Few-shot Generative modelling

Sergey Bartunov
DeepMind

Talk plan

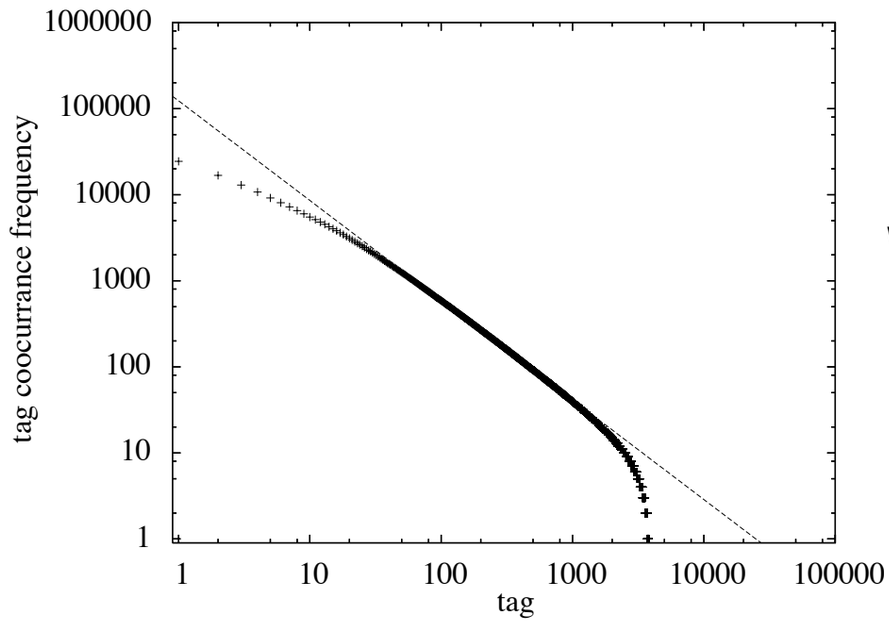
- Introduction: few-shot learning and failure modes of gradient learning methods
- Remainder on generative models
- Old and new in few-shot density estimation
- Learning to learn or meta-learning

ILSVRC and the IMAGENET dataset

Image classification annotations (1000 object classes) Russakovsky et al, 2015

Year	Train images (per class)	Val images (per class)	Test images (per class)
ILSVRC2010	1,261,406 (668-3047)	50,000 (50)	150,000 (150)
ILSVRC2011	1,229,413 (384-1300)	50,000 (50)	100,000 (100)
ILSVRC2012-14	1,281,167 (732-1300)	50,000 (50)	100,000 (100)

Power law distribution!



Weinberger et al, 2008

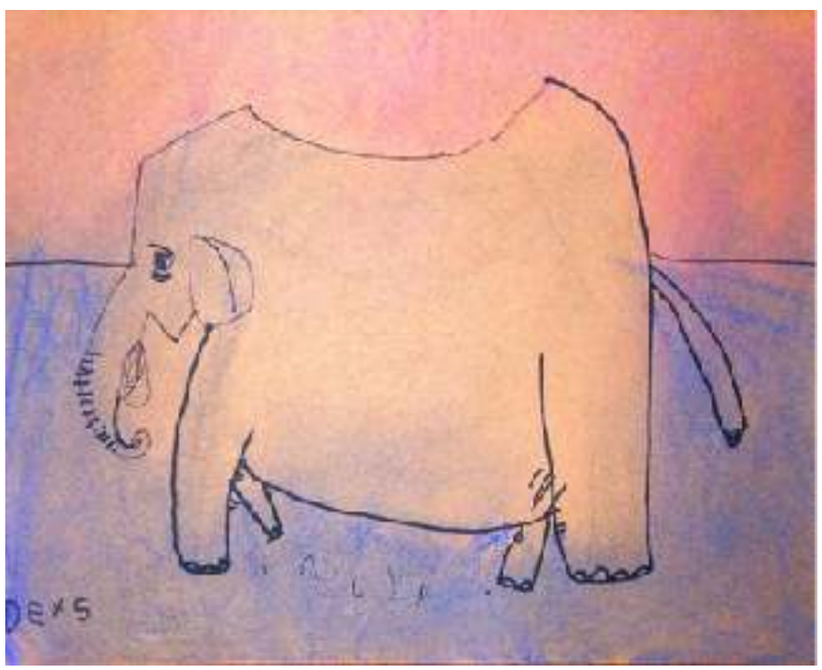
Learning with gradient methods

$$\theta^{(t+1)} = \theta^{(t)} - \eta^{(t)} \nabla L(x^{(t)}; \theta^{(t)}), \quad x^{(t)} \sim p_{\text{data}}(x)$$

- Slow, requires many passes over training data
- Prone to catastrophic forgetting
- Also prone to overfitting

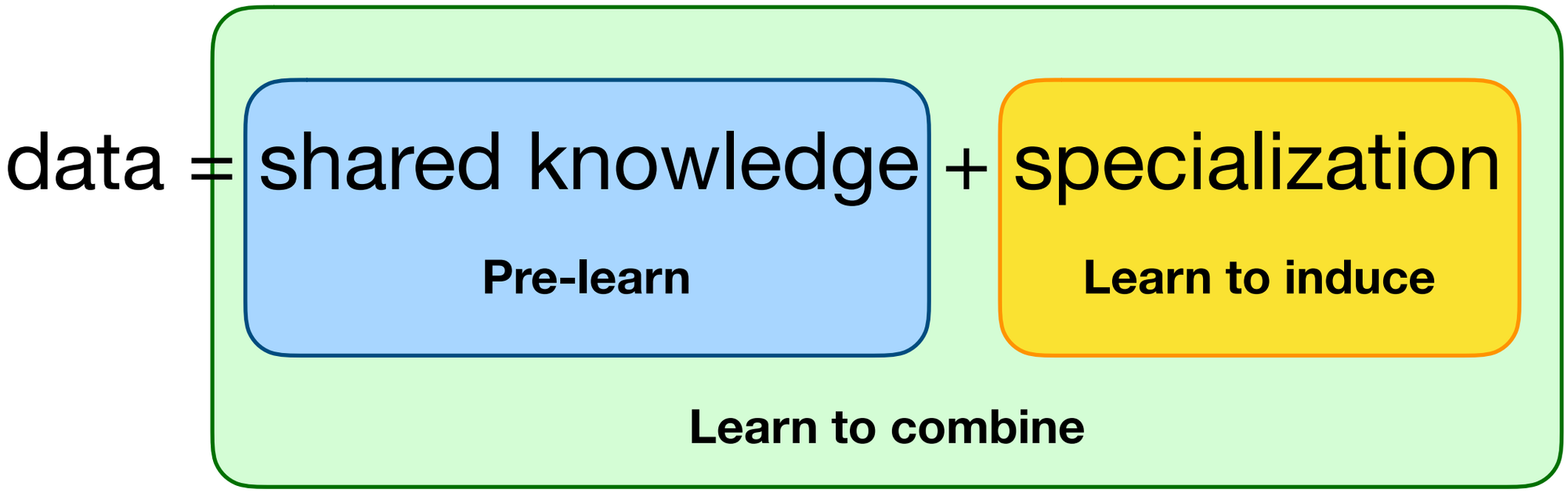
Fast learning

© Morkel Erasmus



kinderart.com

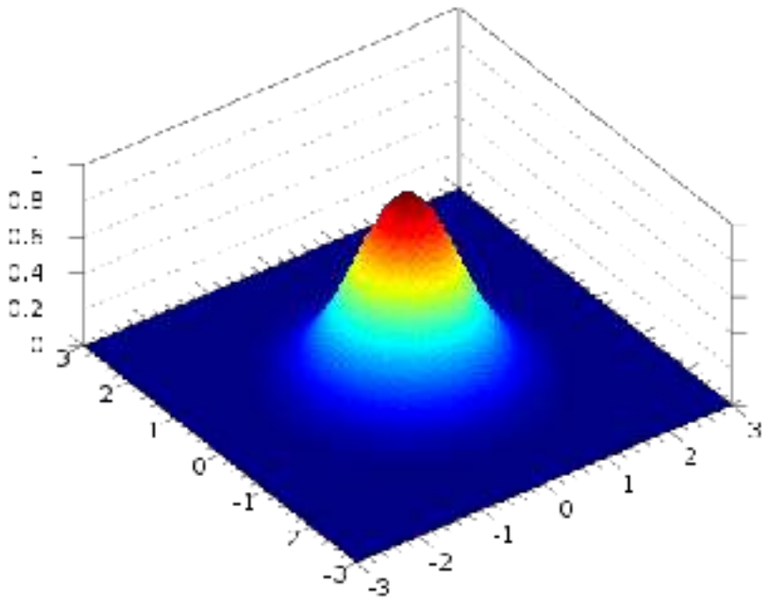
General idea of fast learning



Omniglot dataset for few-shot learning

- De facto standard dataset for few-shot learning, introduced by Lake at al, 2015
- “Transposed” MNIST - 1623 classes of handwritten characters, 20 examples for each
- Standard methods tend to show relatively poor performance on Omniglot

