10707
Deep Learning

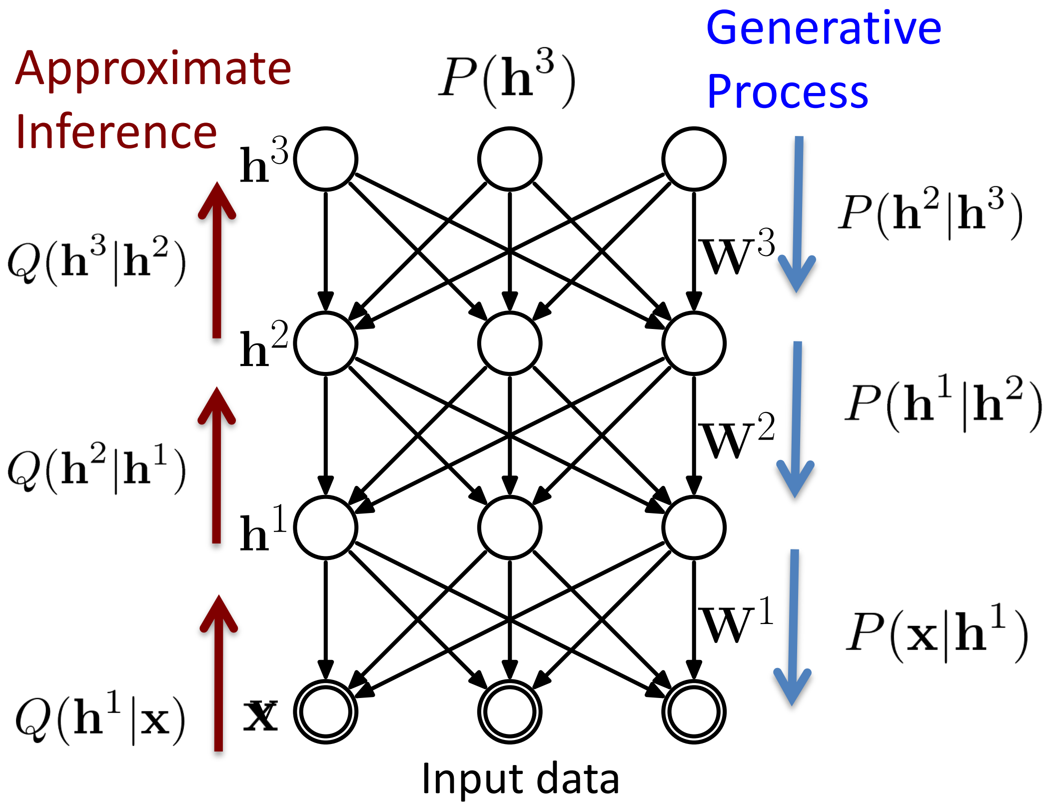
Russ Salakhutdinov

Machine Learning Department
rsalakhu@cs.cmu.edu

Variational Autoencoders

Variational Autoencoders (VAEs)

- Hinton, G. E., Dayan, P., Frey, B. J. and Neal, R., Science 1995



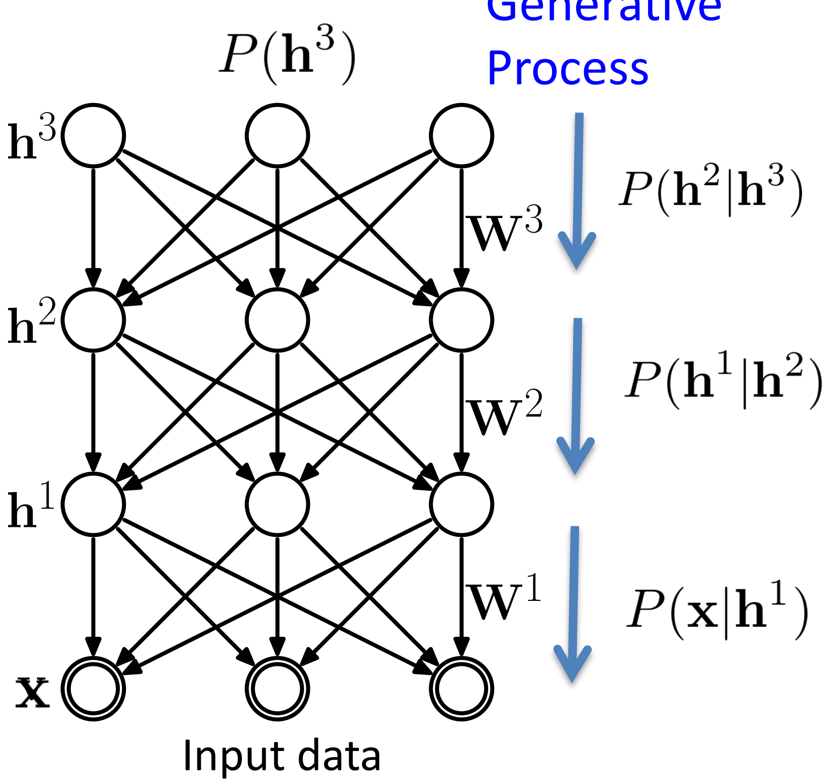
- Kingma & Welling, 2014
- Rezende, Mohamed, Daan, 2014
- Mnih & Gregor, 2014
- Bornschein & Bengio, 2015
- Tang & Salakhutdinov, 2013

Variational Autoencoders (VAEs)

- The VAE defines a generative process in terms of ancestral sampling through a cascade of hidden stochastic layers:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{h}^1,\ldots,\mathbf{h}^L} p(\mathbf{h}^L|\boldsymbol{\theta})p(\mathbf{h}^{L-1}|\mathbf{h}^L,\boldsymbol{\theta})\cdots p(\mathbf{x}|\mathbf{h}^1,\boldsymbol{\theta})$$

Each term may denote a complicated nonlinear relationship



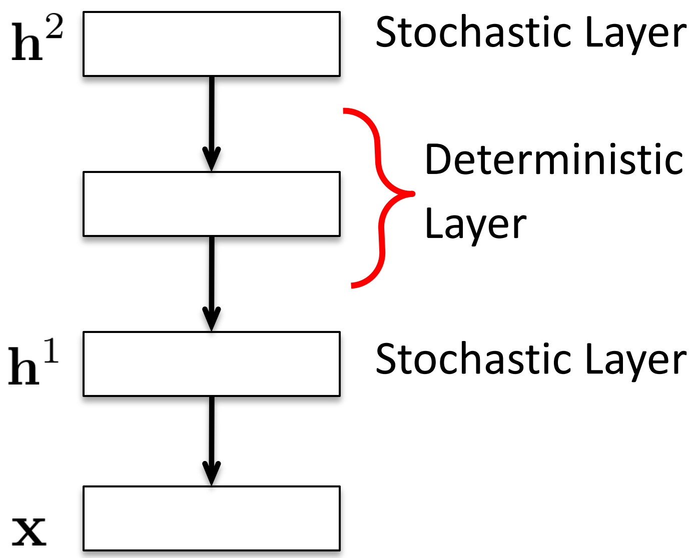
- $\boldsymbol{\theta}$ denotes parameters of VAE.
- L is the number of **stochastic** layers.
- Sampling and probability evaluation is tractable for each $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$.

VAE: Example

- The VAE defines a generative process in terms of ancestral sampling through a cascade of hidden stochastic layers:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{h}^1,\mathbf{h}^2} p(\mathbf{h}^2|\boldsymbol{\theta})p(\mathbf{h}^1|\mathbf{h}^2,\boldsymbol{\theta})p(\mathbf{x}|\mathbf{h}^1,\boldsymbol{\theta})$$

This term denotes a one-layer neural net.



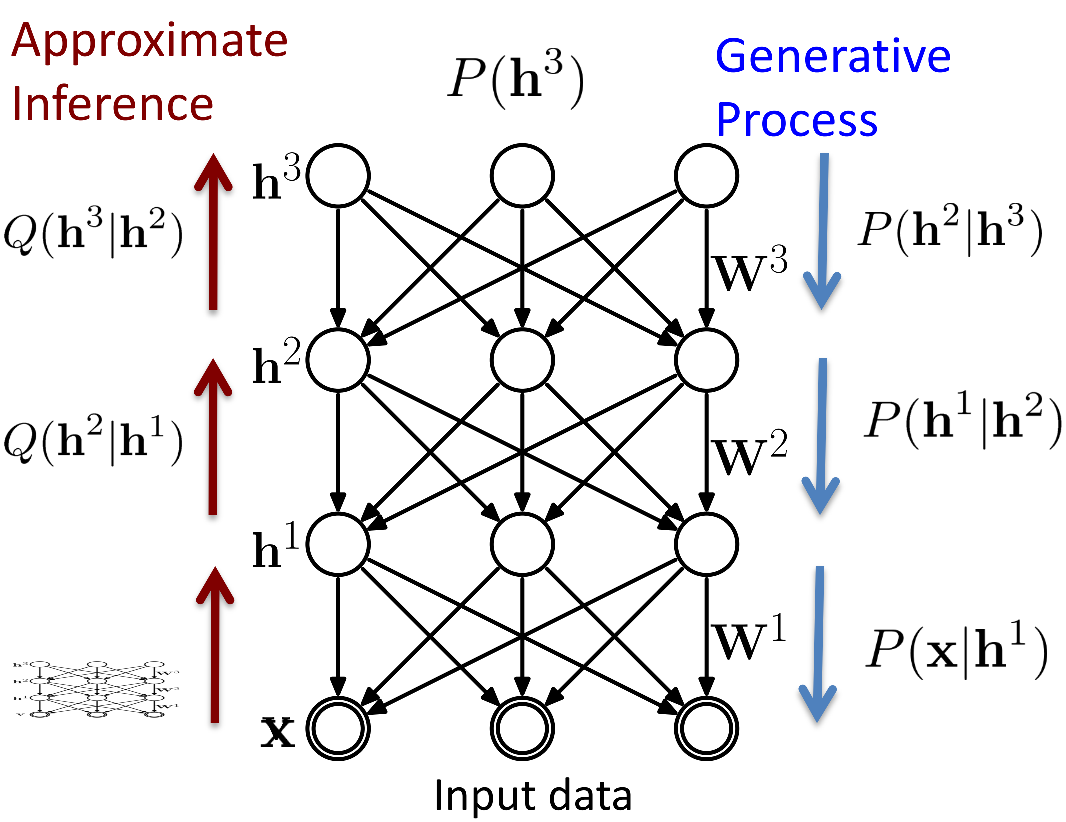
- $\boldsymbol{\theta}$ denotes parameters of VAE.
- L is the number of **stochastic** layers.
- Sampling and probability evaluation is tractable for each $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$.

