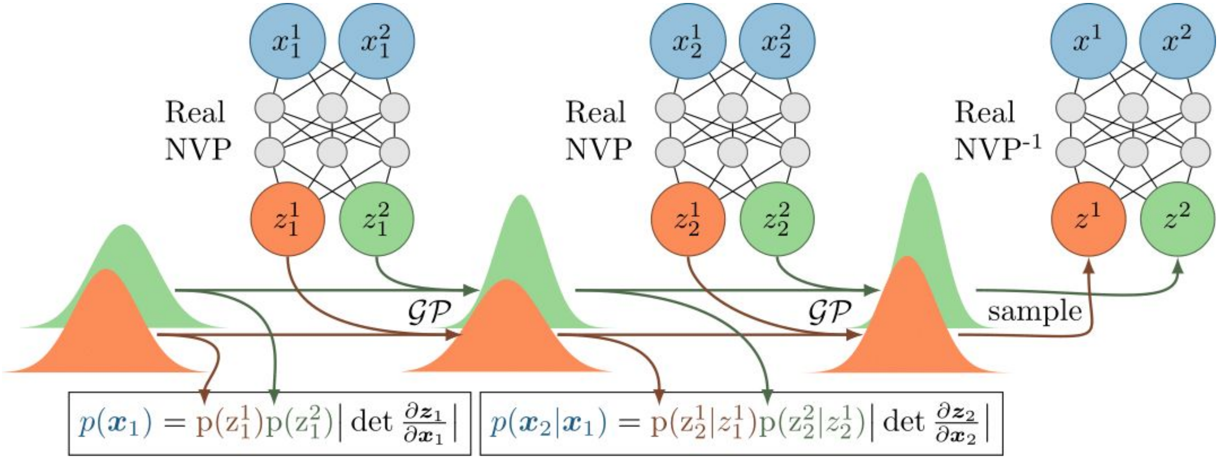
Can we model $p(\mathbf{x}_{n+1}|\mathbf{h}_{n+1}, \mathbf{x}_{1:n}, \mathbf{h}_{1:n})$?

scene view-point



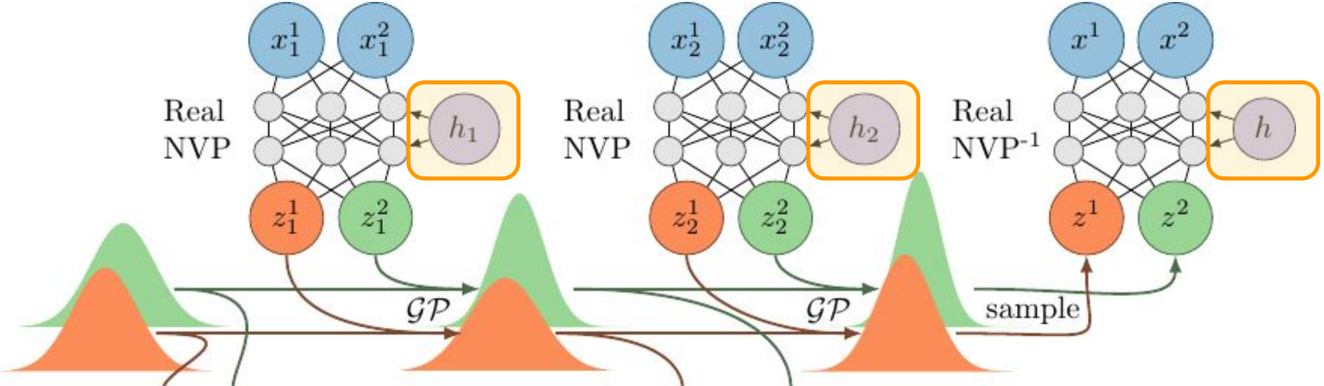
Eslami et al. Neural scene representation and rendering. Science'18