Generative models

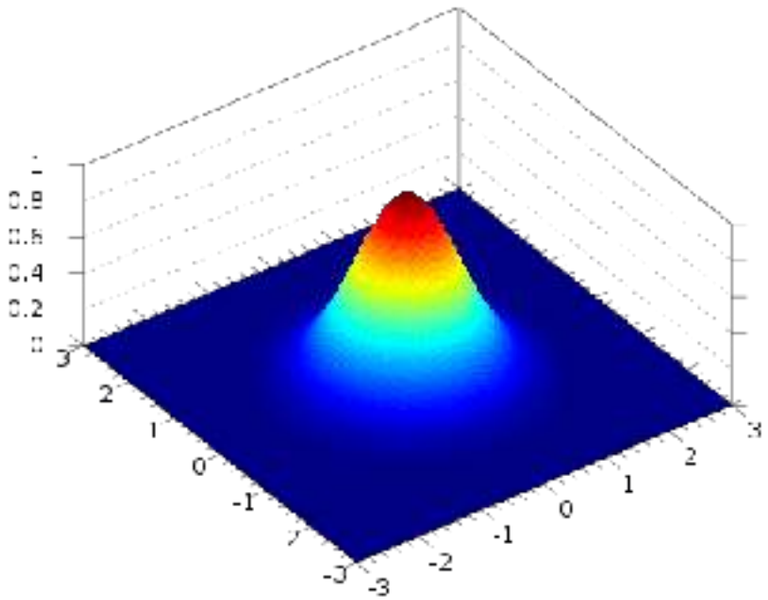


$$\epsilon \sim p(\epsilon)$$

Radford et al, 2015

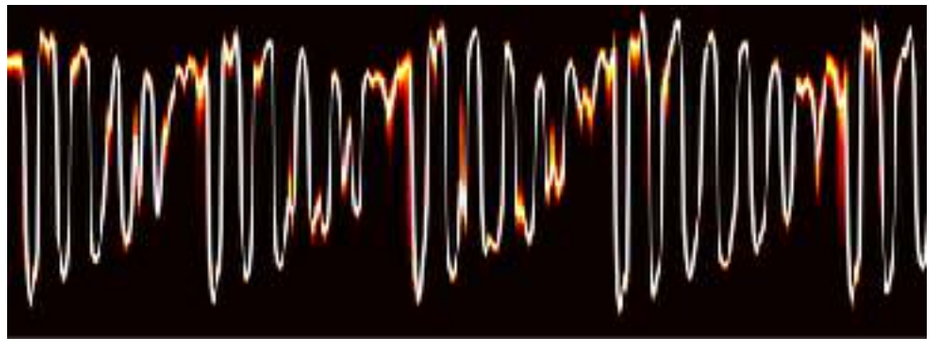
$$x = f(\epsilon; \theta)$$

Generative models



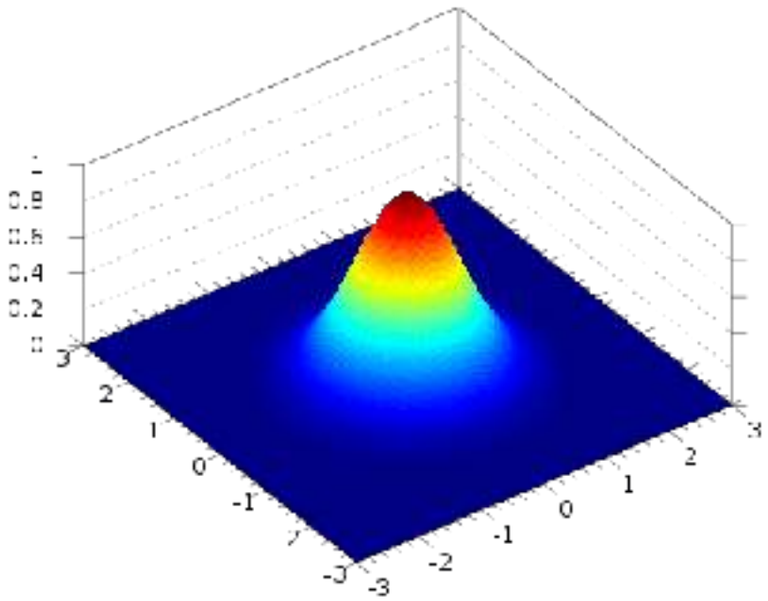
$$\epsilon \sim p(\epsilon)$$

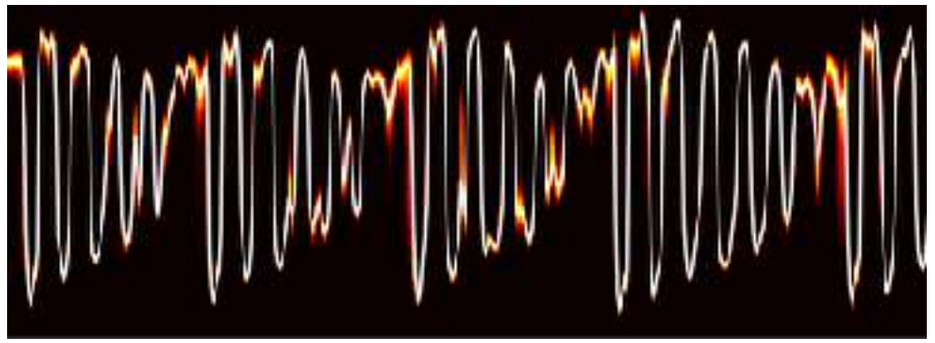
Van den Oord, 2016



$$x = f(\epsilon; \theta)$$

Variational autoencoders





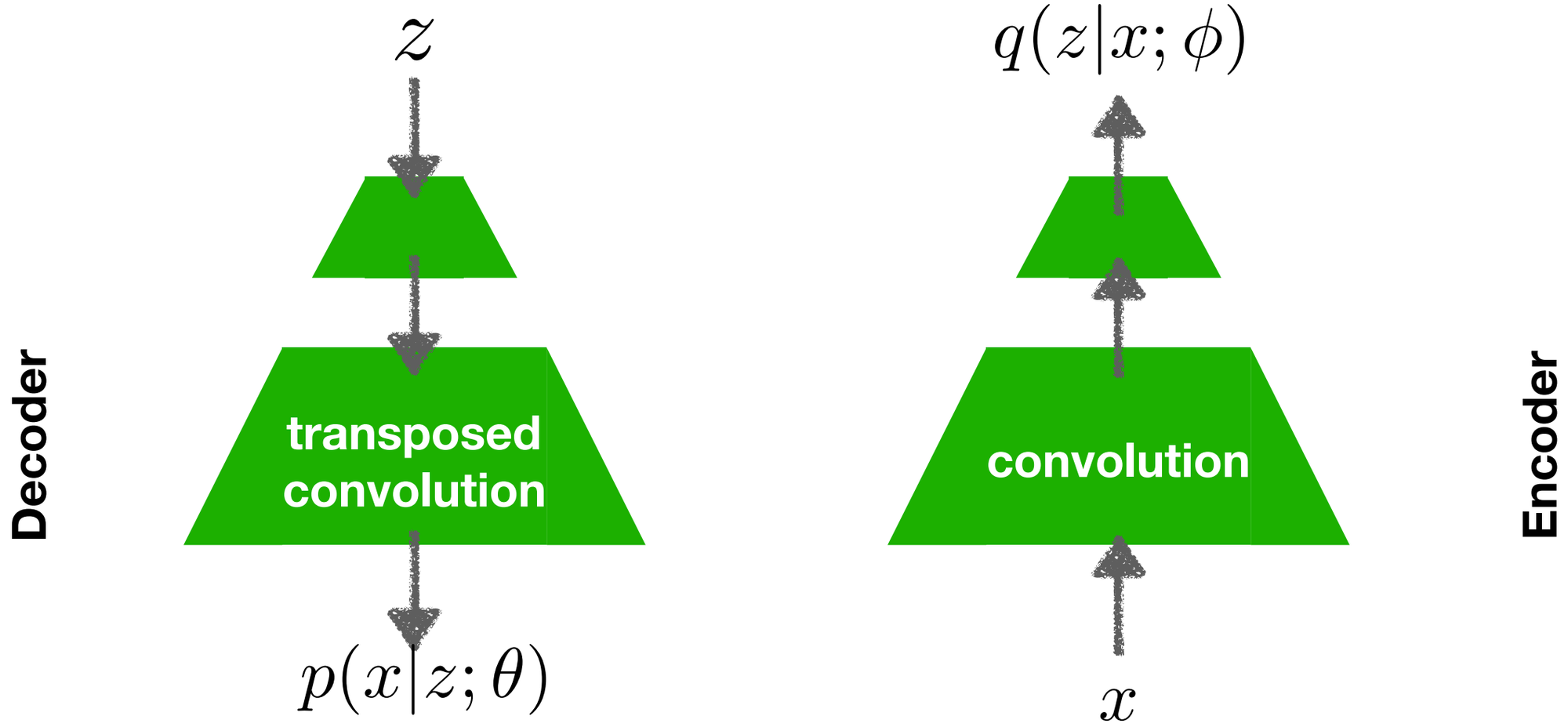
$z \sim p(z)$

$x \sim p(x|z;\theta)$

$q(z|x;\phi)$

Rezende et al, 2014; Kingma & Welling, 2014

Example: convolutional VAE



Variational inference

Standard Normal prior

Likelihood (generative network)

$$z \sim \mathcal{N}(0, I) \longrightarrow x \sim \text{Bernoulli}(\alpha(z, \theta))$$

$$q(z|x;\phi) = \mathcal{N}(\mu(x,\phi), \Sigma(x,\phi))$$

Approximate posterior (inference network)

$$\begin{aligned}\log p(x;\theta) &\geq \mathcal{L}(\theta,\phi) \\ &= \mathbb{E}_{q(z|x;\phi)} \log p(x|z;\theta) - \text{KL}(q(z|x;\phi)||p(z)) \\ &\rightarrow \max_{\theta,\phi}\end{aligned}$$

Fast learning in probabilistic models

- For exchangeable data one can find a model such that

$$p(X) = \int p(\alpha) \prod_{i=1}^{N} p(x_i|\alpha) d\alpha$$

- Justified by the de Finetti theorem

Intractable integral!

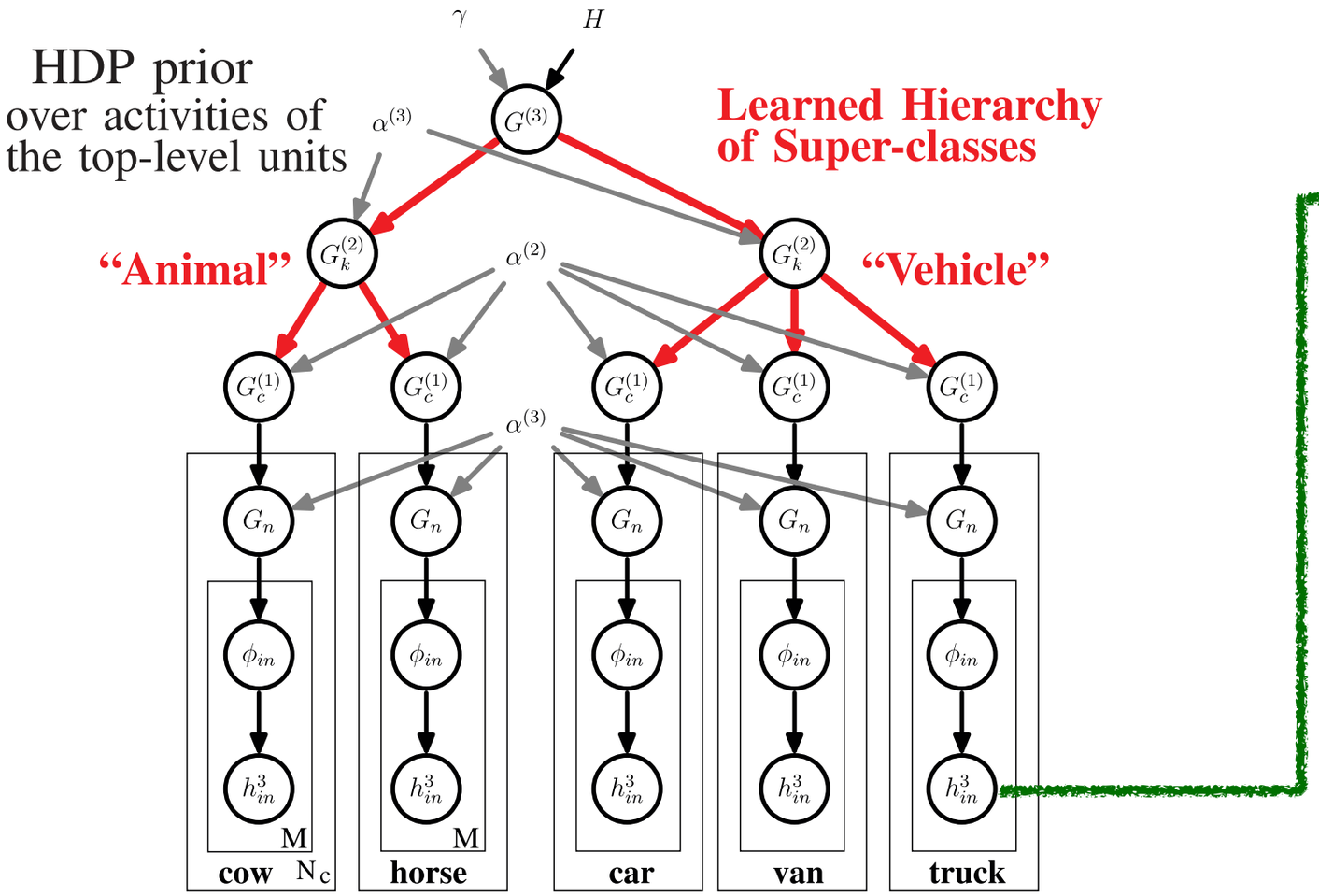
- Generating a new sample is then simple

$$p(x'|X) = \int p(\alpha|X) p(x'|\alpha) d\alpha$$

- The only caveat is posterior inference $p(\alpha|X) = \dfrac{p(\alpha)\prod_{i=1}^{N} p(x_i|\alpha)}{p(X)}$

See (Salakhutdinov et al, 2013) and (Lake at al, 2015)