Recognition Network

- The recognition model is defined in terms of an analogous factorization:

$$q(\mathbf{h}|\mathbf{x},\boldsymbol{\theta}) = q(\mathbf{h}^1|\mathbf{x},\boldsymbol{\theta})q(\mathbf{h}^2|\mathbf{h}^1,\boldsymbol{\theta})\cdots q(\mathbf{h}^L|\mathbf{h}^{L-1},\boldsymbol{\theta})$$

Each term may denote a complicated nonlinear relationship



- We assume that $\mathbf{h}^L \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

- The conditionals:

$$p(\mathbf{h}^\ell|\ \mathbf{h}^{\ell+1})$$
$$q(\mathbf{h}^\ell|\mathbf{h}^{\ell-1})$$

are Gaussians with diagonal covariances

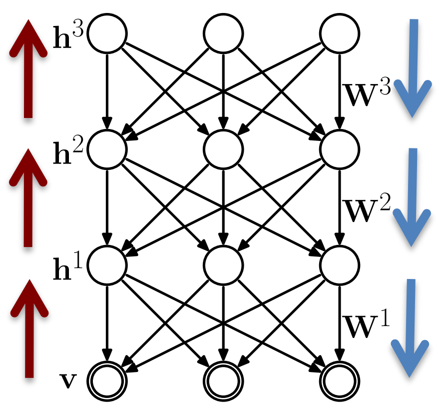
Variational Bound

- The VAE is trained to maximize the variational lower bound:

$$\log p(\mathbf{x}) = \log \mathbb{E}_{q(\mathbf{h}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{h})}{q(\mathbf{h}|\mathbf{x})} \right] \geq \mathbb{E}_{q(\mathbf{h}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{h})}{q(\mathbf{h}|\mathbf{x})} \right] = \mathcal{L}(\mathbf{x})$$

$$\mathcal{L}(\mathbf{x}) = \log p(\mathbf{x}) - \mathrm{D}_{\mathrm{KL}} \left(q(\mathbf{h}|\mathbf{x}) \right) || p(\mathbf{h}|\mathbf{x}))$$

- Trading off the data log-likelihood and the KL divergence from the true posterior.



- Hard to optimize the variational bound with respect to the recognition network (high-variance).
- Key idea of Kingma and Welling is to use reparameterization trick.

Reparameterization Trick

- Assume that the recognition distribution is Gaussian:

$$q(\mathbf{h}^{\ell}|\mathbf{h}^{\ell-1},\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{h}^{\ell-1},\boldsymbol{\theta}), \boldsymbol{\Sigma}(\mathbf{h}^{\ell-1},\boldsymbol{\theta}))$$

  with mean and covariance computed from the state of the hidden units at the previous layer.

- Alternatively, we can express this in term of auxiliary variable:

$$\boldsymbol{\epsilon}^{\ell} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

$$\mathbf{h}^{\ell}\left(\boldsymbol{\epsilon}^{\ell}, \mathbf{h}^{\ell-1}, \boldsymbol{\theta}\right) = \boldsymbol{\Sigma}(\mathbf{h}^{\ell-1},\boldsymbol{\theta})^{1/2}\boldsymbol{\epsilon}^{\ell} + \boldsymbol{\mu}(\mathbf{h}^{\ell-1},\boldsymbol{\theta})$$

Reparameterization Trick

- Assume that the recognition distribution is Gaussian:

$$q(\mathbf{h}^\ell | \mathbf{h}^{\ell-1}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{h}^{\ell-1}, \boldsymbol{\theta}), \boldsymbol{\Sigma}(\mathbf{h}^{\ell-1}, \boldsymbol{\theta}))$$

- Or

$$\boldsymbol{\epsilon}^\ell \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

$$\mathbf{h}^\ell \left(\boldsymbol{\epsilon}^\ell, \mathbf{h}^{\ell-1}, \boldsymbol{\theta}\right) = \boldsymbol{\Sigma}(\mathbf{h}^{\ell-1}, \boldsymbol{\theta})^{1/2} \boldsymbol{\epsilon}^\ell + \boldsymbol{\mu}(\mathbf{h}^{\ell-1}, \boldsymbol{\theta})$$

- The recognition distribution $q(\mathbf{h}^\ell | \mathbf{h}^{\ell-1}, \boldsymbol{\theta})$ can be expressed in terms of a deterministic mapping:

$$\underbrace{\mathbf{h}\left(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta}\right)}_{\text{Deterministic Encoder}}, \quad \text{with} \quad \boldsymbol{\epsilon} = \underbrace{(\boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^L)}_{\text{Distribution of } \boldsymbol{\epsilon} \text{ does not depend on } \boldsymbol{\theta}}$$

Deterministic Encoder

Distribution of $\boldsymbol{\epsilon}$ does not depend on $\boldsymbol{\theta}$

Computing the Gradients

- The gradient w.r.t the parameters: both recognition and generative:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{h} \sim q(\mathbf{h}|\mathbf{x},\boldsymbol{\theta})} \left[\log \frac{p(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta})}{q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta})} \right]$$

$$= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^L \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\log \frac{p(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\mathbf{x}, \boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^L \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\mathbf{x}, \boldsymbol{\theta})} \right]$$

Gradients can be computed by backprop

The mapping **h** is a deterministic neural net for fixed $\boldsymbol{\epsilon}$.

Computing the Gradients

- The gradient w.r.t the parameters: recognition and generative:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{h} \sim q(\mathbf{h}|\mathbf{x},\boldsymbol{\theta})} \left[\log \frac{p(\mathbf{x},\mathbf{h}|\boldsymbol{\theta})}{q(\mathbf{h}|\mathbf{x},\boldsymbol{\theta})} \right] = \mathbb{E}_{\boldsymbol{\epsilon}^1,\ldots,\boldsymbol{\epsilon}^L \sim \mathcal{N}(\mathbf{0},\boldsymbol{I})} \left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\mathbf{x},\mathbf{h}(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta})|\mathbf{x},\boldsymbol{\theta})} \right]$$

- Approximate expectation by generating k samples from $\boldsymbol{\epsilon}$:

$$\frac{1}{k} \sum_{i=1}^{k} \nabla_{\boldsymbol{\theta}} \log w\left(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta}\right)$$

where we defined unnormalized importance weights:

$$w(\mathbf{x},\mathbf{h},\boldsymbol{\theta}) = p(\mathbf{x},\mathbf{h}|\boldsymbol{\theta})/q(\mathbf{h}|\mathbf{x},\boldsymbol{\theta})$$

- VAE update: Low variance as it uses the log-likelihood gradients with respect to the latent variables.

VAE: Assumptions

- Remember the variational bound:

$$\mathcal{L}(\mathbf{x}) = \log p(\mathbf{x}) - \mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{h}|\mathbf{x})\right)||p(\mathbf{h}|\mathbf{x}))$$

- The variational assumptions must be approximately satisfied.

- The posterior distribution must be approximately factorial (common practice) and predictable with a feed-forward net.

- We show that we can relax these assumptions using a tighter lower bound on marginal log-likelihood.

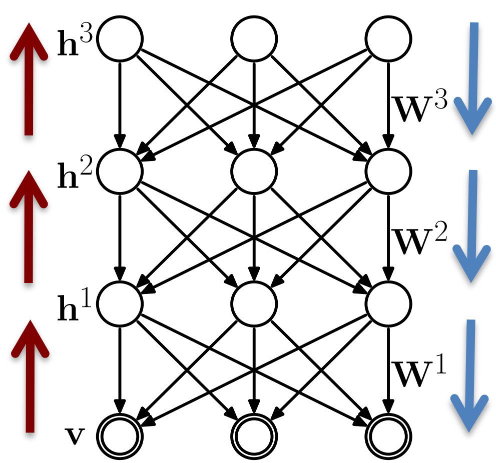
Importance Weighted Autoencoders

- Consider the following k-sample importance weighting of the log-likelihood:

$$\mathcal{L}_k(\mathbf{x}) = \mathbb{E}_{\mathbf{h}_1,\ldots,\mathbf{h}_k \sim q(\mathbf{h}|\mathbf{x})} \left[\log \frac{1}{k} \sum_{i=1}^{k} \frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})} \right]$$

$$= \mathbb{E}_{\mathbf{h}_1,\ldots,\mathbf{h}_k \sim q(\mathbf{h}|\mathbf{x})} \left[\log \frac{1}{k} \sum_{i=1}^{k} w_i \right]$$

unnormalized importance weights



Input data

where $\mathbf{h}_1, \ldots, \mathbf{h}_k$ are sampled from the recognition network.

Importance Weighted Autoencoders

- Consider the following k-sample importance weighting of the log-likelihood:

$$\mathcal{L}_k(\mathbf{x}) = \mathbb{E}_{\mathbf{h}_1,\ldots,\mathbf{h}_k \sim q(\mathbf{h}|\mathbf{x})}\left[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p(\mathbf{x},\mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})}\right]$$

- This is a lower bound on the marginal log-likelihood:

$$\mathcal{L}_k(\mathbf{x}) = \mathbb{E}\left[\log \frac{1}{k}\sum_{i=1}^{k} w_i\right] \le \log \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} w_i\right] = \log p(\mathbf{x})$$

- Special Case of k=1: Same as standard VAE objective.

- Using more samples → Improves the tightness of the bound.

Tighter Lower Bound

- Using more samples can only improve the tightness of the bound.

- For all k, the lower bounds satisfy:

$$\log p(\mathbf{x}) \geq \mathcal{L}_{k+1}(\mathbf{x}) \geq \mathcal{L}_k(\mathbf{x})$$

- Moreover if $p(\mathbf{h}, \mathbf{x})/q(\mathbf{h}|\mathbf{x})$ is bounded, then:

$$\mathcal{L}_k(\mathbf{x}) \rightarrow \log p(\mathbf{x}), \quad \text{as} \quad k \rightarrow \infty$$

Computing the Gradients

- We can use the unbiased estimate of the gradient using reparameterization trick:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_k(\mathbf{x}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{h}_1,\ldots,\mathbf{h}_k \sim q(\mathbf{h}|\mathbf{x})} \left[\log \frac{1}{k} \sum_{i=1}^{k} w_i \right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}_1,\ldots,\boldsymbol{\epsilon}_k} \left[\nabla_{\boldsymbol{\theta}} \log \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, h(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta}) \right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}_1,\ldots,\boldsymbol{\epsilon}_k} \left[\sum_{i=1}^{k} \widetilde{w}_i \nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, h(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta}) \right]$$

where we define normalized importance weights:

$$\widetilde{w}_i = w_i / \sum_{i=1}^{k} w_i, \quad \text{where } w_i = \frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})}$$

IWAEs vs. VAEs

- Draw k-samples form the recognition network $q(\mathbf{h}|\mathbf{x})$
  - or k-sets of auxiliary variables $\boldsymbol{\epsilon}$.
- Obtain the following Monte Carlo estimate of the gradient:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_k(\mathbf{x}) \approx \sum_{i=1}^{k} \widetilde{w}_i \boxed{\nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta})}$$

- Compare this to the VAE's estimate of the gradient:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{x}) \approx \frac{1}{k} \sum_{i=1}^{k} \boxed{\nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta})}$$

IWAE: Intuition

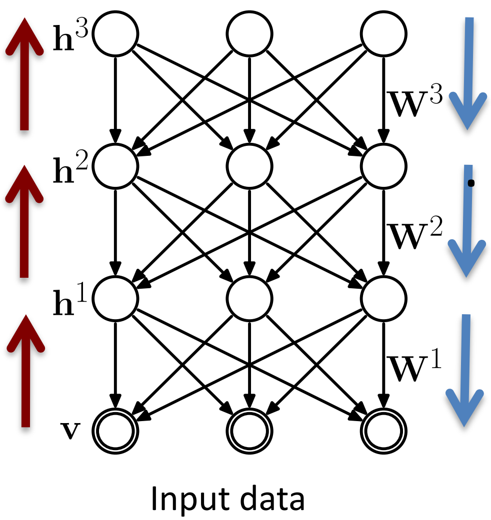
- The gradient of the log weights decomposes:

$$\nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta})$$

$$= \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta})|\boldsymbol{\theta}) - \log q(\mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta})|\mathbf{x}, \boldsymbol{\theta})$$

Deterministic decoder

Deterministic Encoder



First term:

- Decoder: encourages the generative model to assign high probability to each $\mathbf{h}^l|\mathbf{h}^{l+1}$.
- Encoder: encourages the recognition net to adjust its latent states h so that the generative network makes better predictions.