BRUNO



x_1^1
x_1^2
Real NVP
z_1^1
z_1^2
x_2^1
x_2^2
Real NVP
z_2^1
z_2^2
x^1
x^2
Real NVP-1
z^1
z^2
$\mathcal{GP}$
$\mathcal{GP}$
sample
$p(\boldsymbol{x}_1) = \mathrm{p}(\mathrm{z}_1^1)\mathrm{p}(\mathrm{z}_1^2)\left|\det \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{x}_1}\right|$
$p(\boldsymbol{x}_2|\boldsymbol{x}_1) = \mathrm{p}(\mathrm{z}_2^1|z_1^1)\mathrm{p}(\mathrm{z}_2^2|z_2^1)\left|\det \frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{x}_2}\right|$

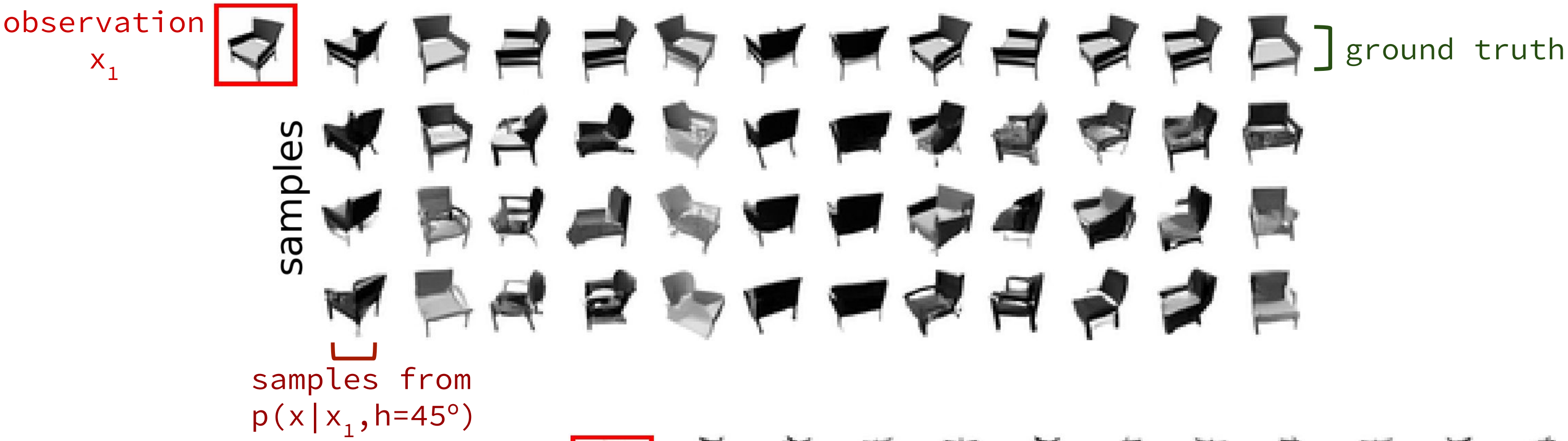
Conditional BRUNO

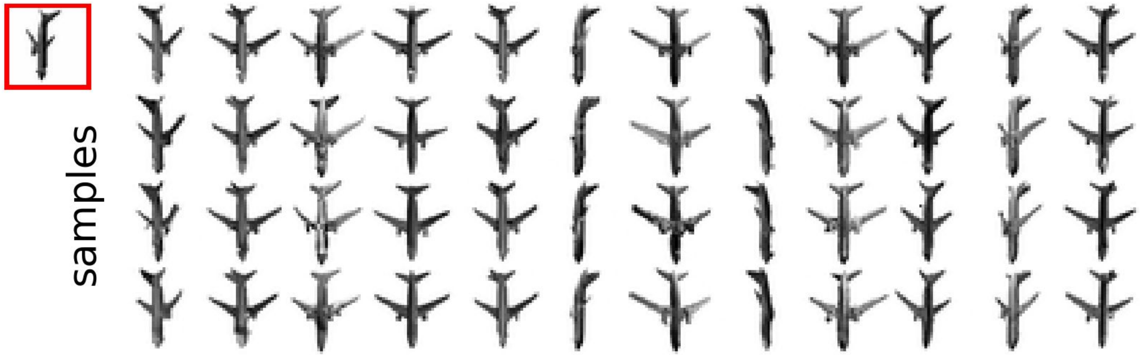


$p(\boldsymbol{x}_1|\boldsymbol{h}_1) = \mathrm{p}(\mathrm{z}_1^1)\mathrm{p}(\mathrm{z}_1^2)\left|\det \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{x}_1}\right|$