Hierarchical Dirichlet Process Deep Boltzmann Machine



High-level feature clustering

Shared generator

Salakhutdinov et al, 2013

Hierarchical Dirichlet Process Deep Boltzmann Machine

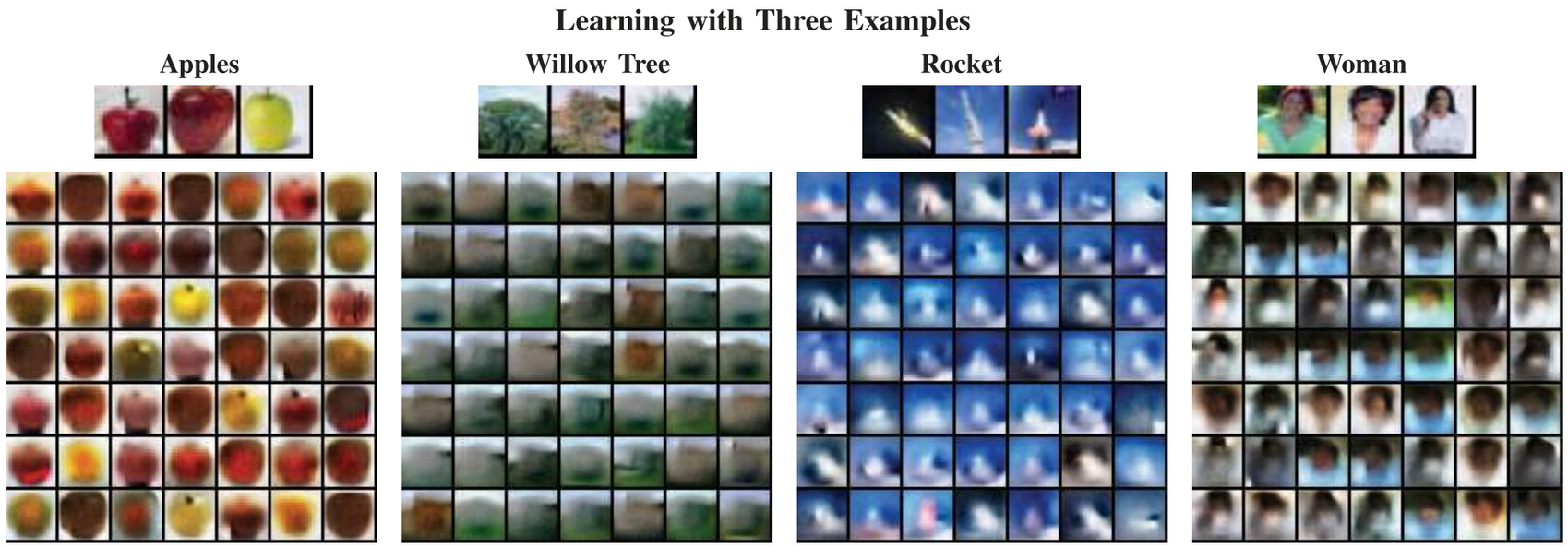
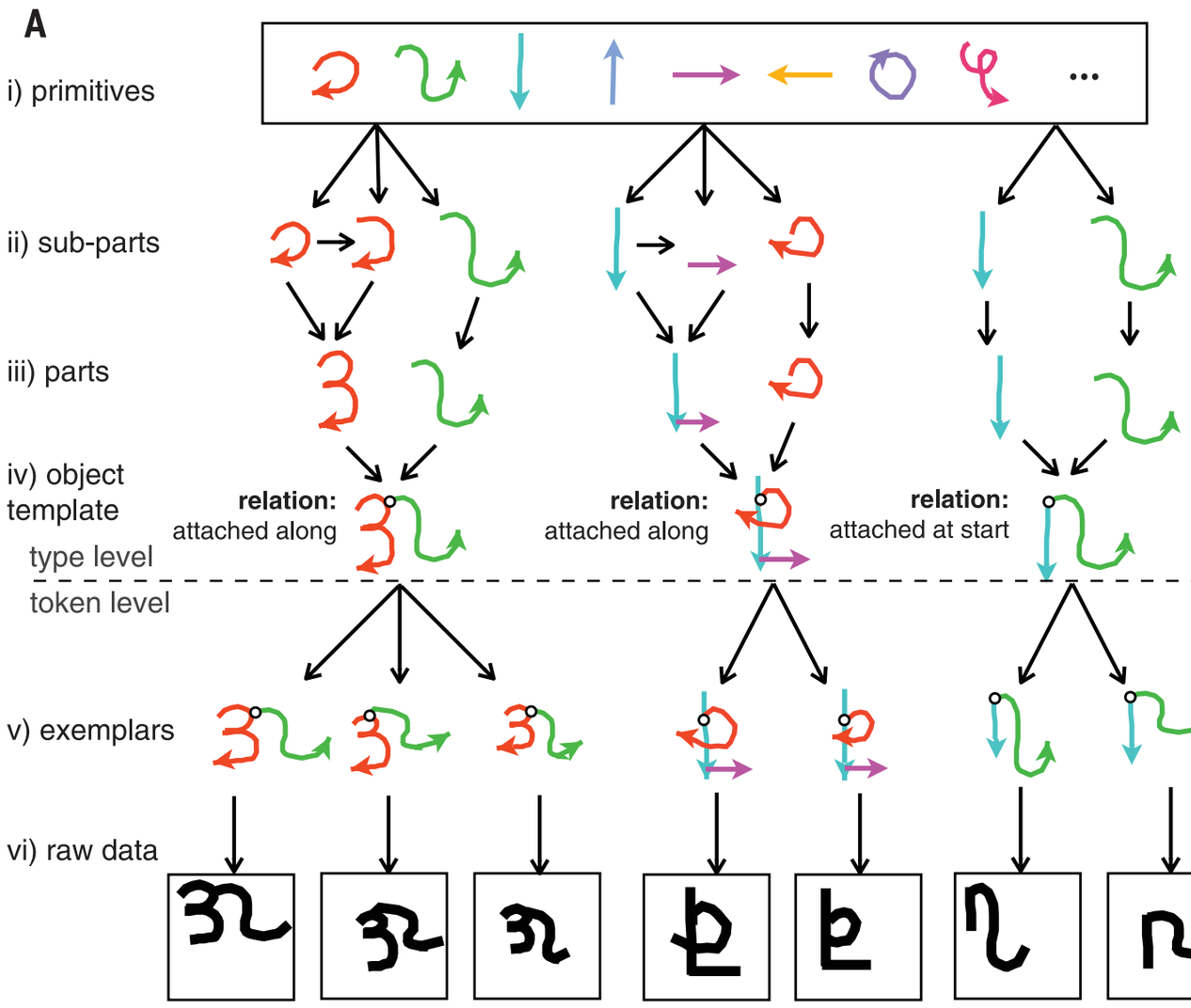


Fig. 7. Conditional samples generated by the HDP-DBM model when learning with only three training examples of a novel class: Top: Three training examples, Bottom: 49 conditional samples. Best viewed in color.

Fig. 12. Each panel shows three figures from left to right: 1) three training examples of a novel character class, 2) 12 synthesized examples of that class, and 3) training characters in the same supercategory that the novel character has been assigned to.

Salakhutdinov et al, 2013

Human-level concept learning through probabilistic program induction



B

procedure GENERATETYPE
- $\kappa \leftarrow P(\kappa)$ ▷ Sample number of parts
- **for** i = 1 ... κ **do**
  - $n_i \leftarrow P(n_i|\kappa)$ ▷ Sample number of sub-parts
  - **for** j = 1 ... n_i **do**
    - $s_{ij} \leftarrow P(s_{ij}|s_{i(j-1)})$ ▷ Sample sub-part sequence
  - **end for**
  - $R_i \leftarrow P(R_i|S_1, ..., S_{i-1})$ ▷ Sample relation
- **end for**
- $\psi \leftarrow \{\kappa, R, S\}$
- **return** @GENERATETOKEN(ψ) ▷ Return program

procedure GENERATETOKEN(ψ)
- **for** $i = 1...\kappa$ **do**
  - $S_i^{(m)} \leftarrow P(S_i^{(m)}|S_i)$ ▷ Add motor variance
  - $L_i^{(m)} \leftarrow P(L_i^{(m)}|R_i, T_1^{(m)}, ..., T_{i-1}^{(m)})$ ▷ Sample part's start location
  - $T_i^{(m)} \leftarrow f(L_i^{(m)}, S_i^{(m)})$ ▷ Compose a part's trajectory
- **end for**
- $A^{(m)} \leftarrow P(A^{(m)})$ ▷ Sample affine transform
- $I^{(m)} \leftarrow P(I^{(m)}|T^{(m)}, A^{(m)})$ ▷ Sample image
- **return** $I^{(m)}$

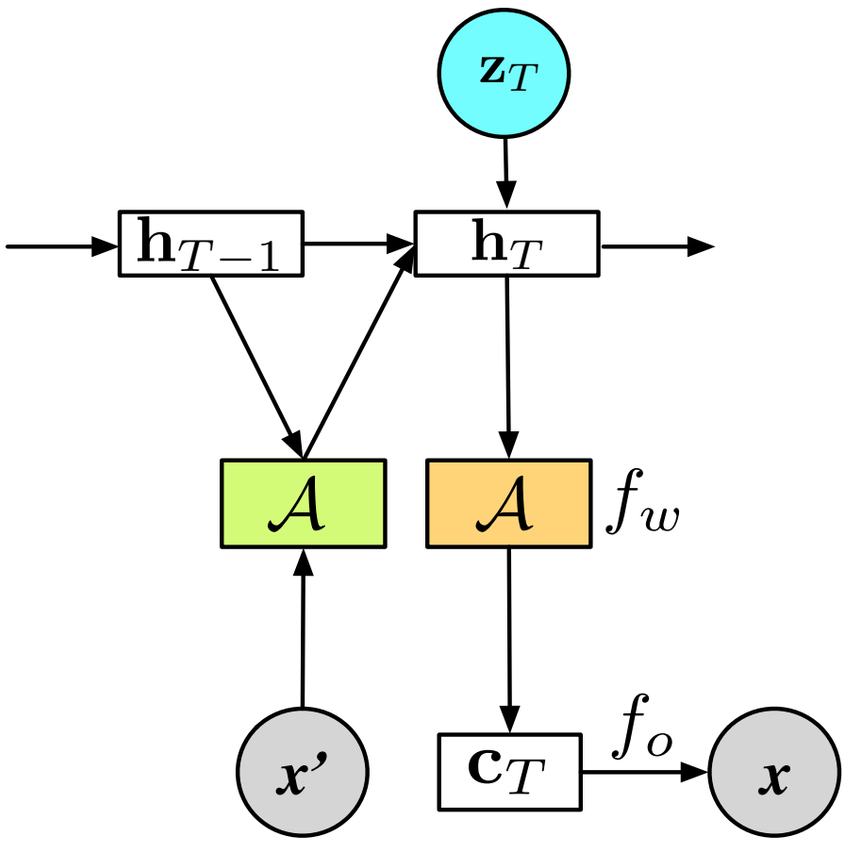
Fig. 3. A generative model of handwritten characters. (**A**) New types are generated by choosing primitive actions (color coded) from a library (i), combining these subparts (ii) to make parts (iii), and combining parts with relations to define simple programs (iv). New tokens are generated by running these programs (v), which are then rendered as raw data (vi). (**B**) Pseudocode for generating new types ψ and new token images $I^{(m)}$ for m = 1, ..., M. The function f (·, ·) transforms a subpart sequence and start location into a trajectory.

Lake et al, 2015

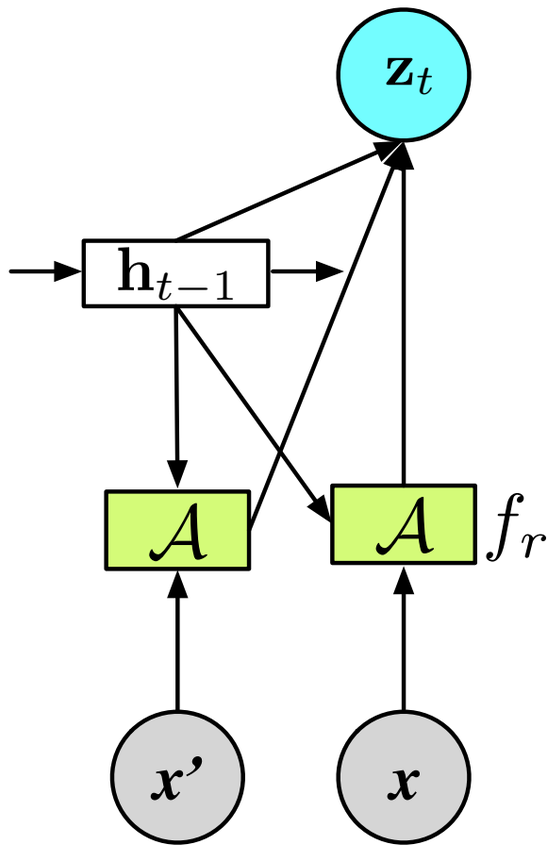
One-Shot Generalization in Deep Generative Models