IWAE: Intuition

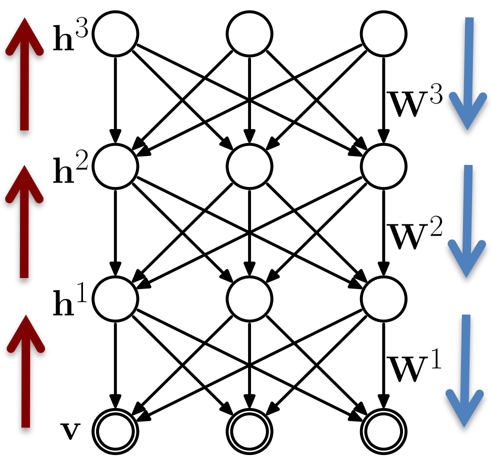
- The gradient of the log weights decomposes:

$$\nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta})$$

$$= \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}) | \boldsymbol{\theta}) - \log q(\mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}) | \mathbf{x}, \boldsymbol{\theta})$$

Deterministic decoder

Deterministic Encoder



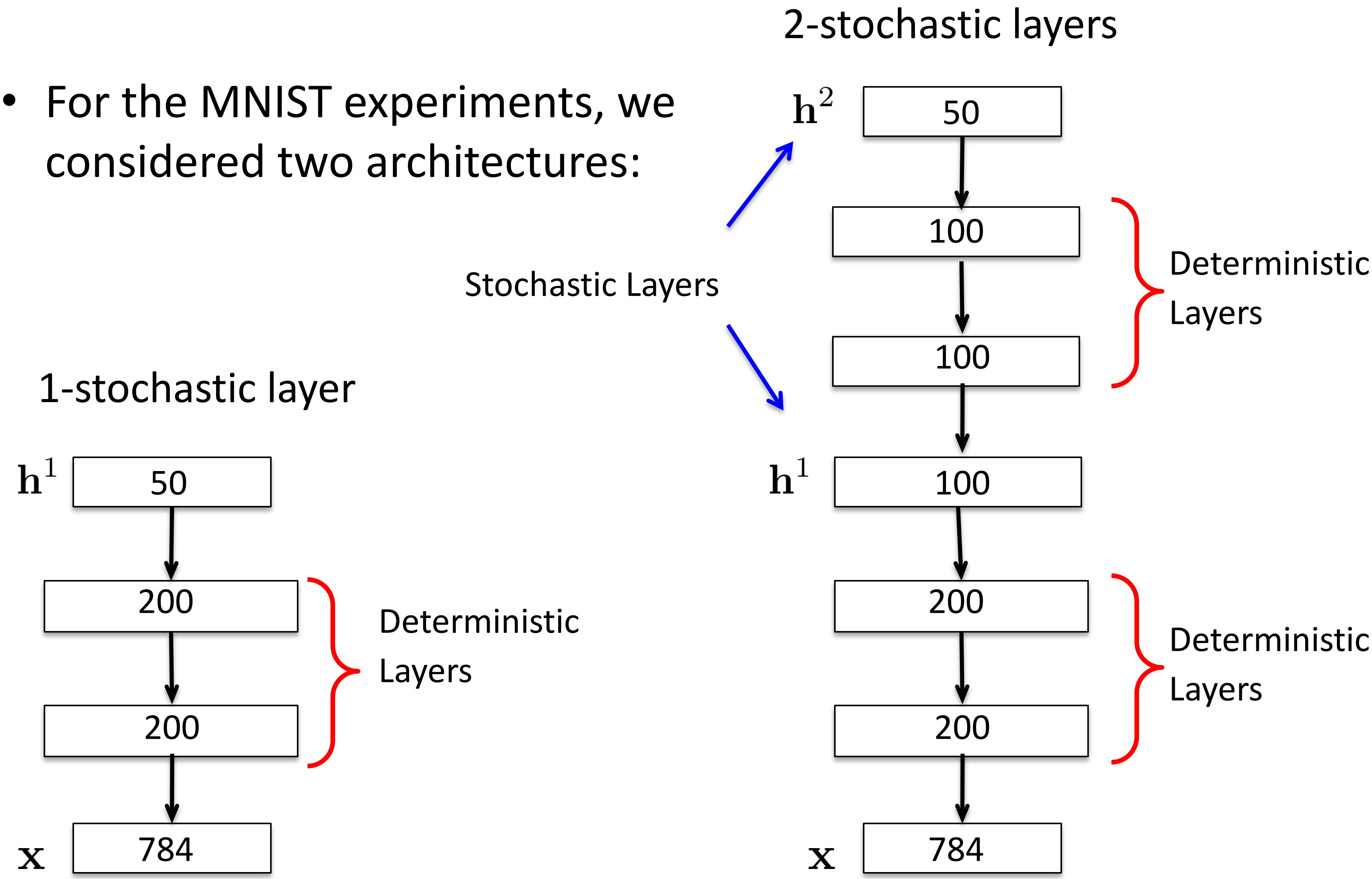
Input data

Second term:

– Encoder: encourages the recognition network to have a spread-out distribution over predictions.

Two Architectures

- For the MNIST experiments, we considered two architectures:



MNIST Results

		MNIST			
		VAE		IWAE	
# stoch. layers	k	NLL	active units	NLL	active units
1	1	86.76	19	86.76	19
	5	86.47	20	85.54	22
	50	86.35	20	84.78	25

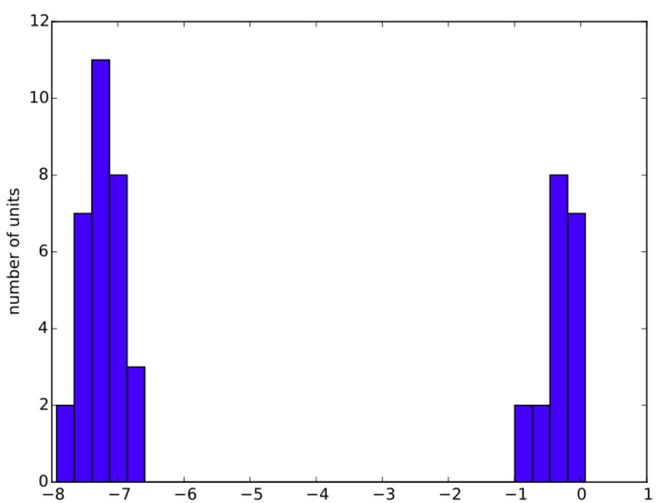
MNIST Results

		MNIST			
		VAE		IWAE	
# stoch. layers	k	NLL	active units	NLL	active units
1	1	86.76	19	86.76	19
	5	86.47	20	85.54	22
	50	86.35	20	84.78	25
2	1	85.33	16+5	85.33	16+5
	5	85.01	17+5	83.89	21+5
	50	84.78	17+5	82.90	26+7

Latent Space Representation

- Both VAEs and IWAEs tend to learn latent representations with effective dimensions far below their capacity.
- Measure the activity of the latent dimension u using the statistics:

$$A_u = \mathrm{Cov}_{\mathbf{x}}\left(\mathbb{E}_{u\sim q(u|\mathbf{x})}[u]\right)$$



- The distribution of $\log A_u$ consist of two separated modes.
- Inactive dimensions → units dying out.
- Optimization issue?

IWAEs vs. VAEs

First stage

trained as	NLL	active units
VAE	86.76	19
IWAE, $k = 50$	84.78	25

IWAEs vs. VAEs

First stage

trained as	NLL	active units
VAE	86.76	19
IWAE, $k = 50$	84.78	25

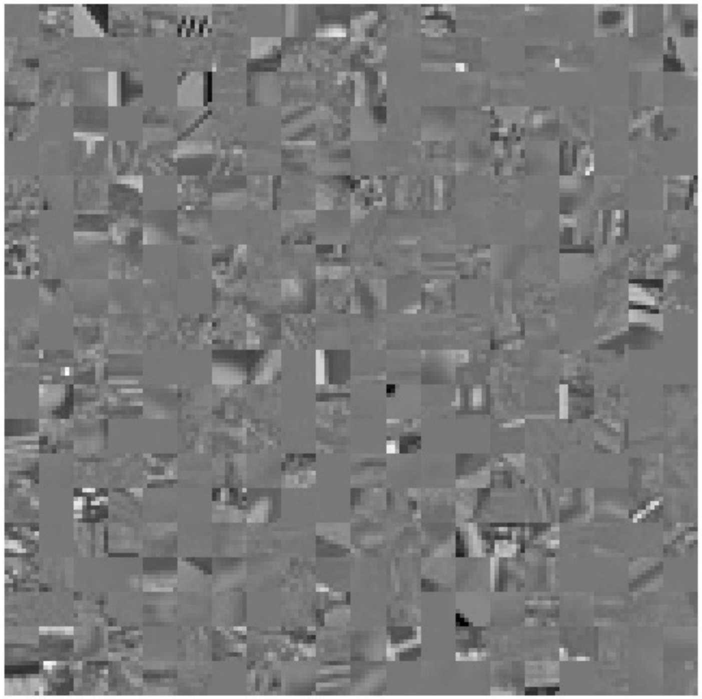
Second stage

trained as	NLL	active units
IWAE, $k = 50$	84.88	22
VAE	86.02	23

OMNIGLOT Experiments

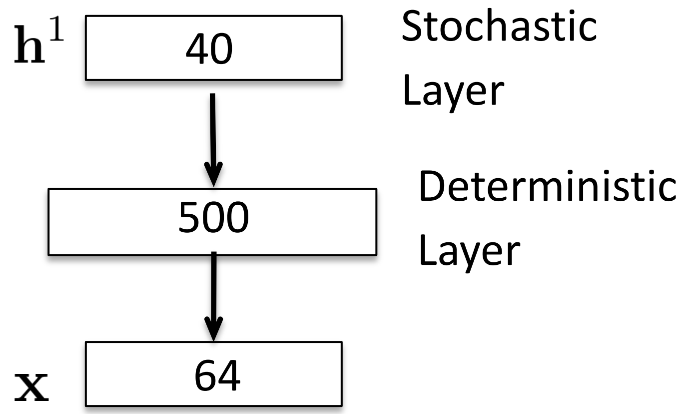
		OMNIGLOT			
		VAE		IWAE	
# stoch. layers	k	NLL	active units	NLL	active units
1	1	108.11	28	108.11	28
	5	107.62	28	106.12	34
	50	107.80	28	104.67	41
2	1	107.58	28+4	107.56	30+5
	5	106.31	30+5	104.79	38+6
	50	106.30	30+5	103.38	44+7

Modeling Image Patches
BSDS Dataset



- Model 8x8 patches.