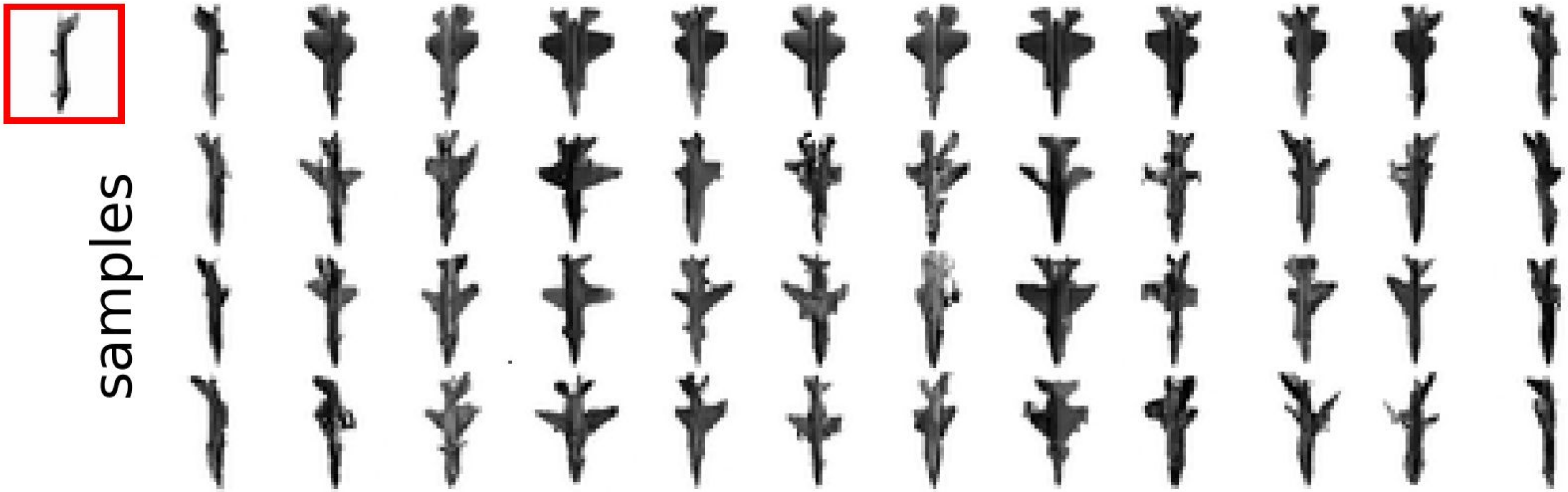
$p(\boldsymbol{x}_2|\boldsymbol{x}_1, \boldsymbol{h}_{1:2}) = \mathrm{p}(\mathrm{z}_2^1|z_1^1)\mathrm{p}(\mathrm{z}_2^2|z_2^1)\left|\det \frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{x}_2}\right|$