Rezende et al, 2016

Generative model



Inference model



$$p(x, \mathbf{z}) = \prod_{t=1}^{T} p(z_t) p(x|\mathbf{z}, x')$$

$$q(\mathbf{z}|x, x') = \prod_{t=1}^{T} q(z_t|\mathbf{z}_{<t}, x, x')$$

One-Shot Generalization in Deep Generative Models

Rezende et al, 2016

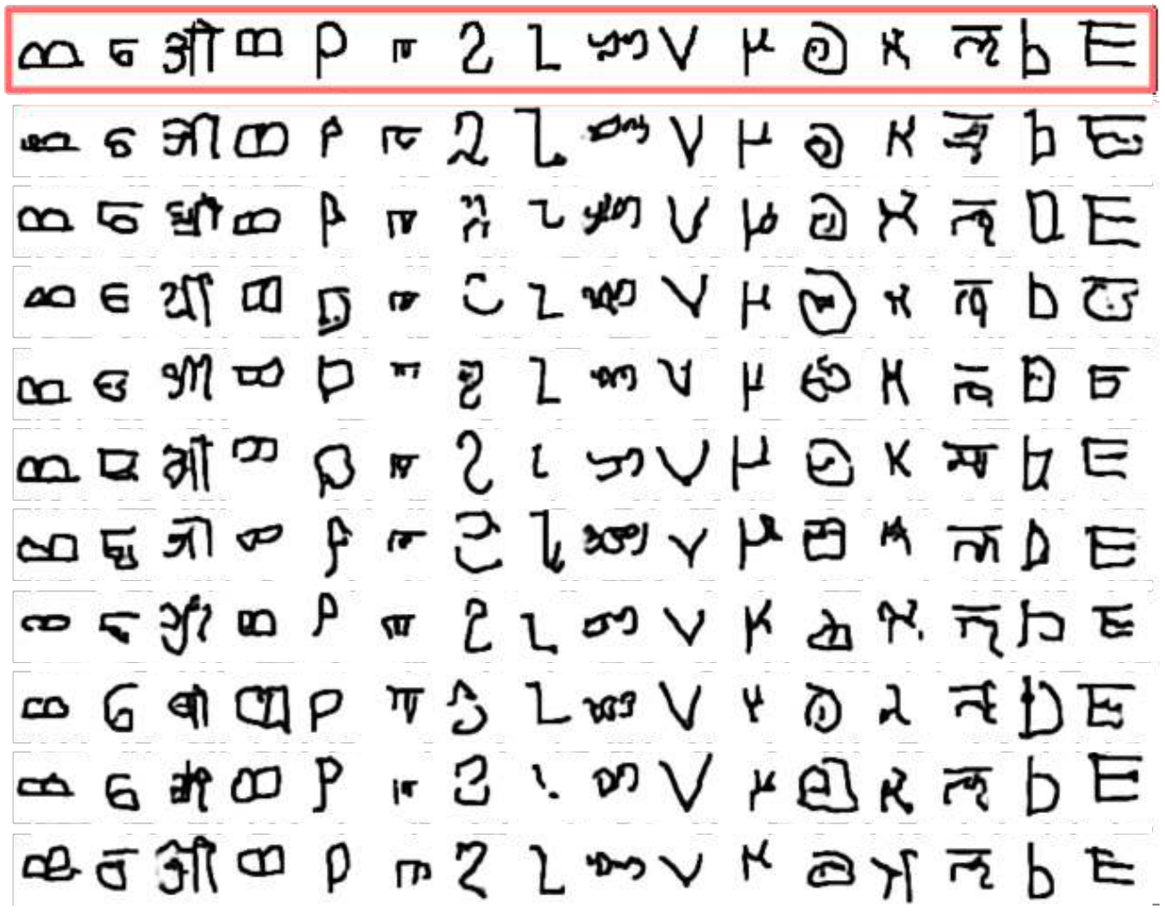


Figure 9. Generating new examplars of a given character for the weak generalization test (task 2*a*). The first row shows the test images and the next 10 are one-shot samples from the model.

Figure 11. Generating new exemplars from a novel alphabet (task 3). The first row shows the test images, and the next 10 rows are one-shot samples generated by the model.

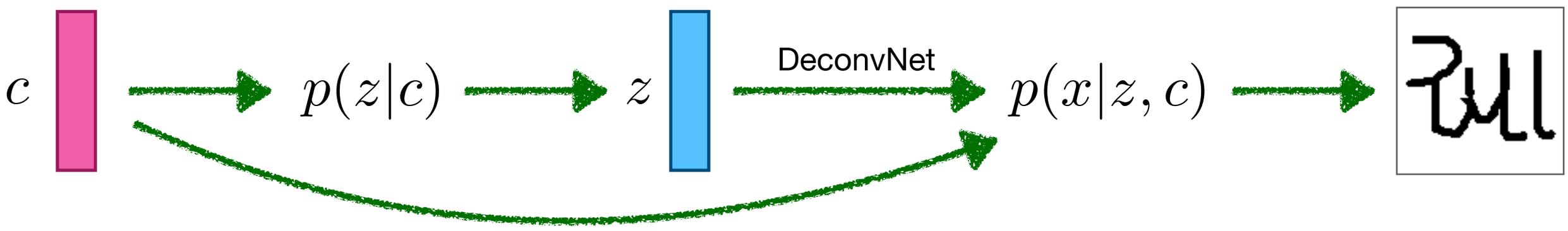
Neural statistician

Edwards & Storkey, 2016

Variational autoencoder for **datasets**

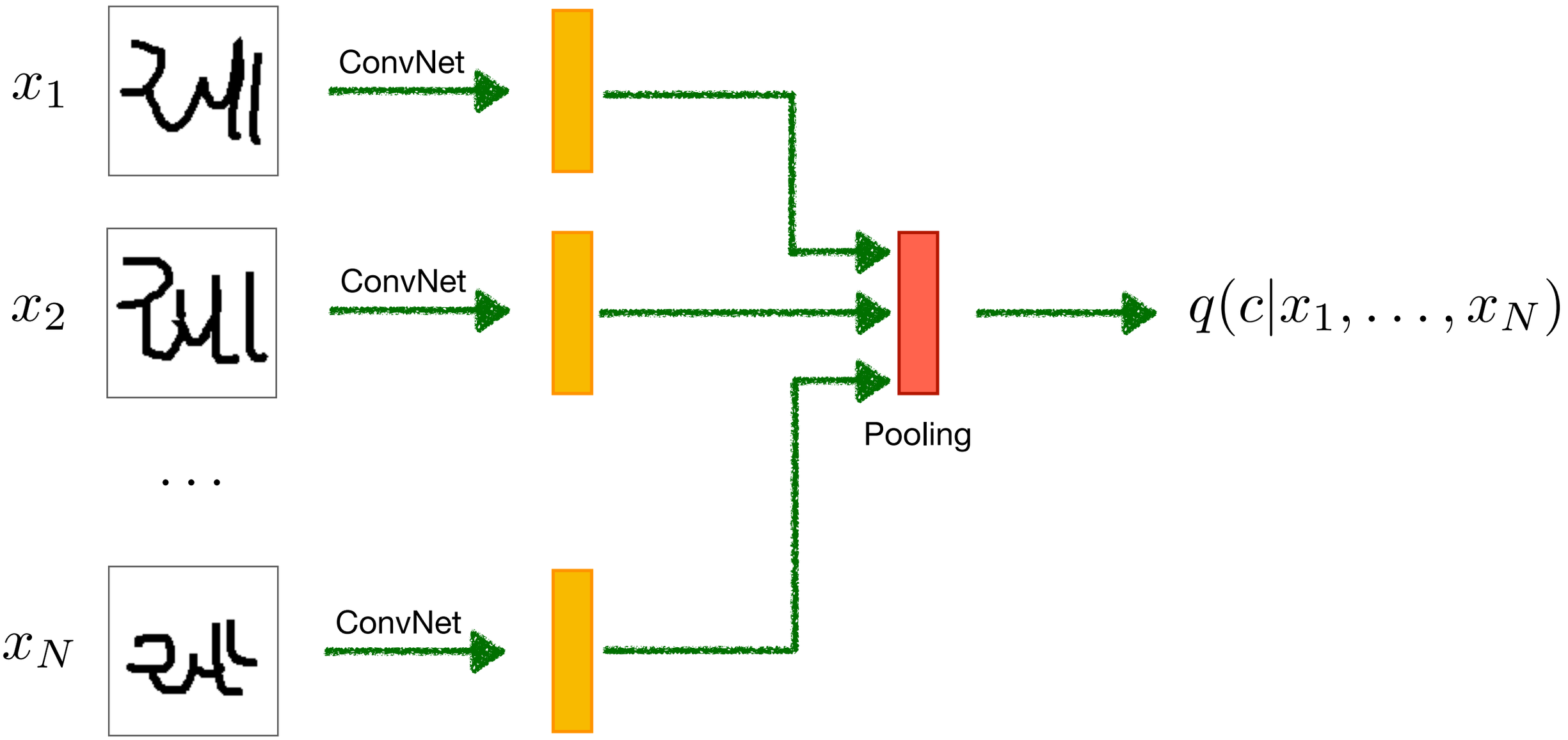
$$p(x|X) = \int p(c|X) \int p(z|c)p(x|z,c)dzdc$$

$$\approx \int q(c|X) \int p(z|c)p(x|z,c)dzdc$$



Neural statistician

Edwards & Storkey, 2016



Neural statistician

Edwards & Storkey, 2016

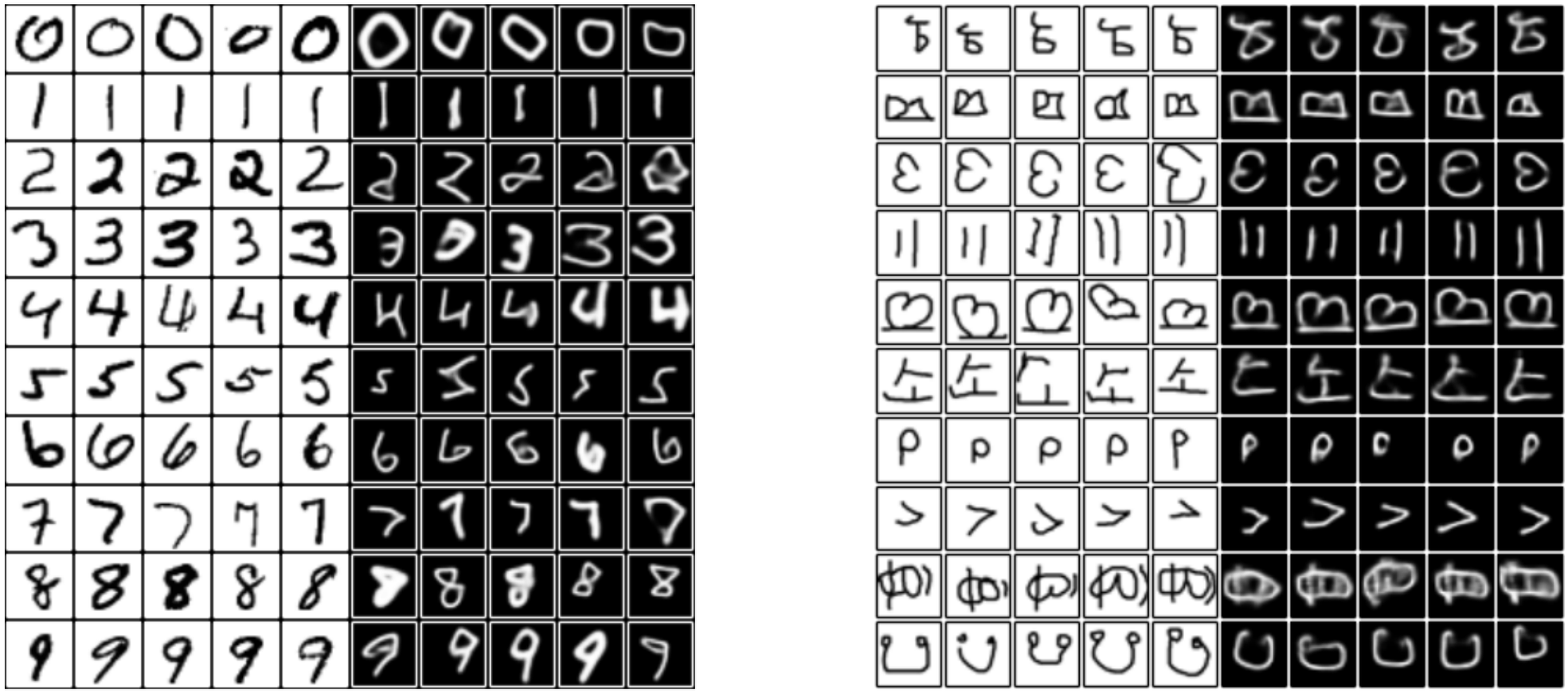


Figure 5: Few-shot learning *Left*: Few-shot learning from OMNIGLOT to MNIST. Left rows are input sets, right rows are samples given the inputs. *Right*: Few-shot learning from with OMNIGLOT data to unseen classes. Left rows are input sets, right rows are samples given the inputs. Black-white inversion is applied for ease of viewing.