1-stochastic layer



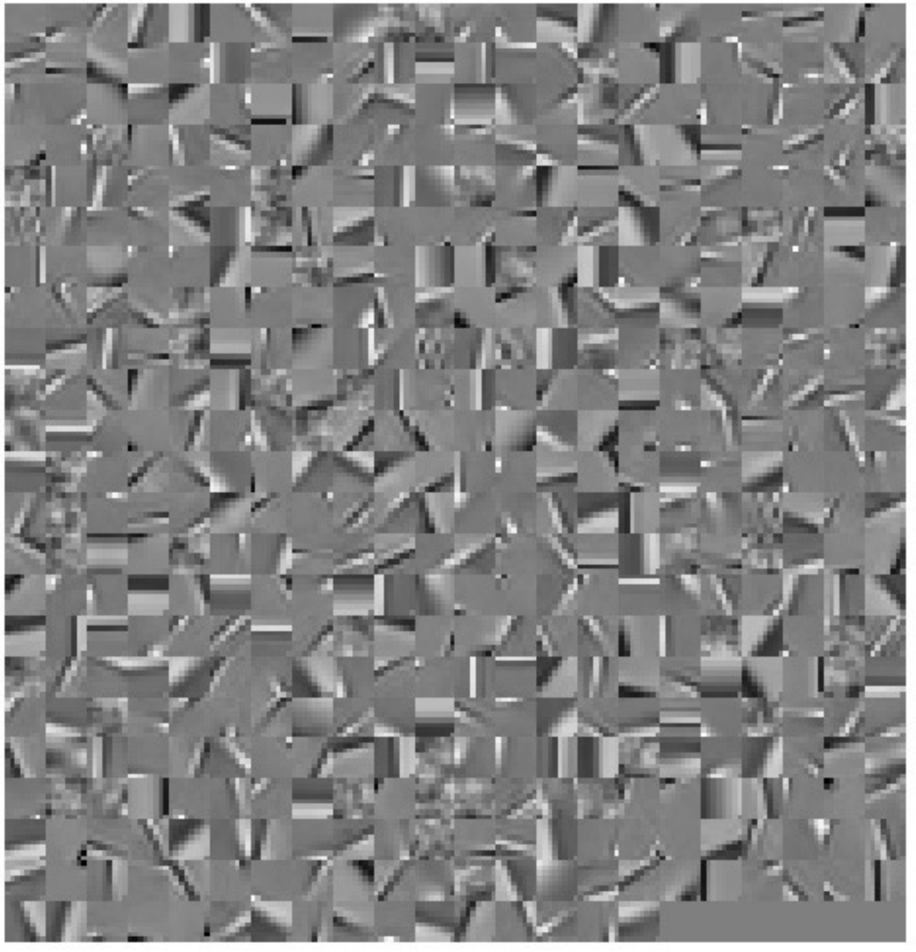
- Report test log-likelihoods on 10^6 8x8 patches, extracted from BSDS test dataset.
- Evaluation protocol established by Uria, Murray and Larochelle):
  - add uniform noise between 0 and 1, divide by 256,
  - subtract the mean and discarding the last pixel

Test Log-probabilities

Model	nats	Bits/pixel
RNADE 6 hidden layers (Uria et. al. 2013)	155.2 nats	3.55 bit/pixel
MoG, 200 full-covariance mixture (Zoran and Weiss, 2012)	152.8 nats	3.50 bit/pixel
IWAE (k=500)	151.4 nats	3.47 bit/pixel
VAE (k=500)	148.0 nats	3.39 bit/pixel
GSM (Gaussian Scale Mixture)	142 nats	3.25 bit/pixel
ICA	111 nats	2.54 bit/pixel
PCA	96 nats	2.21 bit/pixel

Burda 2015

Learned Filters



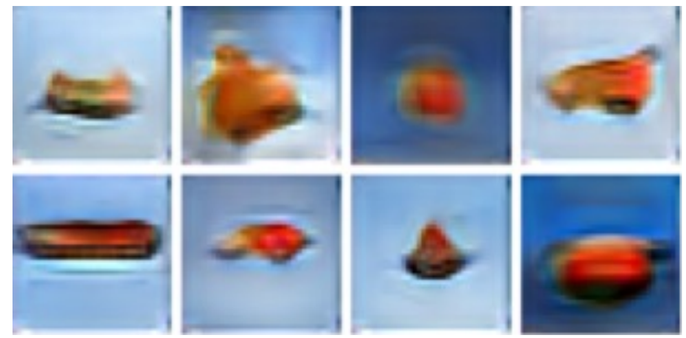
Burda 2015

Motivating Example

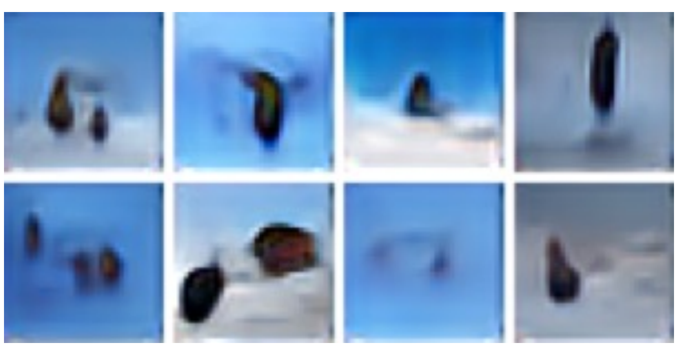
- Can we generate images from natural language descriptions?