Experiments: ShapeNet chairs & airplanes

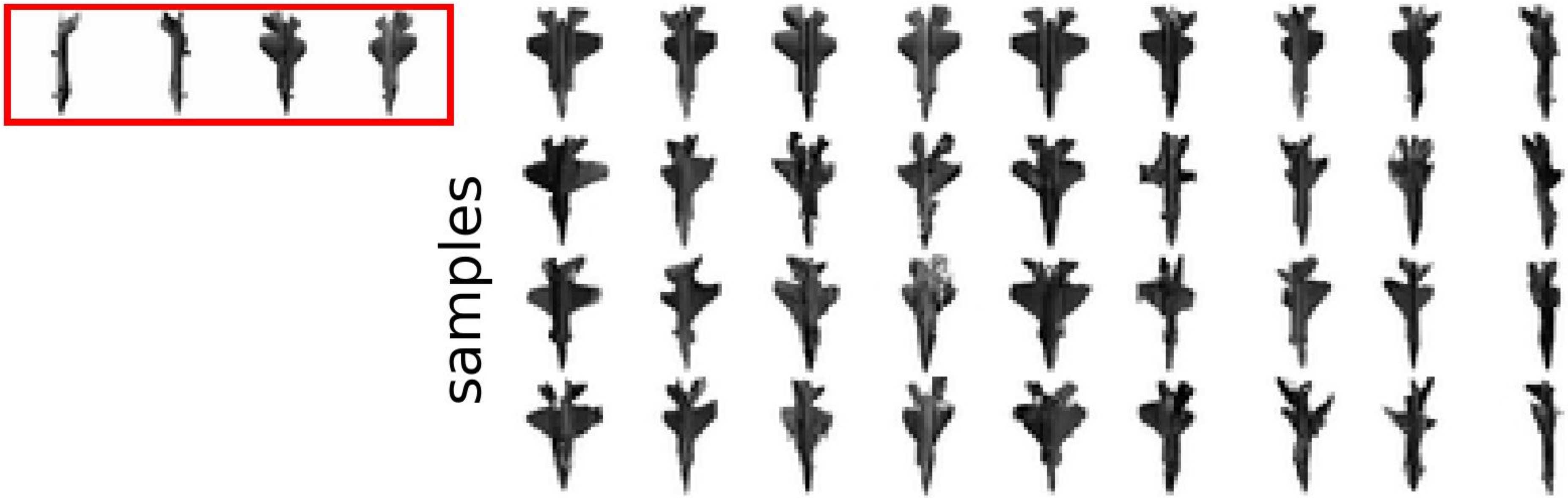




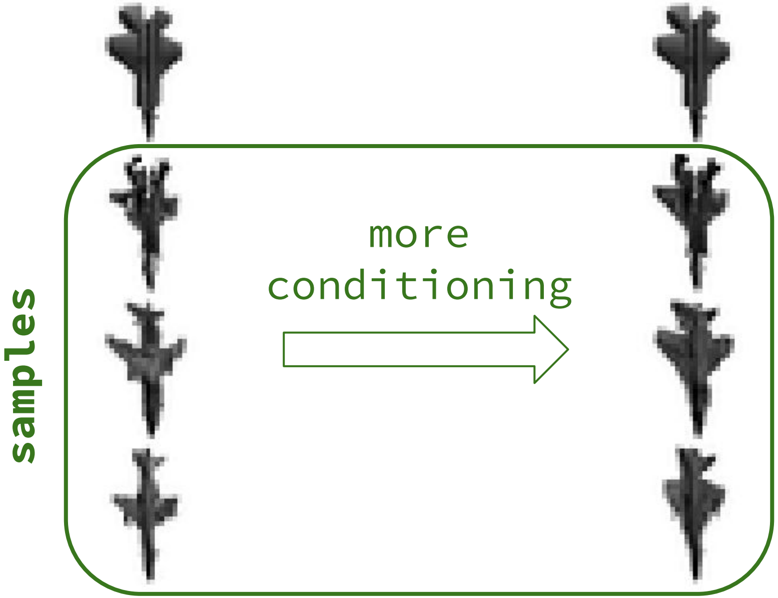
Experiments: ShapeNet chairs & airplanes



Experiments: ShapeNet chairs & airplanes



Experiments: ShapeNet chairs & airplanes



Conclusion

BRUNO = expressiveness of DNNs + data-efficiency of GPs

A meta-learning exchangeable model with

- exact likelihoods
- fast sampling and inference
- no retraining or changes to the architecture at test time
- recurrent formulation