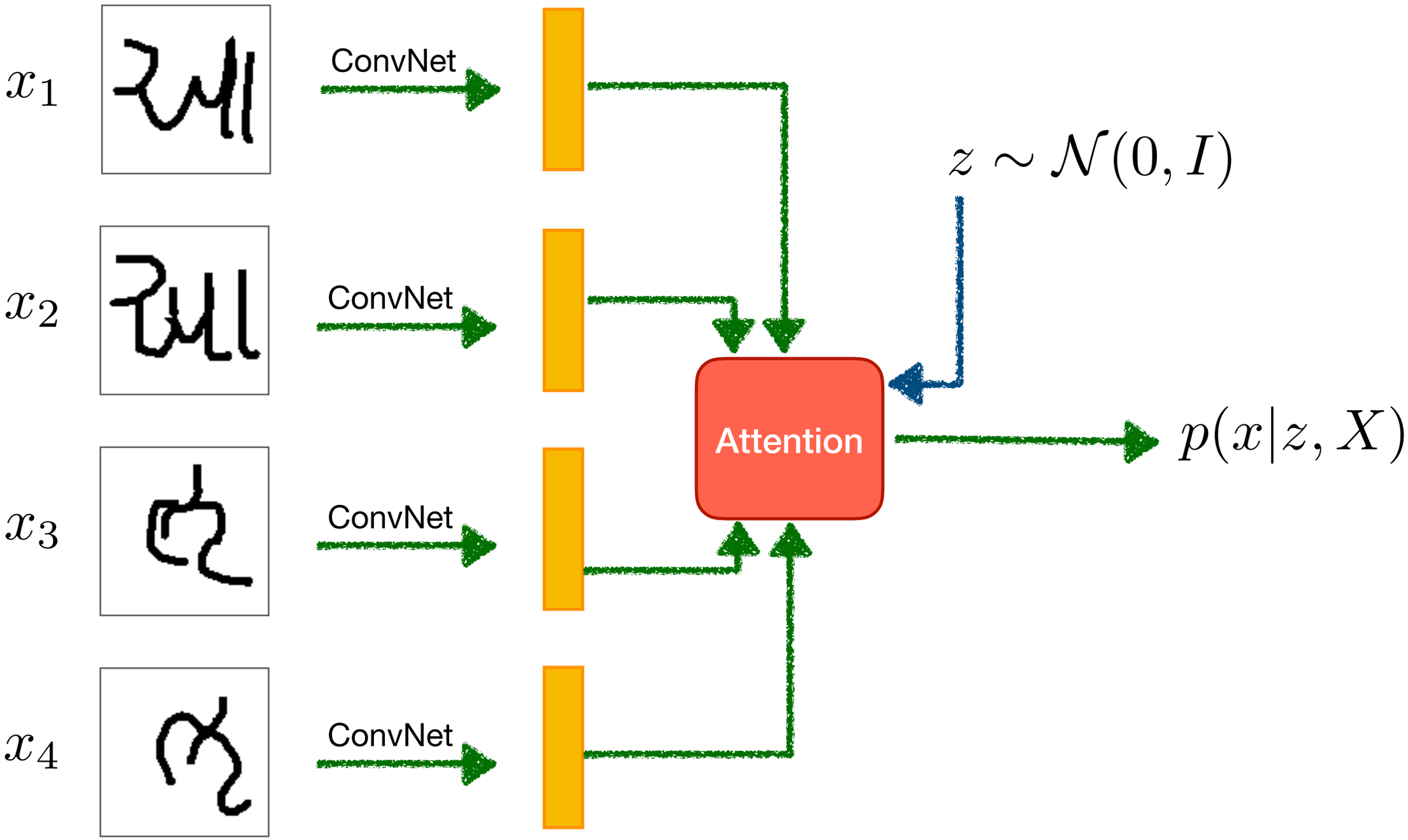
Neural statistician

Edwards & Storkey, 2016

Figure 6: Few-shot learning for face data. Samples are from model trained on Youtube Faces Database. *Left*: Each row shows an input set of size 5. *Center*: Each row shows 5 samples from the model corresponding to the input set on the left. *Right*: Imagined new faces generated by sampling contexts from the prior. Each row consists of 5 samples from the model given a particular sampled context.