A **stop sign** is flying in blue skies

A **pale yellow school bus** is flying in blue skies

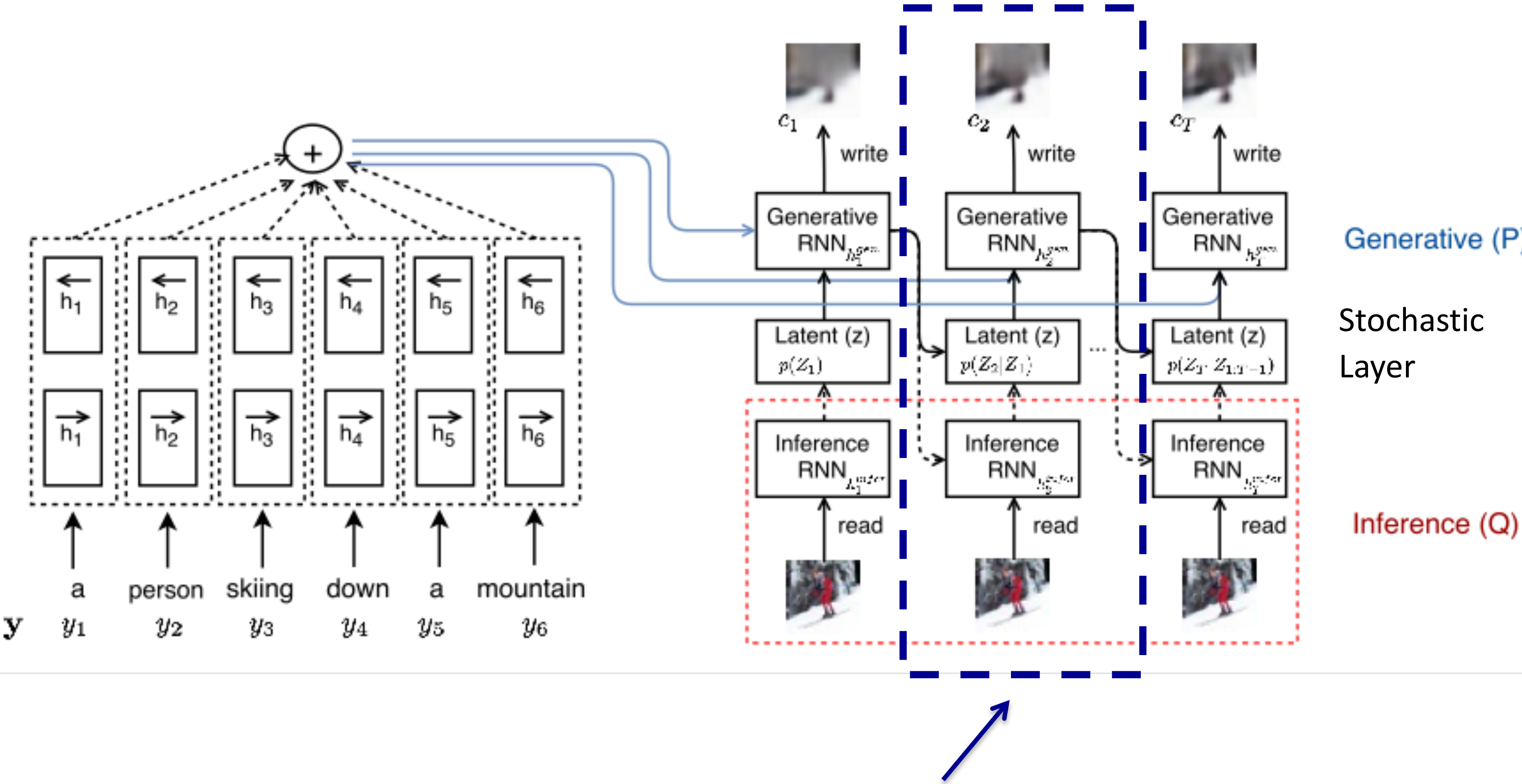


A **herd of elephants** is flying in blue skies



A **large commercial airplane** is flying in blue skies

Overall Model

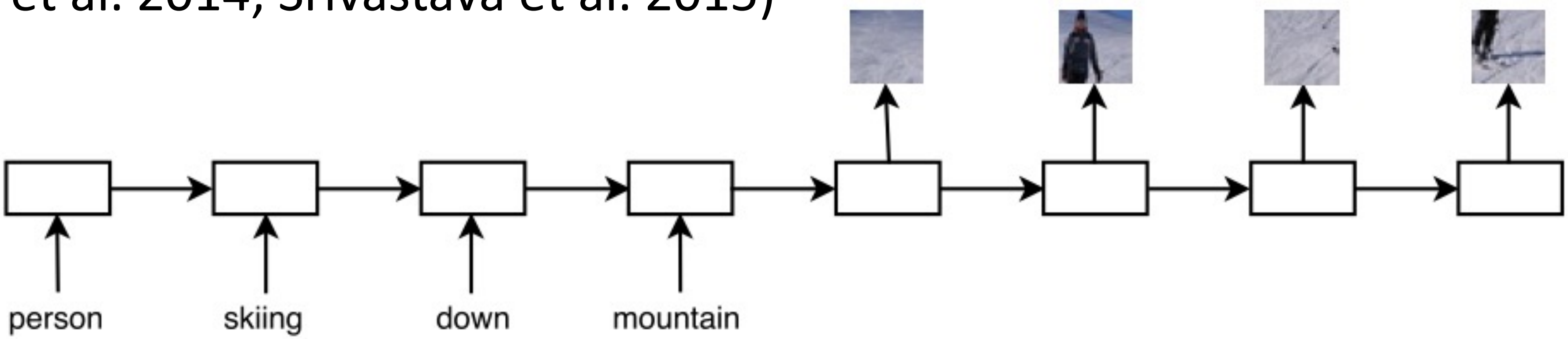


Variational Autoecnoder

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

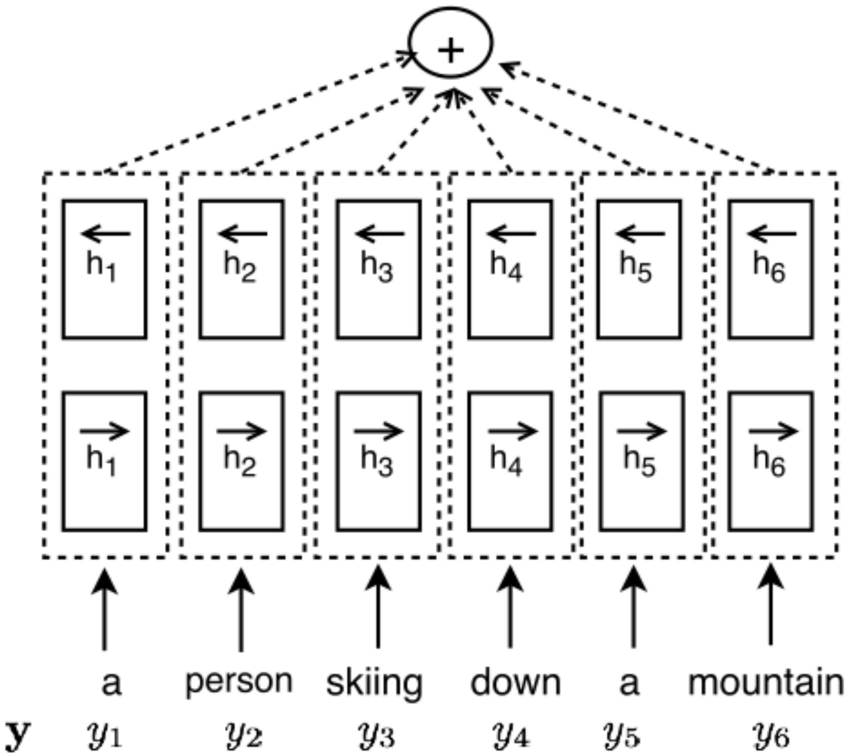
Sequence-to-Sequence

- Sequence-to-sequence framework. (Sutskever et al. 2014; Cho et al. 2014; Srivastava et al. 2015)



- Caption (**y**) is represented as a sequence of consecutive words.
- Image (**x**) is represented as a sequence of patches drawn on canvas.
- Attention mechanism over:
  - Words: Which words to focus on when generating a patch
  - Image Location Where to place the generated patches on the canvas

Representing Captions Bidirectional RNN



- Forward RNN reads the sentence **y** from left to right:

$$[\overrightarrow{\mathbf{h}}_1^{lang}, \overrightarrow{\mathbf{h}}_2^{lang}, , \overrightarrow{\mathbf{h}}_N^{lang}]$$

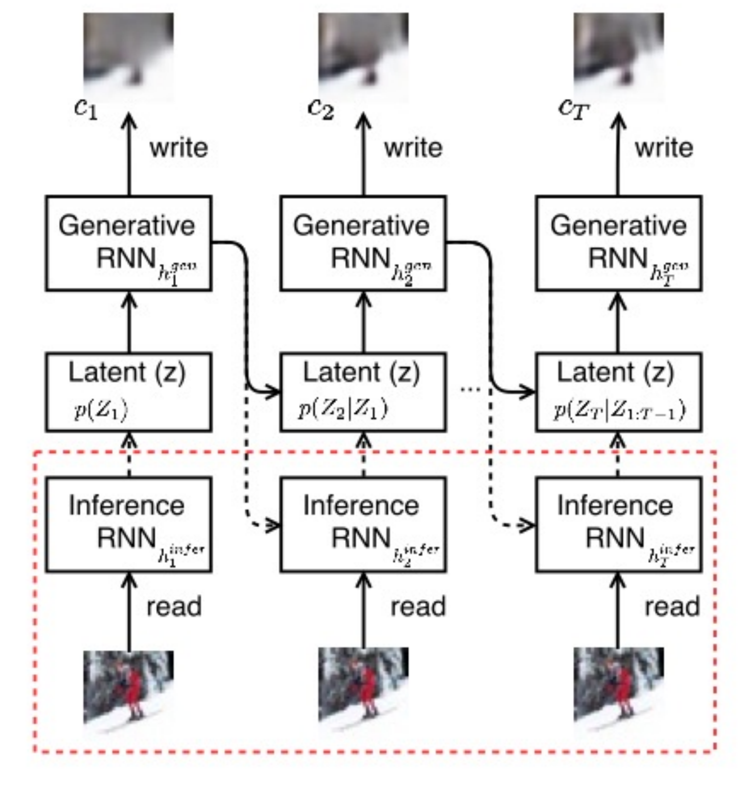
- Backward RNN reads the sentence **y** from right to left:

$$[\overleftarrow{\mathbf{h}}_1^{lang}, \overleftarrow{\mathbf{h}}_2^{lang}, , \overleftarrow{\mathbf{h}}_N^{lang}]$$

- The hidden states are then concatenated:

$$\mathbf{h}^{lang} = [\mathbf{h}_1^{lang}, \mathbf{h}_2^{lang}, \ldots, \mathbf{h}_N^{lang}], \quad \text{with } \mathbf{h}_i^{lang} [\overrightarrow{\mathbf{h}}_i^{lang}, \overleftarrow{\mathbf{h}}_i^{lang}]$$

DRAW Model



write operator:

- At each step the model generates a p x p patch $K(\mathbf{h}_t^{gen}) \in R^{p \times p}$

- It gets transformed into w x h canvas using two arrays of Gaussian filter banks

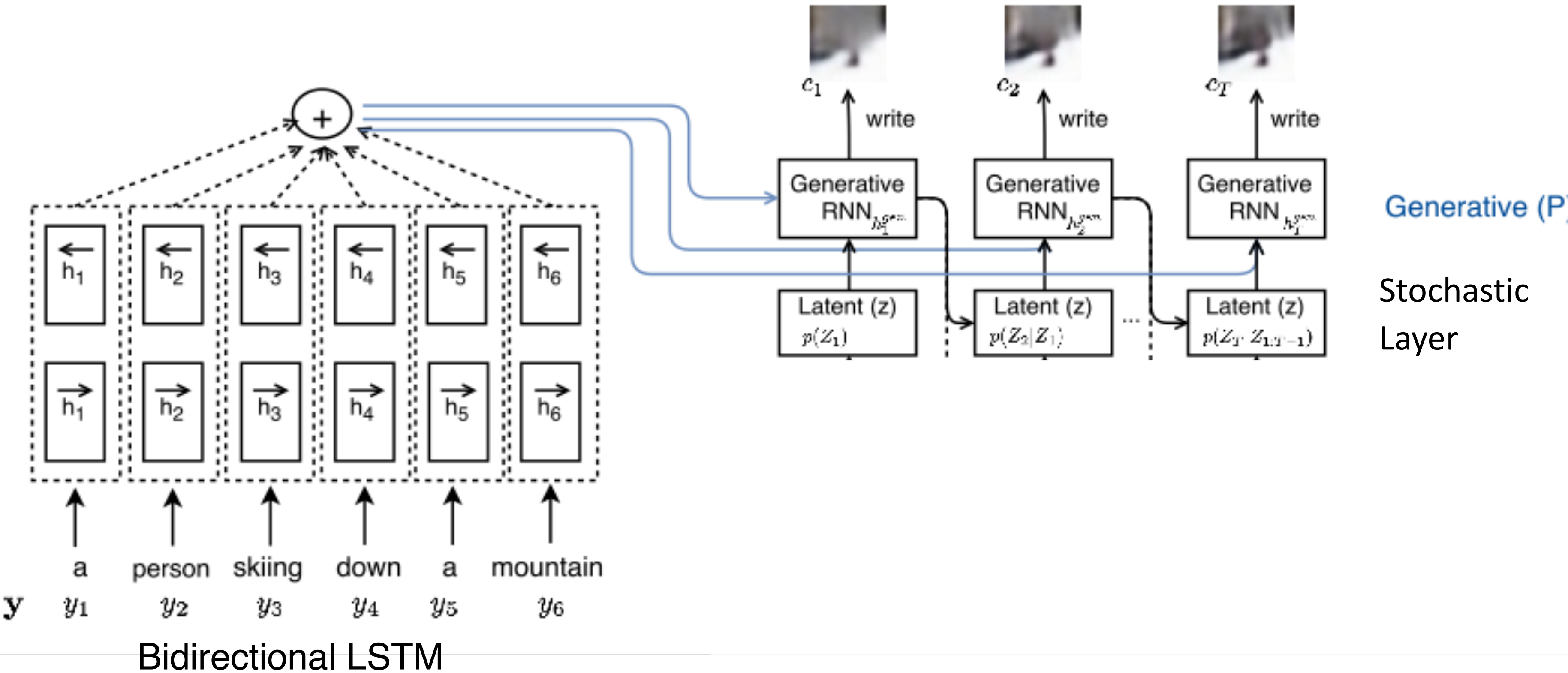
$$F_x(\mathbf{h}_t^{gen}) \in R^{h \times p}$$
$$F_y(\mathbf{h}_t^{gen}) \in R^{w \times p}$$

whose filter locations and scales are computed from $\mathbf{h}_t^{gen}$:

$$write(\mathbf{h}_t^{gen}) = F_x(\mathbf{h}_t^{gen}) \times K(\mathbf{h}_t^{gen}) \times F_y(\mathbf{h}_t^{gen})$$

(Gregor et. al. 2015)

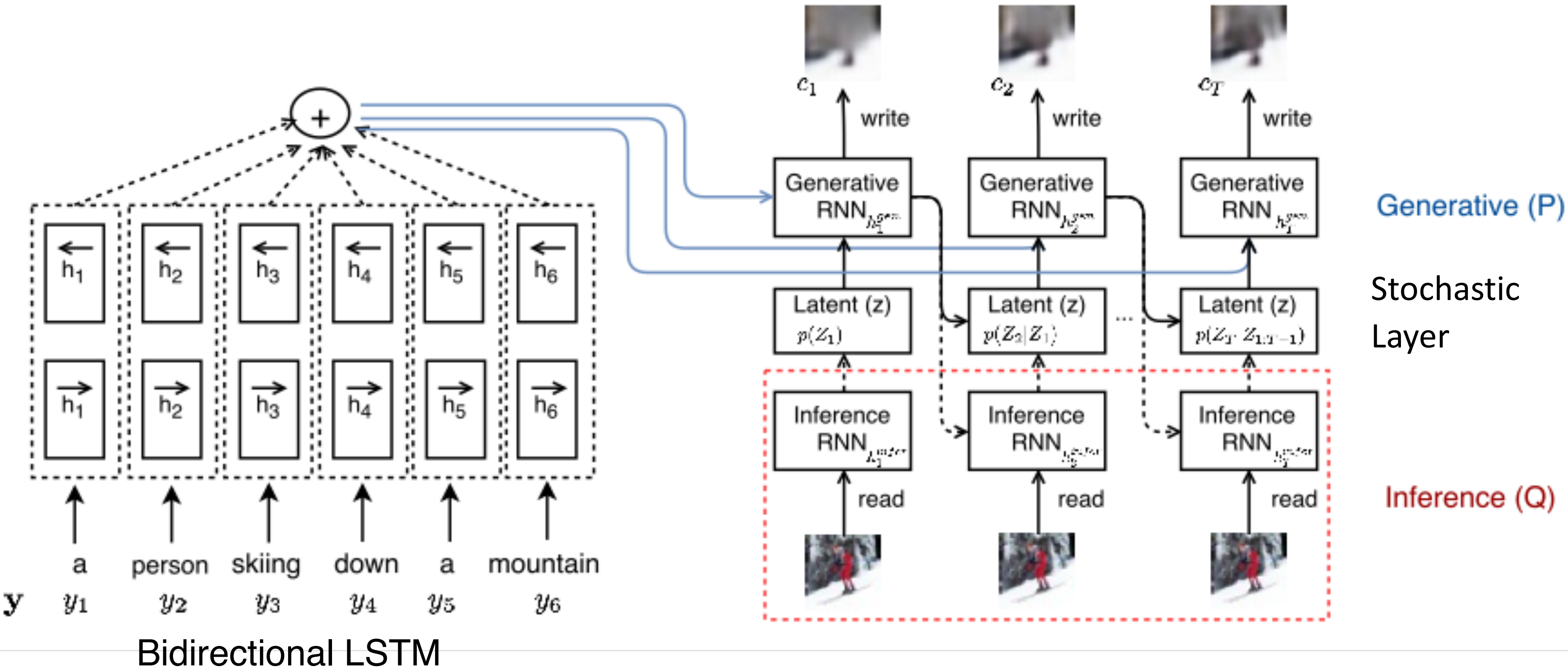
Overall Model



- Generative Model: Stochastic Recurrent Network, chained sequence of Variational Autoencoders, with a single stochastic layer.

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

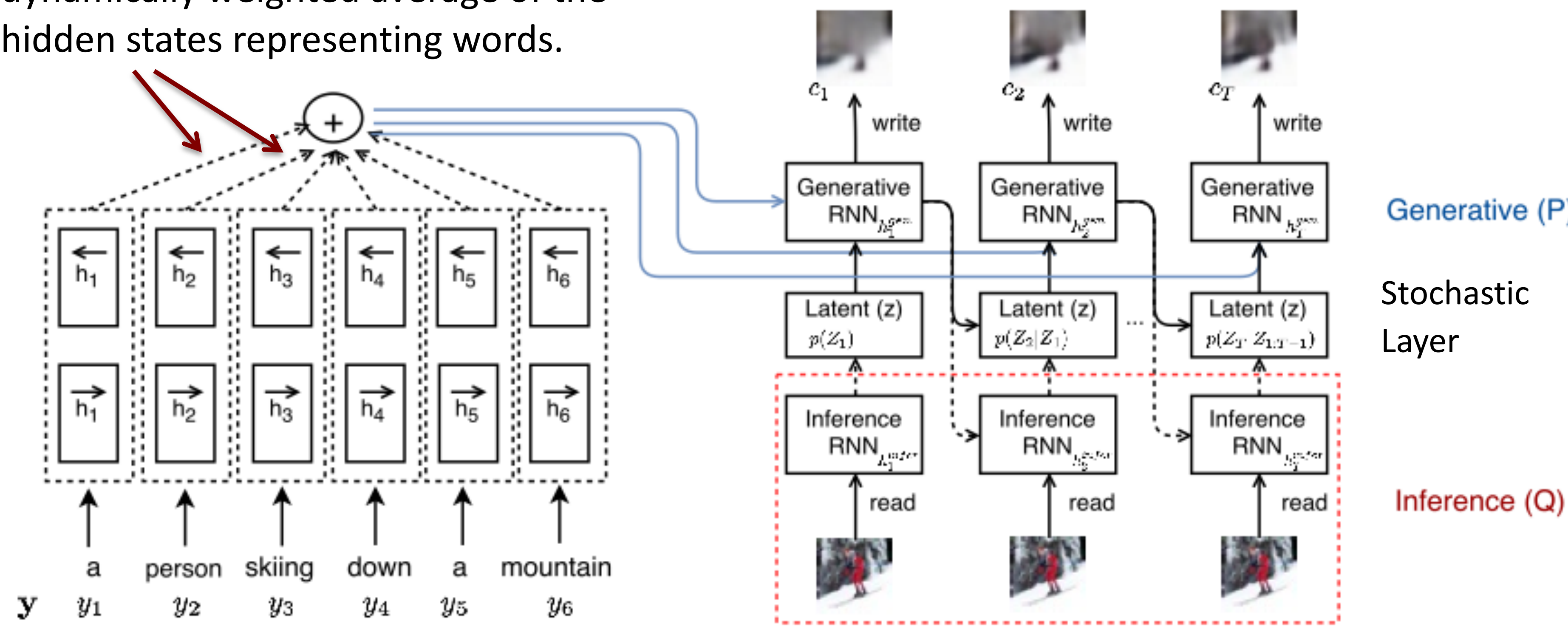
Overall Model



- Generative Model: Stochastic Recurrent Network, chained sequence of Variational Autoencoders, with a single stochastic layer.
- Recognition Model: Deterministic Recurrent Network.

Overall Model

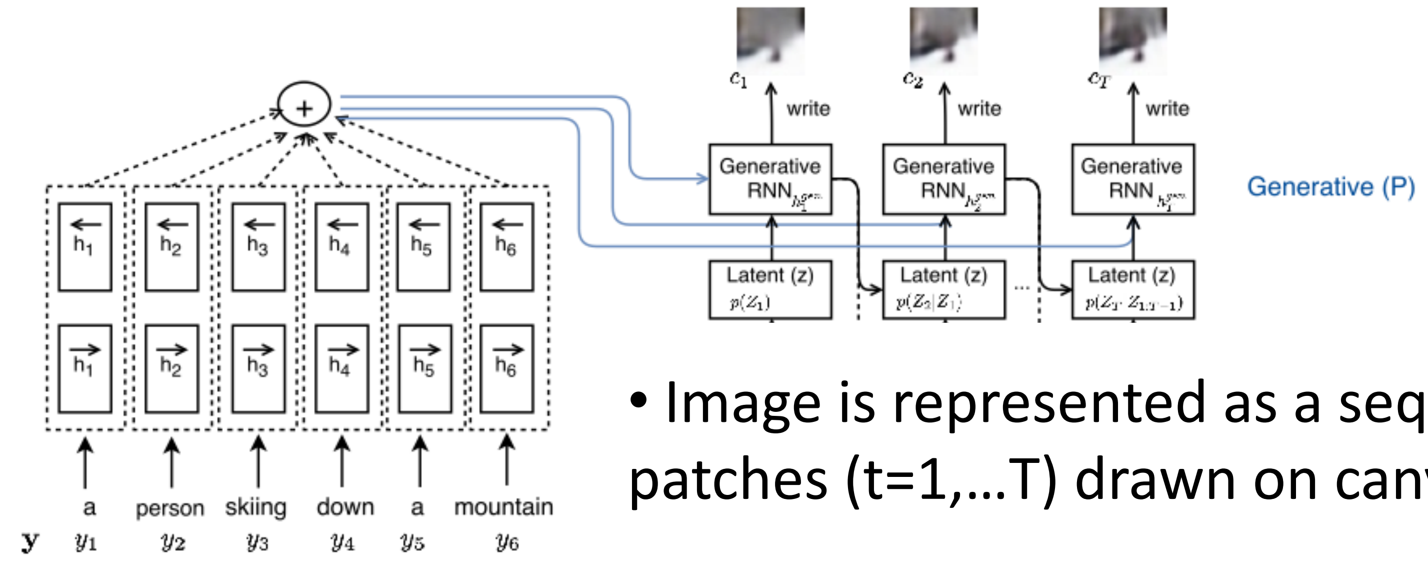
Sentence representation: dynamically weighted average of the hidden states representing words.



- Attention (alignment): Focus on different words at different time steps when generating patches and placing them on the canvas.

Bahdanau et. al. 2015

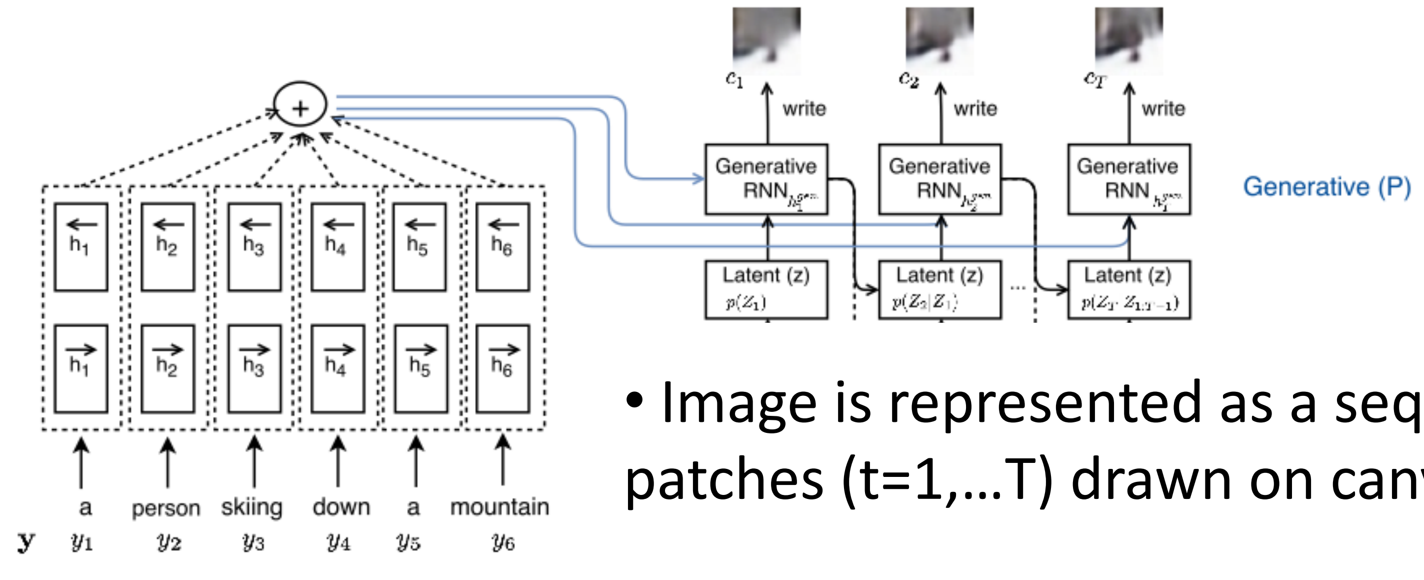
Generating Images



- Image is represented as a sequence of patches (t=1,...T) drawn on canvas:

$$\mathbf{z}_t \sim P(\mathbf{Z}_t|\mathbf{Z}_{1:t-1}) = \mathcal{N}\left(\mu(\mathbf{h}^{gen}_{t-1}), \sigma(\mathbf{h}^{gen}_{t-1})\right), \quad P(\mathbf{Z}_1) = \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$$