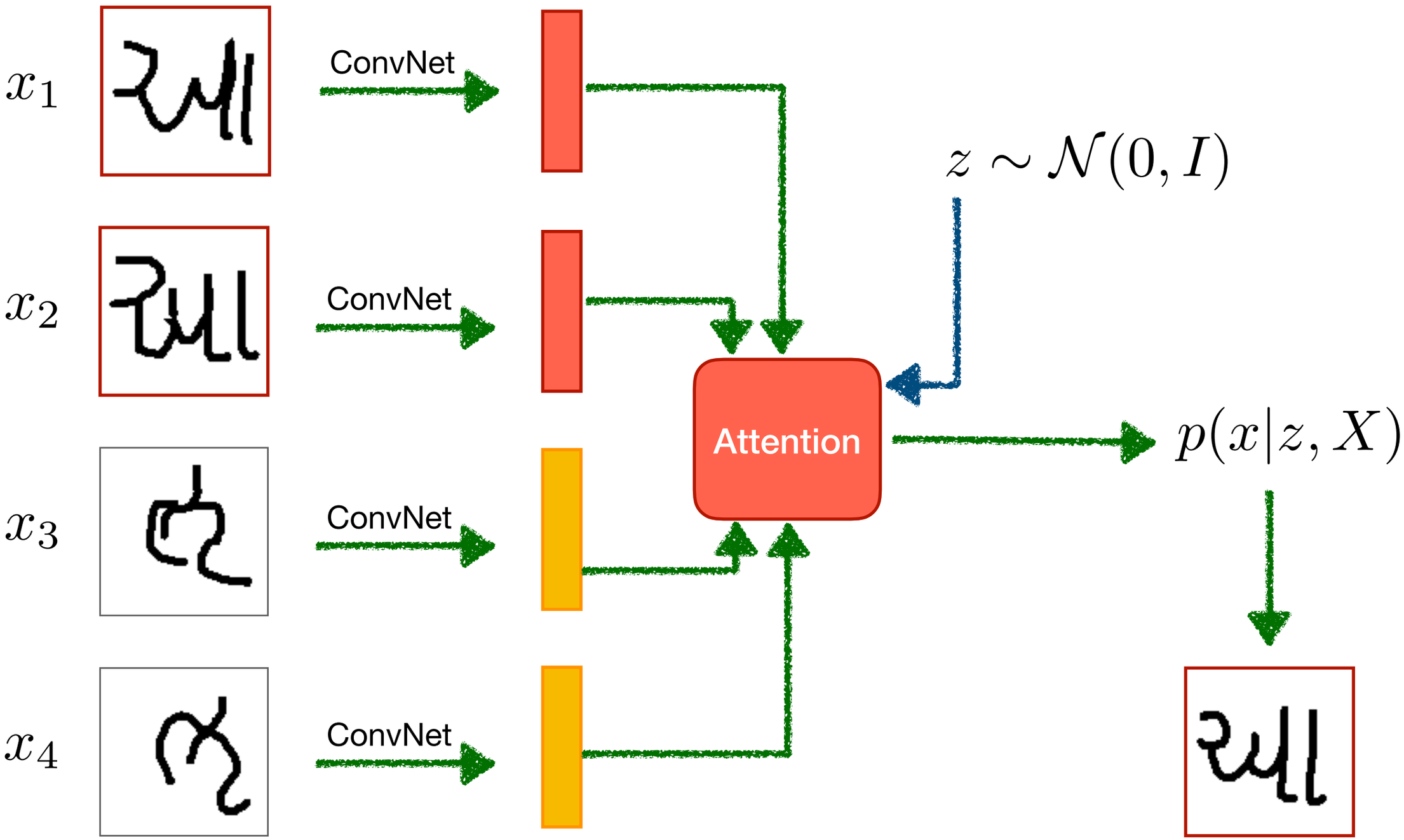
Generative Matching Networks

Bartunov & Vetrov, 2016



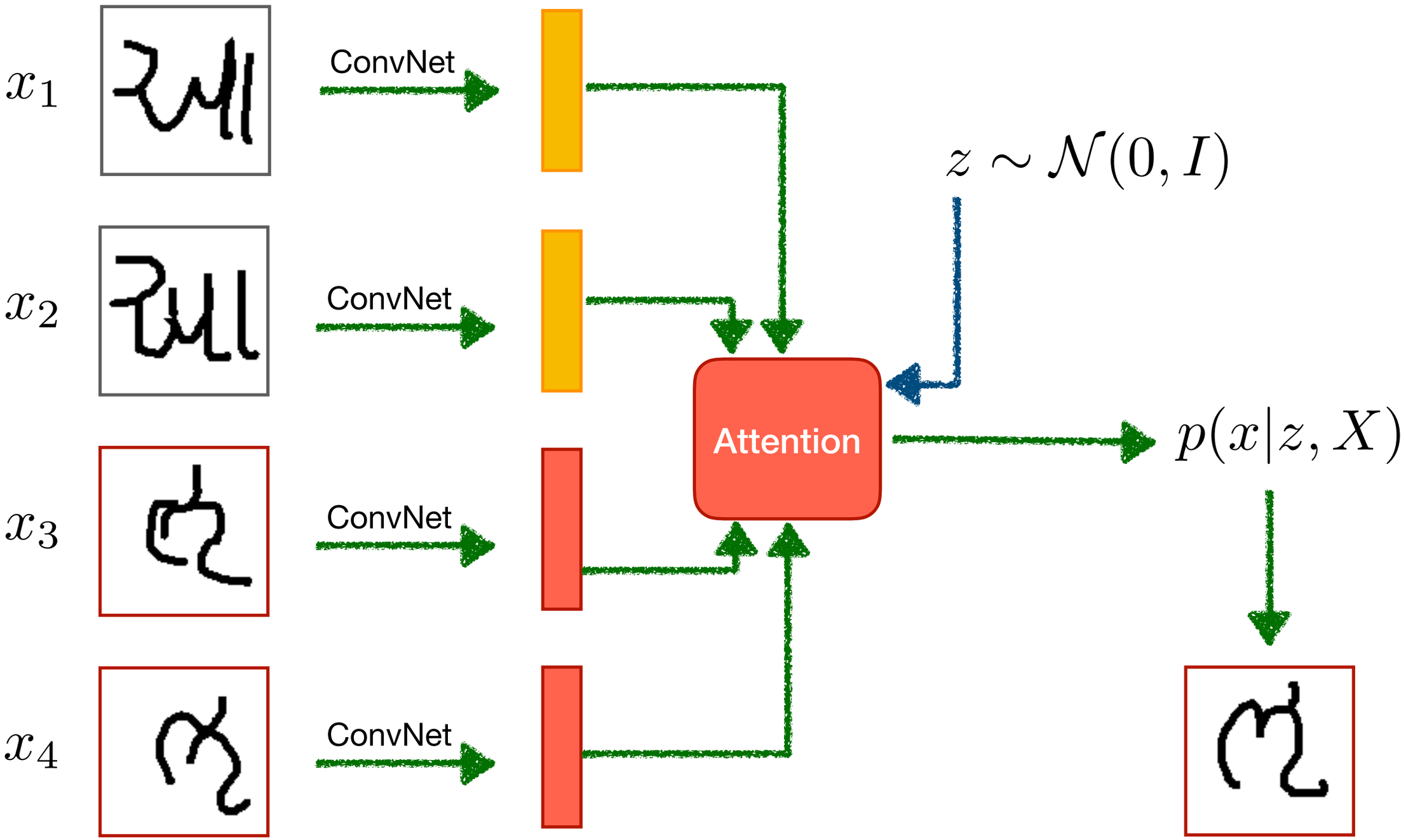
Generative Matching Networks

Bartunov & Vetrov, 2016



Generative Matching Networks

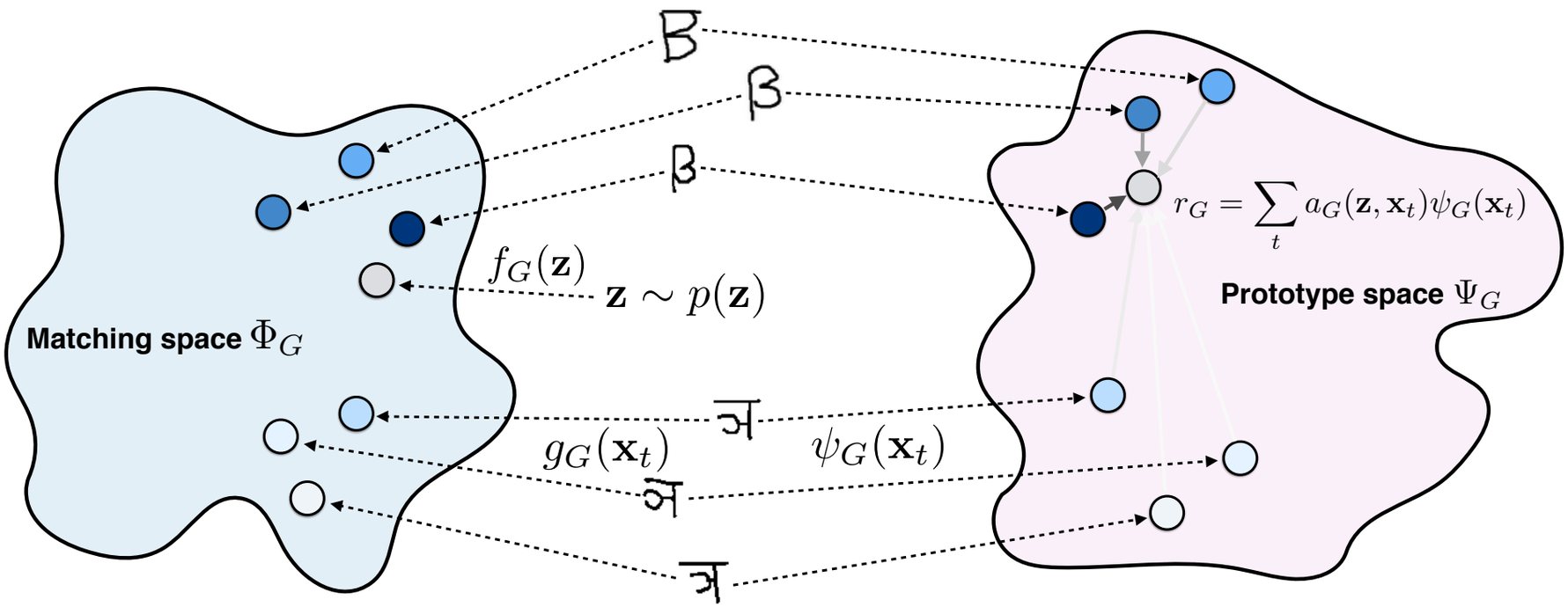
Bartunov & Vetrov, 2016



Generative Matching Networks

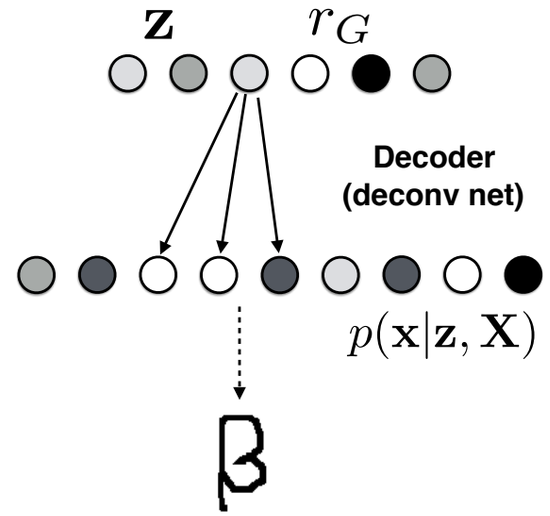
- An explicitly conditional variational autoencoder

$$p(\mathbf{x}|\mathbf{X}, \boldsymbol{\theta}) = \int \underbrace{p(\mathbf{z}|\mathbf{X}, \boldsymbol{\theta})}_{\text{prior}} \underbrace{p(\mathbf{x}|\mathbf{z}, \mathbf{X}, \boldsymbol{\theta})}_{\text{likelihood}} d\mathbf{z}$$



1. Computing attention kernel a_G

2. Aggregating prototypes



3. Decoding an observation

$$a_G(\mathbf{z}, \mathbf{x}_t) = \frac{\exp(\text{sim}(f_G(\mathbf{z}), g_G(\mathbf{x}_t)))}{\sum_{t'=1}^{T} \exp(\text{sim}(f_G(\mathbf{z}), g_G(\mathbf{x}_{t'})))}$$

Inspired by (Vinyals et al, 2016)

Extensions of GMN

- If no data is available, i.e. $\mathbf{X} = \emptyset$ we should still be able to generate something from our prior knowledge

- Define trainable parameters $g^* = g^*(\mathbf{x}^*), \psi^* = \psi^*(\mathbf{x}^*)$ as if there was always a pseudo-input $\mathbf{x}^*$

- Full context matching, allows to use higher-order comparisons between conditioning data and a query via several attentional steps

- Data-dependent prior $p(z|X)$

Training via meta-learning

Traditional (batch) learning

$$\log p(\mathbf{X}|\boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

Meta-learning

$$\mathbb{E}_{p_d(\mathbf{X})}\left[\log p(\mathbf{X}|\boldsymbol{\theta})\right] \to \max_{\boldsymbol{\theta}}$$

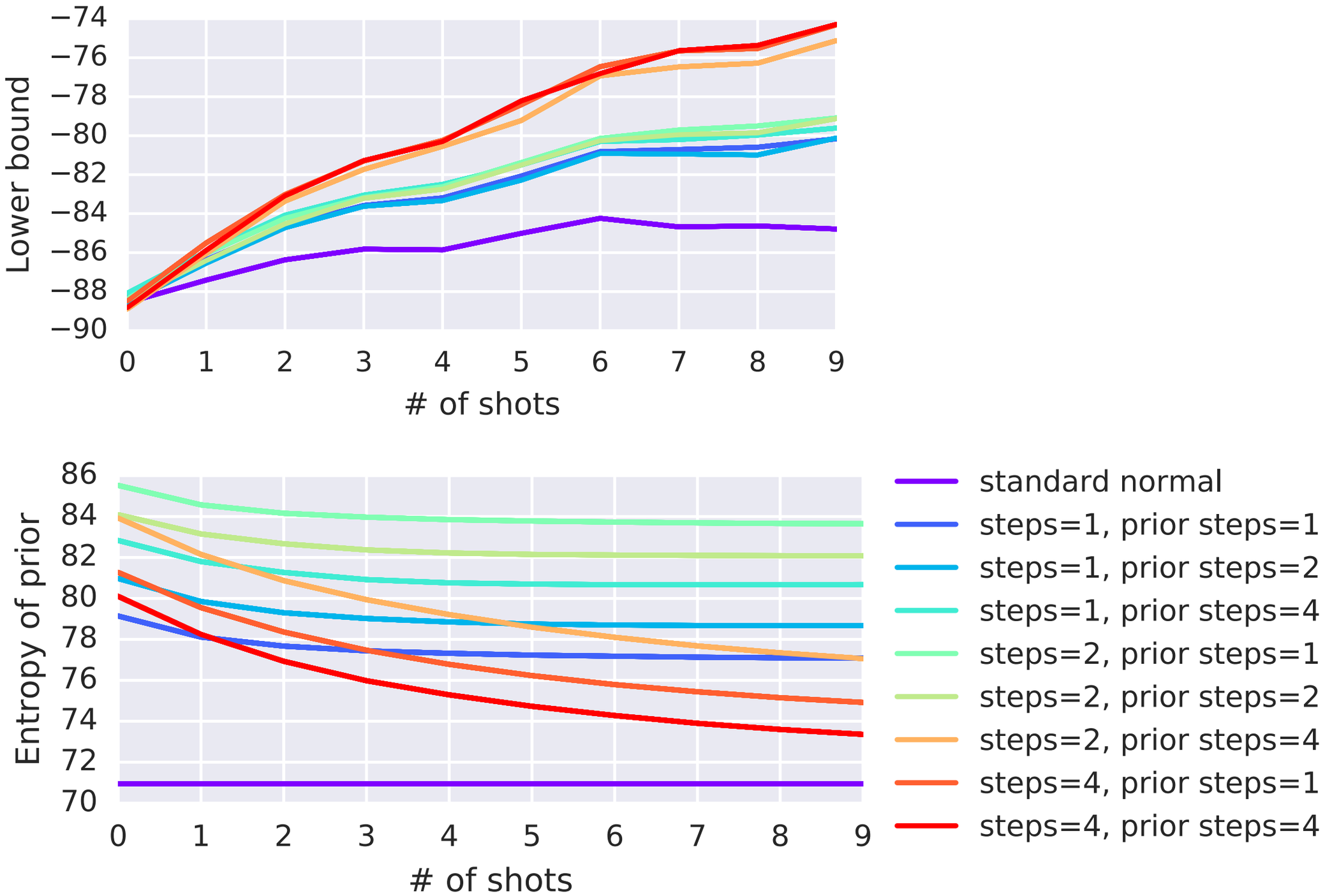
- Joint likelihood of a dataset is given by

$$p(\mathbf{X}|\boldsymbol{\theta}) = \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{X}_{<t}, \boldsymbol{\theta}), \quad \mathbf{X}_{<t} = \{\mathbf{x}_s\}_{s=1}^{t-1}$$

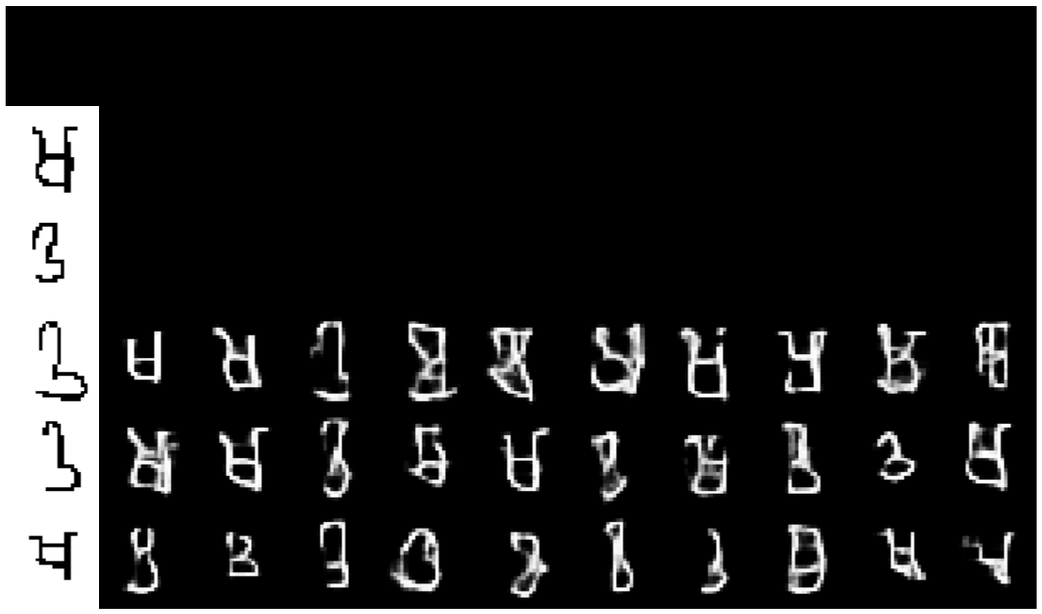
- Using all training data as **X** does not make sense

- We define a data-generating distribution $p_d(\mathbf{X})$, such that it's constrained to subsample datasets with C_{train} classes