Generating Images

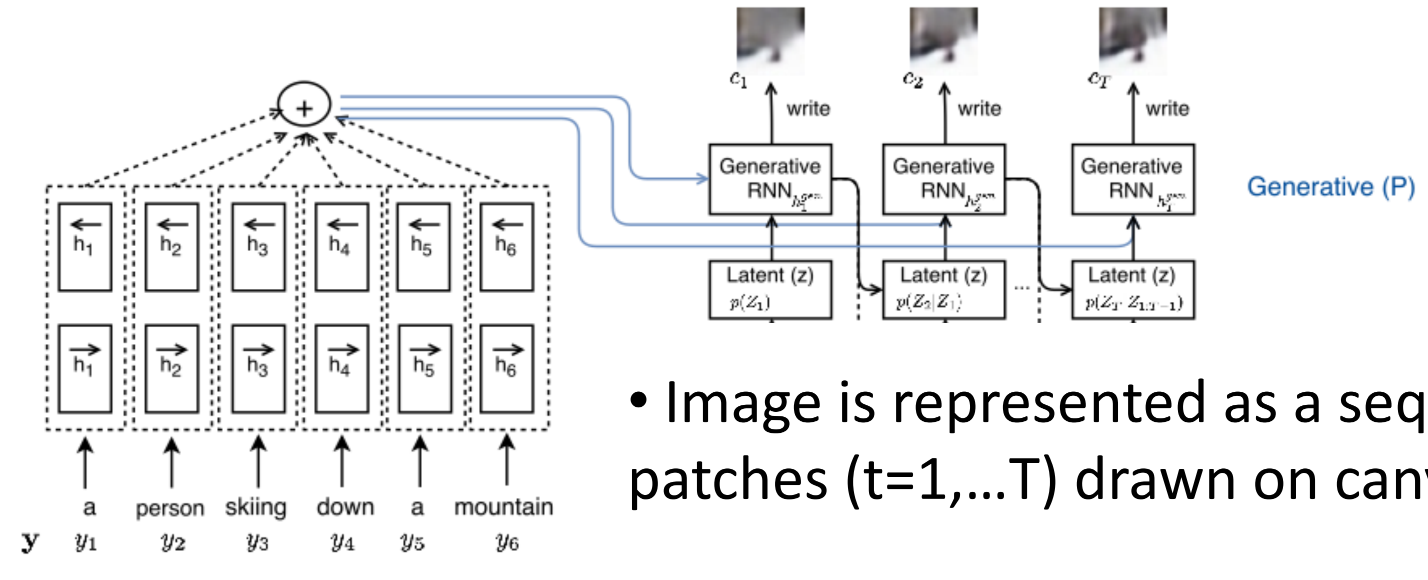


- Image is represented as a sequence of patches (t=1,…T) drawn on canvas:

$$\mathbf{z}_t \sim P(\mathbf{Z}_t|\mathbf{Z}_{1:t-1}) = \mathcal{N}\left(\mu(\mathbf{h}_{t-1}^{gen}), \sigma(\mathbf{h}_{t-1}^{gen})\right), \quad P(\mathbf{Z}_1) = \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$$

$$s_t = align\left(\mathbf{h}_{t-1}^{gen}, \mathbf{h}^{lang}\right) \quad \mathbf{h}_t^{gen} = LSTM^{gen}\left(\mathbf{h}_{t-1}^{gen}, [\mathbf{z}_t, s_t]\right)$$

Generating Images



- Image is represented as a sequence of patches (t=1,…T) drawn on canvas:

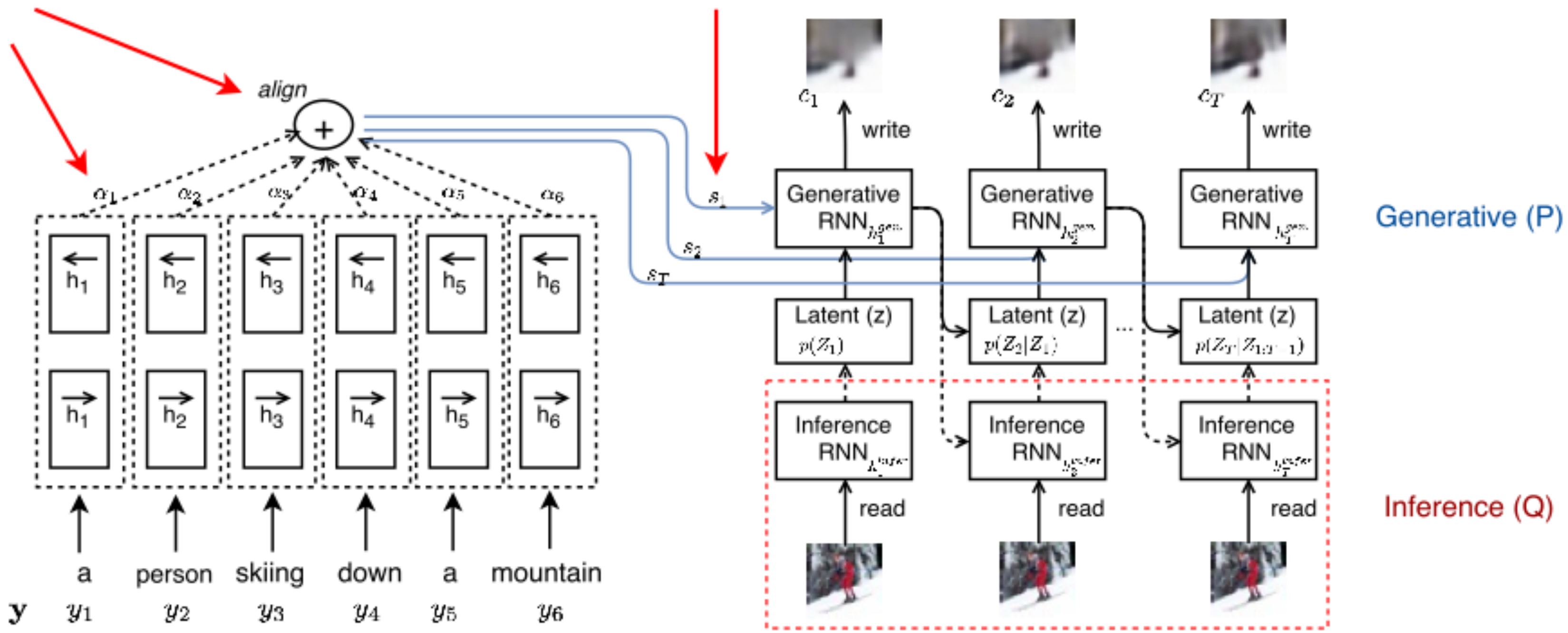
$$\mathbf{z}_t \sim P(\mathbf{Z}_t|\mathbf{Z}_{1:t-1}) = \mathcal{N}\left(\mu(\mathbf{h}^{gen}_{t-1}), \sigma(\mathbf{h}^{gen}_{t-1})\right), \quad P(\mathbf{Z}_1) = \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$$

$$s_t = align\left(\mathbf{h}^{gen}_{t-1}, \mathbf{h}^{lang}\right) \quad \mathbf{h}^{gen}_t = LSTM^{gen}\left(\mathbf{h}^{gen}_{t-1}, [\mathbf{z}_t, s_t]\right)$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} + write(\mathbf{h}^{gen}_t) \quad \mathbf{x} \sim P(\mathbf{x}|\mathbf{y}, \mathbf{Z}_{1:T}) = \prod_i \text{Bern}(\boldsymbol{\sigma}(c_{T,i}))$$

- In practice, we use the conditional mean: $\mathbf{x} = \boldsymbol{\sigma}(\mathbf{c}_T)$.

Alignment Model



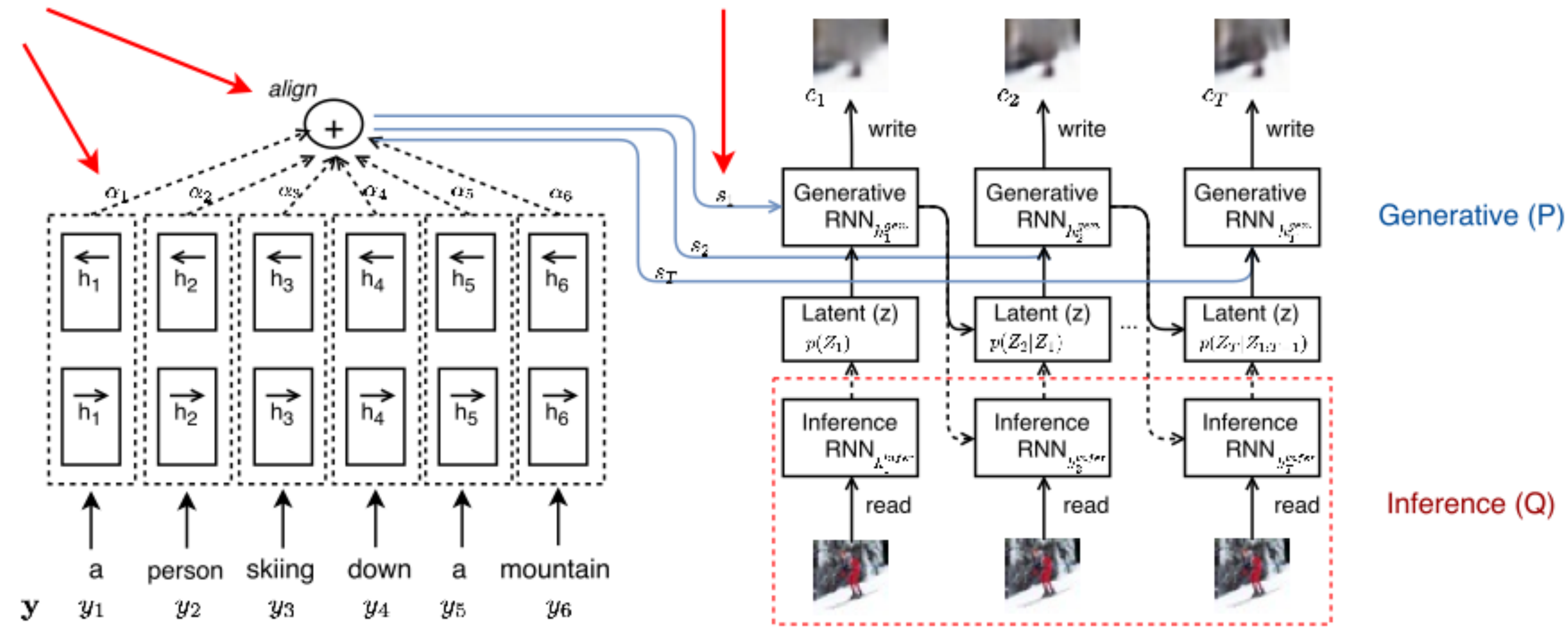
- Dynamic sentence representation at time t: weighted average of the bi-directional hidden states:

$$s_t = align(\mathbf{h}^{gen}_{t-1}, \mathbf{h}^{lang}) = \alpha^t_1 \mathbf{h}^{lang}_1 + \alpha^t_2 \mathbf{h}^{lang}_2 + \cdots + \alpha^t_N \mathbf{h}^{lang}_N$$

where the alignment probabilities are computed as:

$$e^t_k = \mathbf{v}^\top \tanh\left(U\mathbf{h}^{lang}_k + W\mathbf{h}^{gen}_{t-1} + b\right), \;\; \alpha^t_k = \frac{\exp\left(e^t_k\right)}{\sum_{i=1}^N \exp\left(e^t_i\right)}$$

Learning

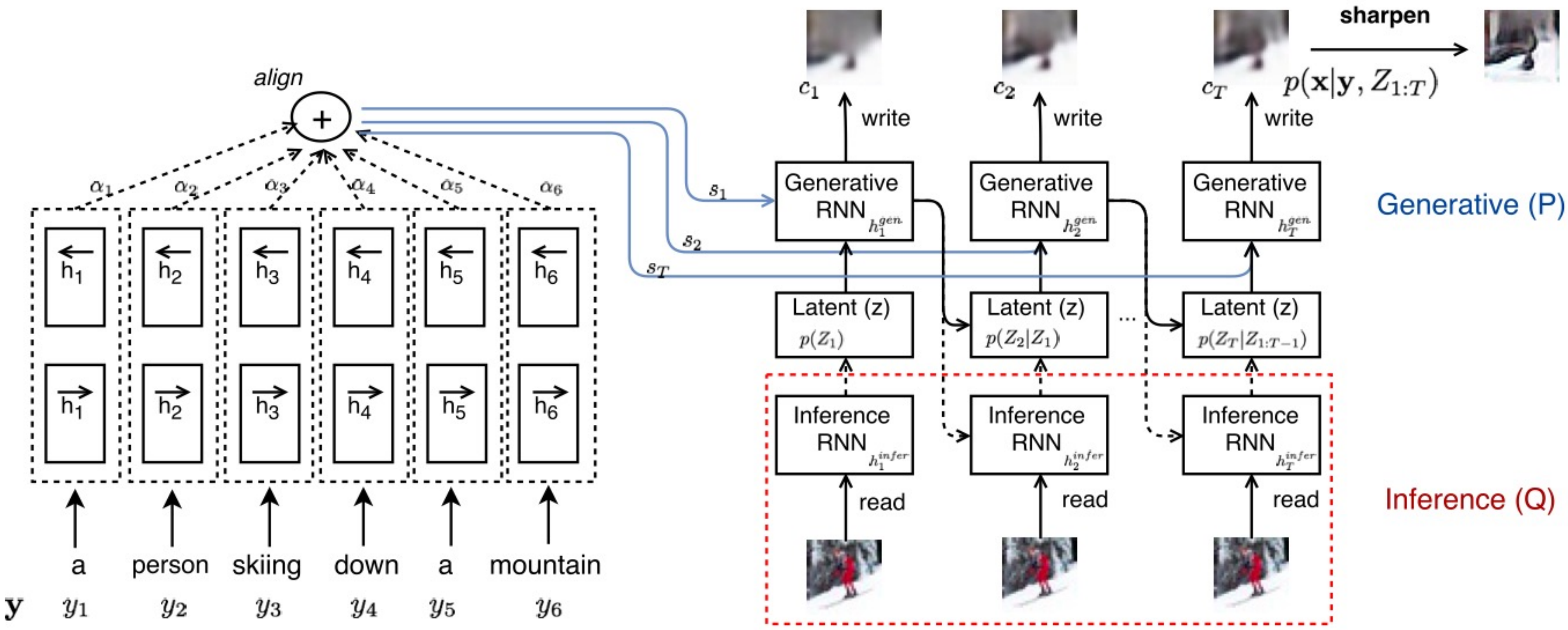


- Maximize the variational lower bound on the marginal log-likelihood of the correct image **x** given the caption **y**:

$$\mathcal{L} = \sum_{Z} Q(Z|\mathbf{x},\mathbf{y}) \log P(\mathbf{x}|Z,\mathbf{y}) - \mathrm{D}_{KL}\big(Q(Z|\mathbf{x},\mathbf{y})||P(Z|\mathbf{y})\big)$$

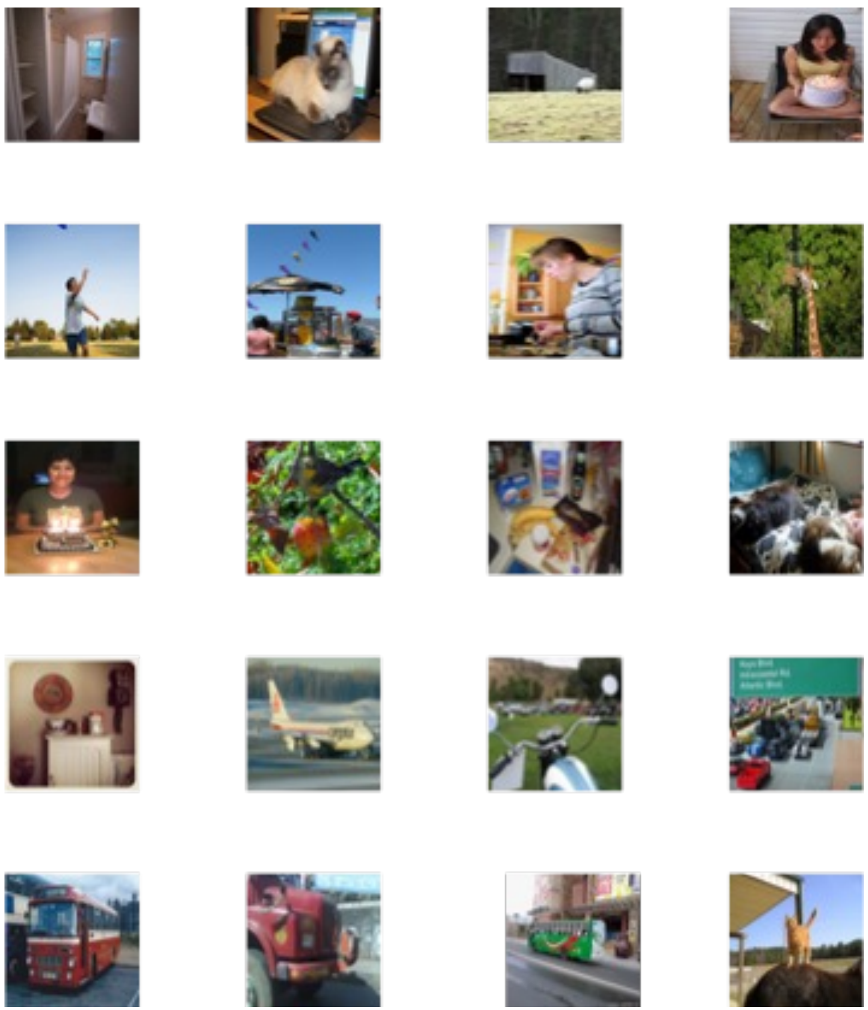
$$\leq \log P(\mathbf{x}|\mathbf{y})$$

Sharpening



- Additional post processing step: use an adversarial network trained on residuals of a Laplacian pyramid to sharpen the generated images (Denton et. al. 2015).

MS COCO Dataset



- Contains 83K images.
- Each image contains 5 captions.
- Standard benchmark dataset for many of the recent image captioning systems.

Lin et. al. 2014

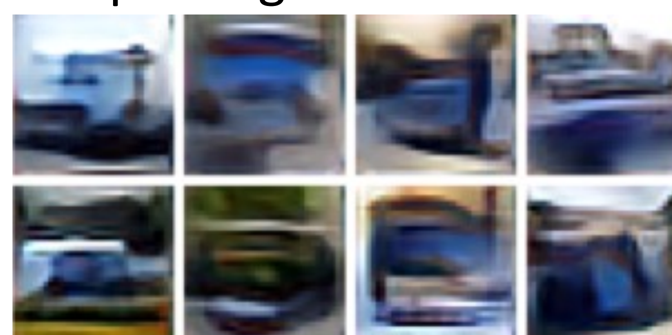
Flipping Colors

A **yellow school bus** parked in the parking lot

A **red school bus** parked in the parking lot

A **green school bus** parked in the parking lot

A **blue school bus** parked in the parking lot



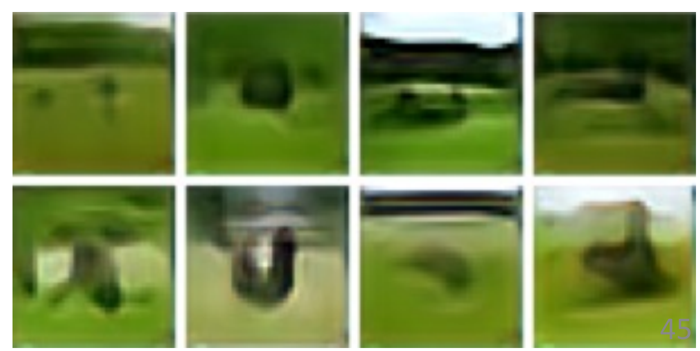
Flipping Backgrounds

A very large commercial plane flying **in clear skies**.

A very large commercial plane flying **in rainy skies**.

A herd of elephants walking across a **dry grass field**.

A herd of elephants walking across a **green grass field**.



Flipping Objects

The decadent chocolate desert is on the table.

A bowl of bananas is on the table..

A vintage photo of **a cat**.

A vintage photo of **a dog**.

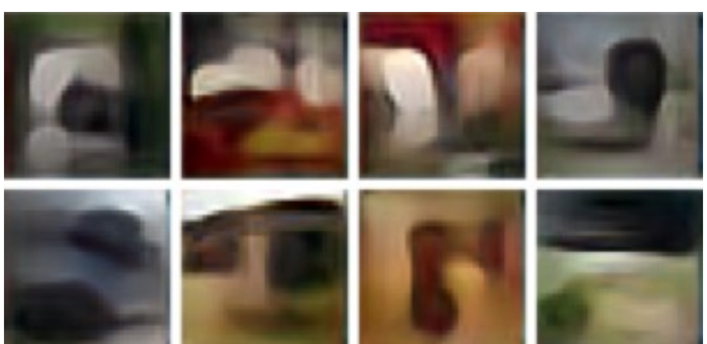
Qualitative Comparison

A group of people walk on a beach with surf boards

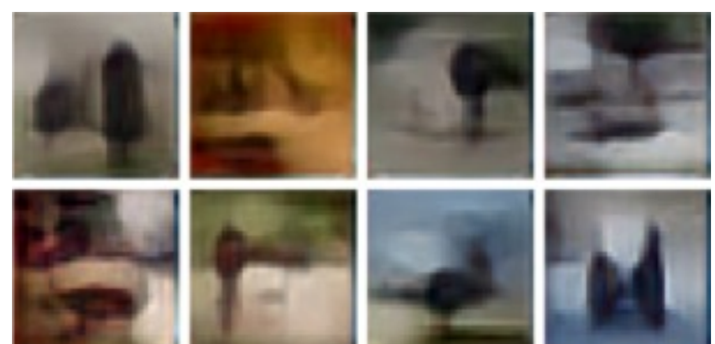
Our Model

LAPGAN (Denton et. al. 2015)

Conv-Deconv VAE



Fully Connected VAE



Variational Lower-Bound

- We can estimate the variational lower-bound on the average test log-probabilities:

Model	Training	Test
Our Model	-1792,15	-1791,53
Skipthought-Draw	-1794,29	-1791,37
noAlignDraw	-1792,14	-1791,15

- At least we can see that we do not overfit to the training data, unlike many other approaches.