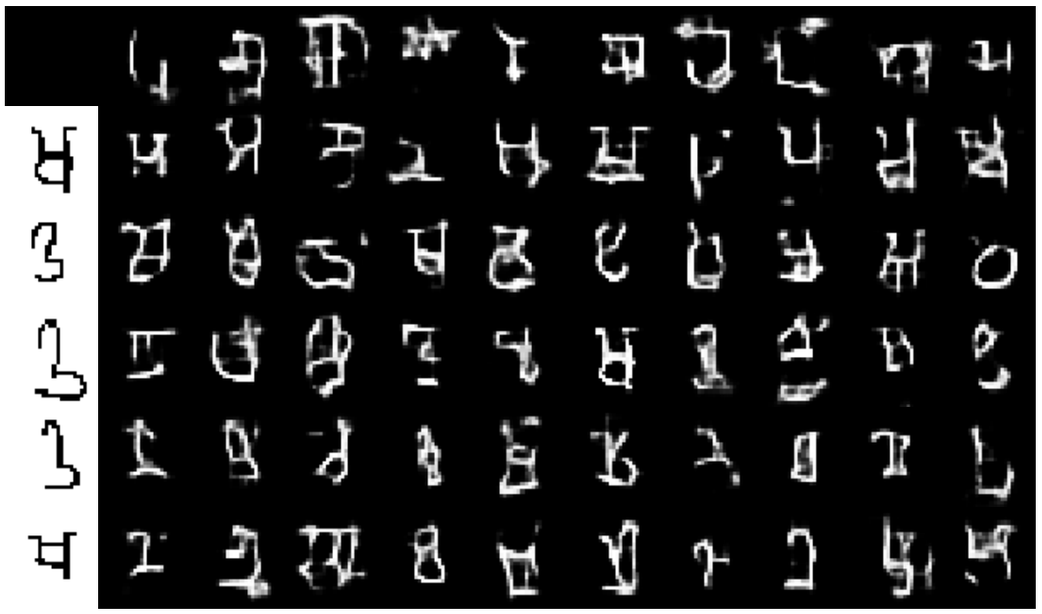
Effect of number of attentional steps



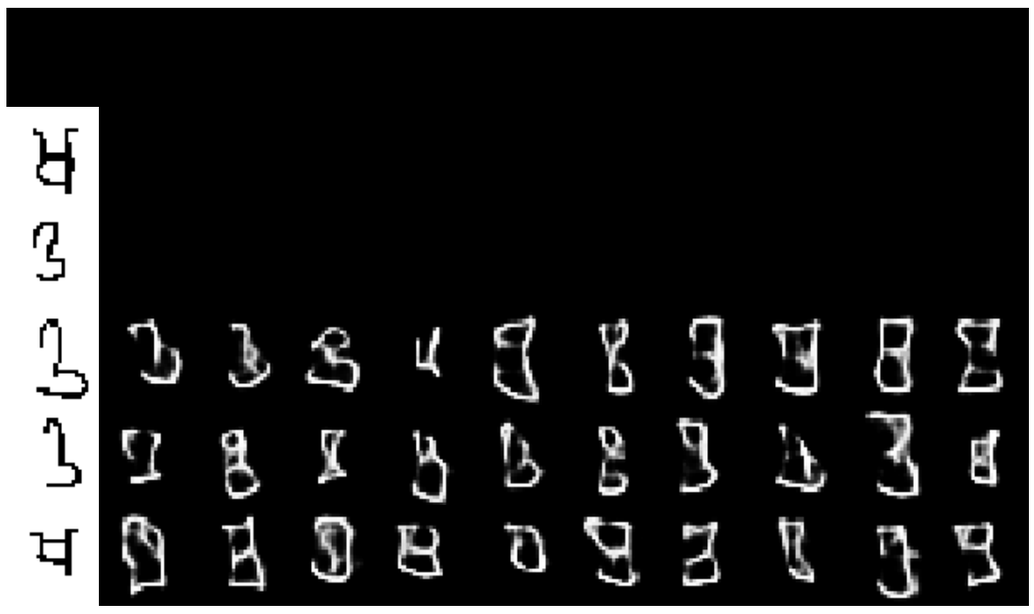
Generated samples



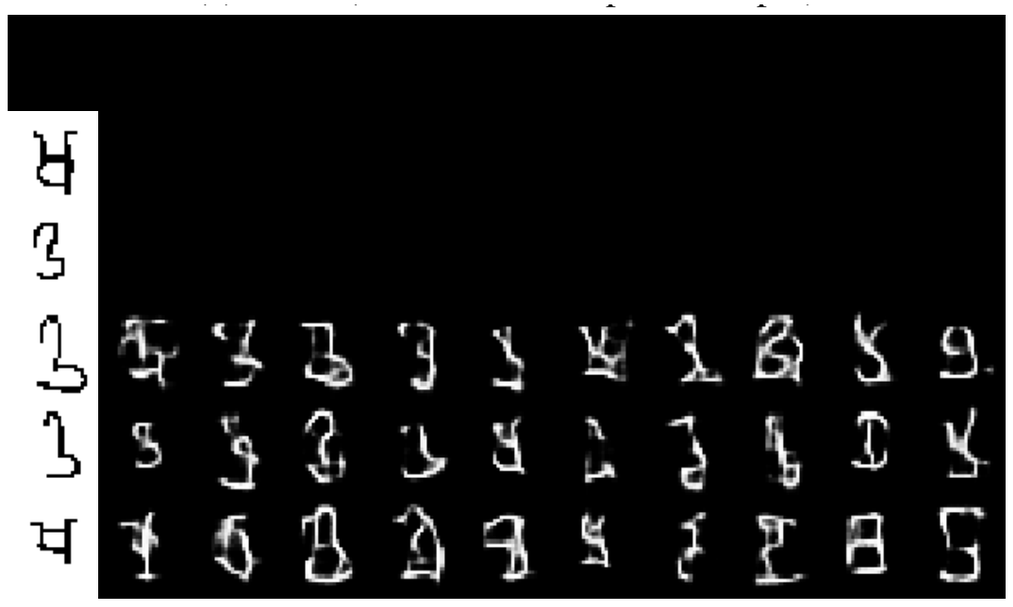
(a) GMN (no pseudo-input)



(b) GMN (one pseudo-input)



(c) GMN (no attention, no pseudo-input)



(d) Neural statistician

Test likelihood

MODEL	C_{test}	NUMBER OF CONDITIONING EXAMPLES 0	1	2	3	4	5	10	19
GMN, $C_{\text{train}} = 2$	1	**89.7**	**83.3**	**78.9**	**75.7**	**72.9**	**70.1**	**59.9**	**45.8**
GMN, $C_{\text{train}} = 2$	2	**89.4**	**86.4**	**84.9**	**82.4**	**81.0**	**78.8**	**71.4**	**61.2**
GMN, $C_{\text{train}} = 2$	3	**89.6**	**88.1**	**86.0**	**85.0**	**84.1**	**82.0**	**76.3**	**69.4**
GMN, $C_{\text{train}} = 2$	4	**89.3**	**88.3**	**87.3**	**86.7**	**85.4**	**84.0**	**80.2**	**73.7**
GMN, $C_{\text{train}} = 2$, no pseudo-input	1		93.5	82.2	78.6	76.8	75.0	69.7	64.3
GMN, $C_{\text{train}} = 2$, no pseudo-input	2			86.1	83.7	82.8	81.0	76.5	71.4
GMN, $C_{\text{train}} = 2$, no pseudo-input	3				86.1	84.7	83.8	79.7	75.3
GMN, $C_{\text{train}} = 2$, no pseudo-input	4					86.8	85.7	82.5	78.0
VAE		89.1							
One-shot VAE	1		83.9						
Neural statistician, $C_{\text{train}} = 1$	1		102	83.4	77.8	75.2	74.6	71.7	71.5
Neural statistician, $C_{\text{train}} = 2$	2			86.4	**82.2**	82.3	80.6	79.7	79.0
GMN, $C_{\text{train}} = 1$, no attention	1	92.4	84.5	82.3	81.4	81.1	80.4	79.8	79.7
GMN, $C_{\text{train}} = 2$, no attention	2	88.2	86.6	86.4	85.7	85.3	84.5	83.7	83.4
GMN, $C_{\text{train}} = 1$, no attention, no pseudo-input	1		88.0	84.1	82.9	82.4	81.7	80.9	80.7
GMN, $C_{\text{train}} = 2$, no attention, no pseudo-input	2			85.7	85.0	85.3	84.6	84.5	83.7

Classification

One-shot feature extraction

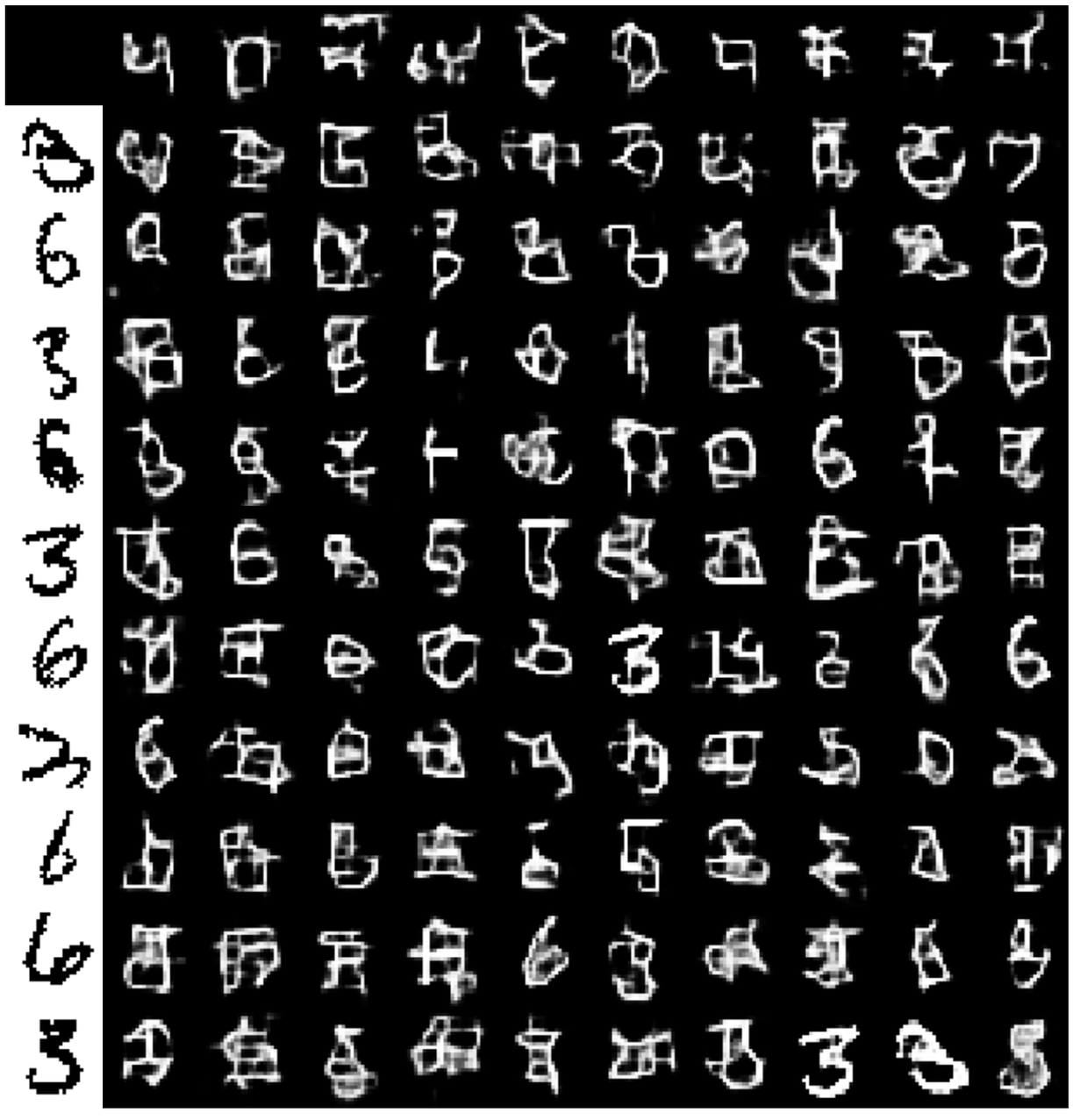
- Extract features using $q(z|X, x)$
- Use a nearest neighbor classifier (e.g. with a cosine distance)

Likelihood

- Define X_c as a training set for class c
- Use $\log p(x|X_c)$ as a score for class c

		5 CLASSES		20 CLASSES	
MODEL	METHOD	1-SHOT	5-SHOT	1-SHOT	5-SHOT
GMN	likelihood	82.7	**97.4**	64.3	**90.8**
GMN, no attention	likelihood	**90.8**	96.7	**77.0**	**91.0**
GMN	cosine	62.7	80.8	45.1	67.2
GMN, no attention	cosine	72.0	86.0	50.1	72.6
One-shot VAE	likelihood	**90.2**		76.3	
One-shot VAE	cosine	72.1		50.1	
Neural statistician	likelihood	82.0	94.8	63.1	87.6
Neural statistician	cosine	66.4	85.5	47.3	71.7
Matching networks, no fine-tuning	cosine	98.1	98.9	93.8	98.5

Transfer to MNIST

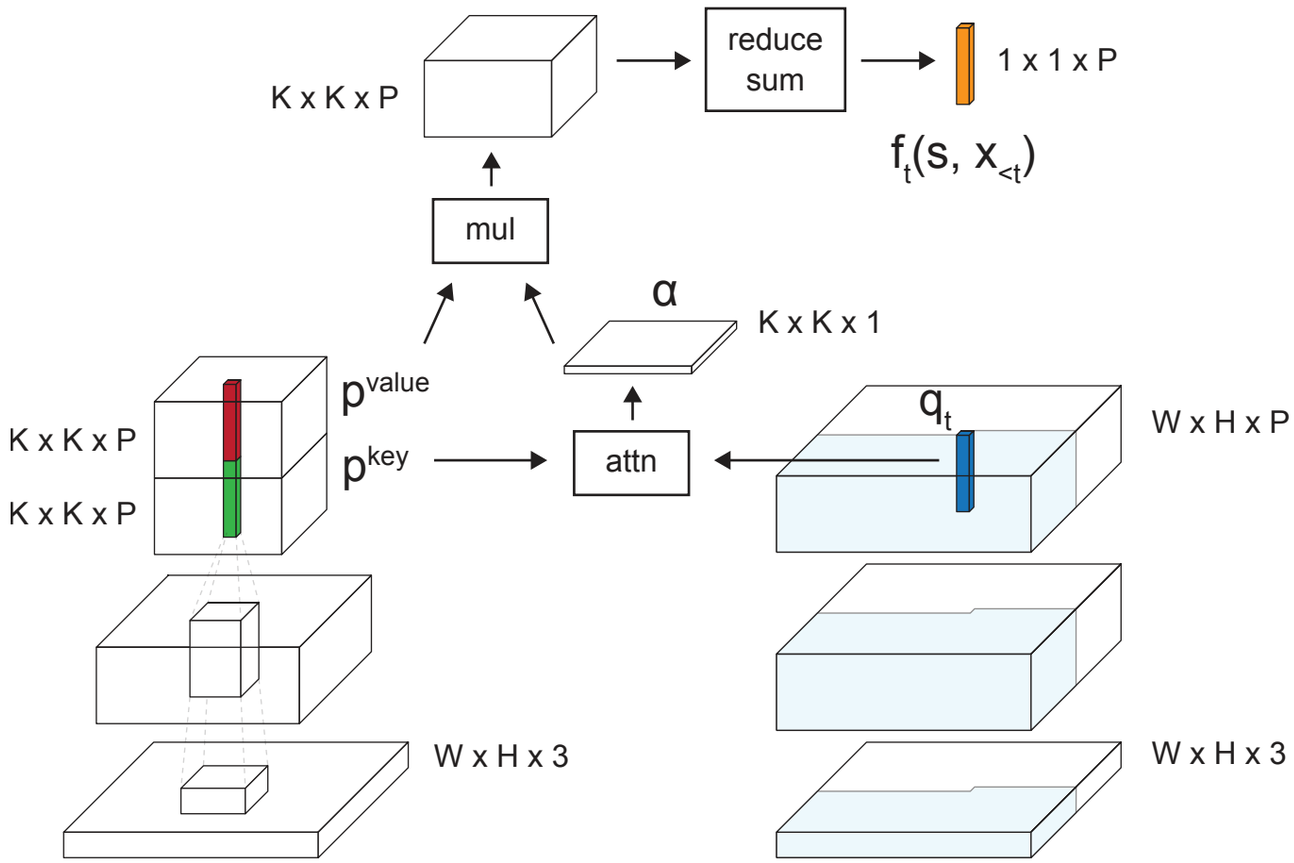


Transfer to MNIST

Model	C_{test}	Number of conditioning examples 0	1	2	3	4	5	10	19
GMN, $C_{\text{train}} = 2$	1	126.7	121.1	118.4	117.6	117.1	117.1	117.1	118.5
GMN, $C_{\text{train}} = 2$	2	126.2	123.1	121.3	120.1	119.4	118.9	118.3	119.6
GMN, $C_{\text{train}} = 2$, no pseudo-input	1		135.1	120.9	117.5	115.7	114.4	111.7	109.8
GMN, $C_{\text{train}} = 2$, no pseudo-input	2			123.1	121.9	119.4	118.8	115.2	113.2
GMN, $C_{\text{train}} = 1$, avg	1	131.5	126.5	123.3	121.9	121.0	120.2	118.6	117.5
GMN, $C_{\text{train}} = 2$, avg	2	126.2	122.8	121.0	119.9	118.9	118.7	117.8	116.8
GMN, $C_{\text{train}} = 1$, avg, no pseudo-input	1		132.1	126.9	125.0	124.8	123.9	121.7	120.9
GMN, $C_{\text{train}} = 2$, avg, no pseudo-input	2			118.4	117.9	117.4	117.1	116.6	115.8

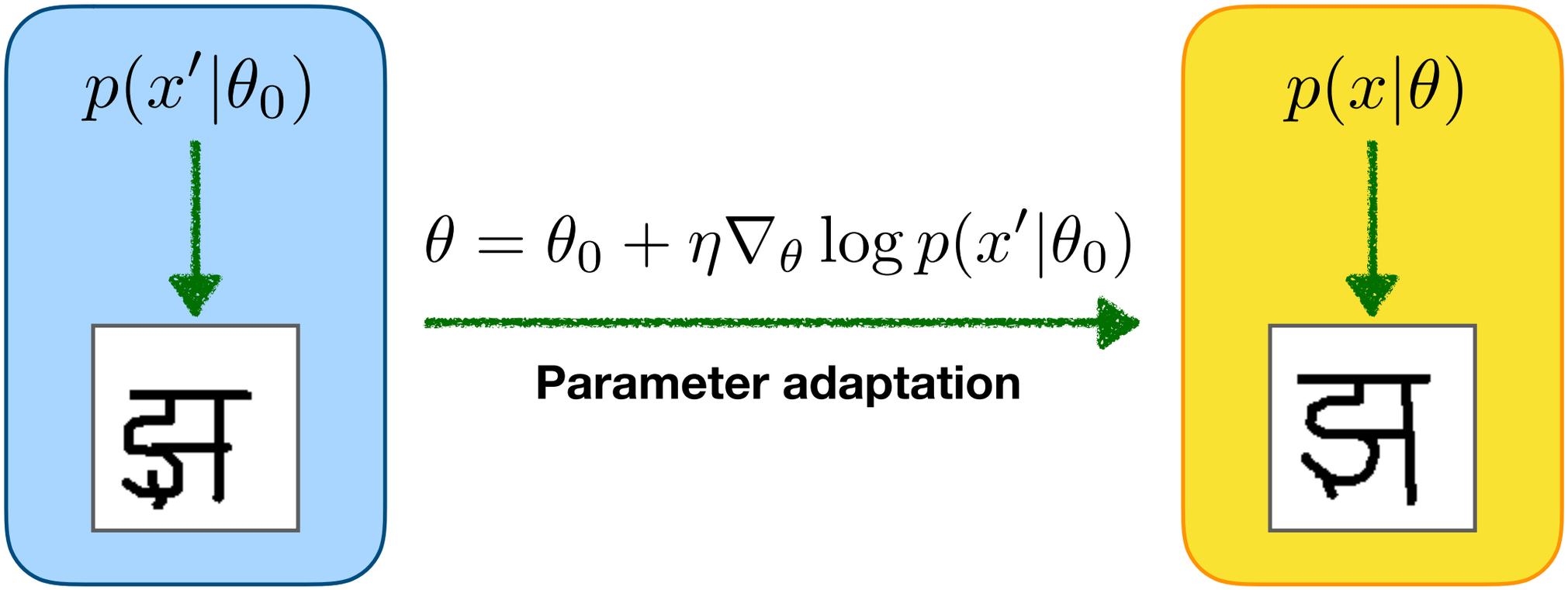
Few-shot Autoregressive Density Estimation

Reed et al, 2018



Parameter adaptation and gradient meta-learning

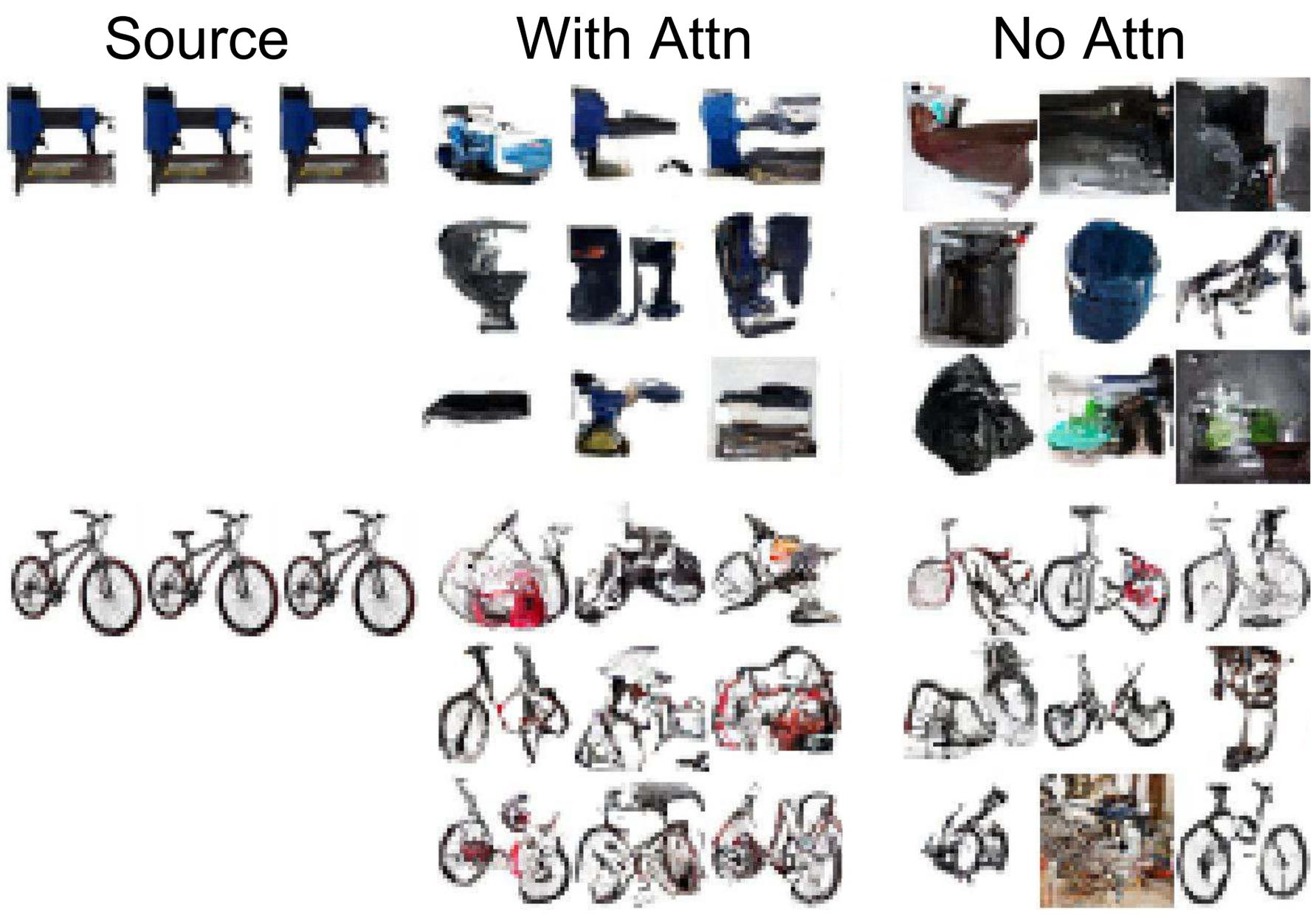
Finn et al, 2017; Reed et al, 2018



$$\mathbb{E}_{p_d(x,x')} \left[\log p(x|\theta) \right] \rightarrow \max_{\theta_0}$$

Parameter adaptation and gradient meta-learning

Finn et al, 2017; Reed et al, 2018



Parameter adaptation vs Memory-based

- No restrictions on the form of a generative model
- Constant space requirements
- Catastrophic forgetting / difficulties with incremental learning

- Need to design a specific architecture
- Need to store conditioning data
- Trivial implementation of incremental learning
- Currently state of the art

Future work

- Better architectures
- New methods for adaptation / meta-learning
- Memory structures
- More datasets

Thank you!