Generating Images with Neural Networks

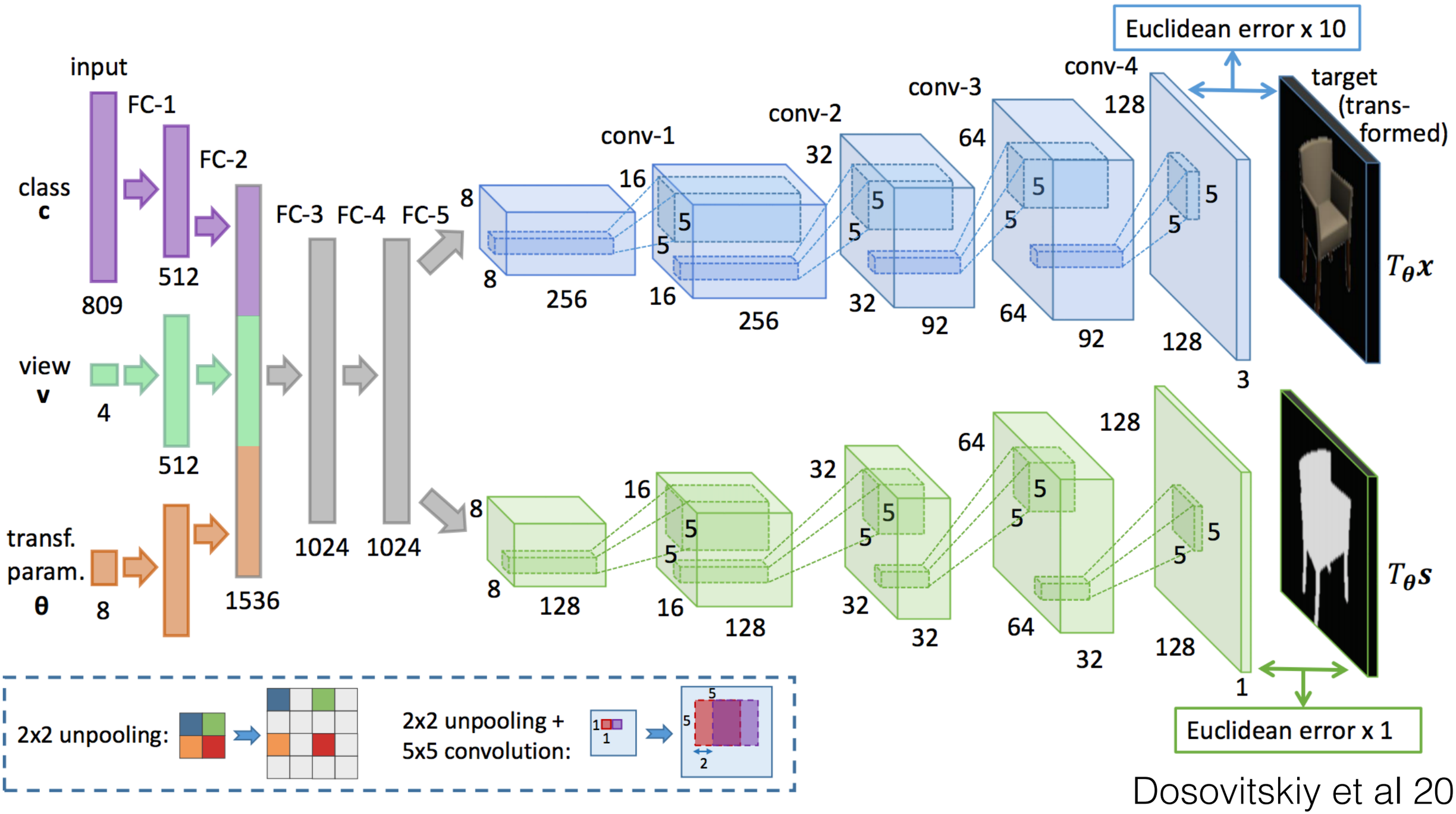
Elman Mansimov

February 9, 2016

Introduction

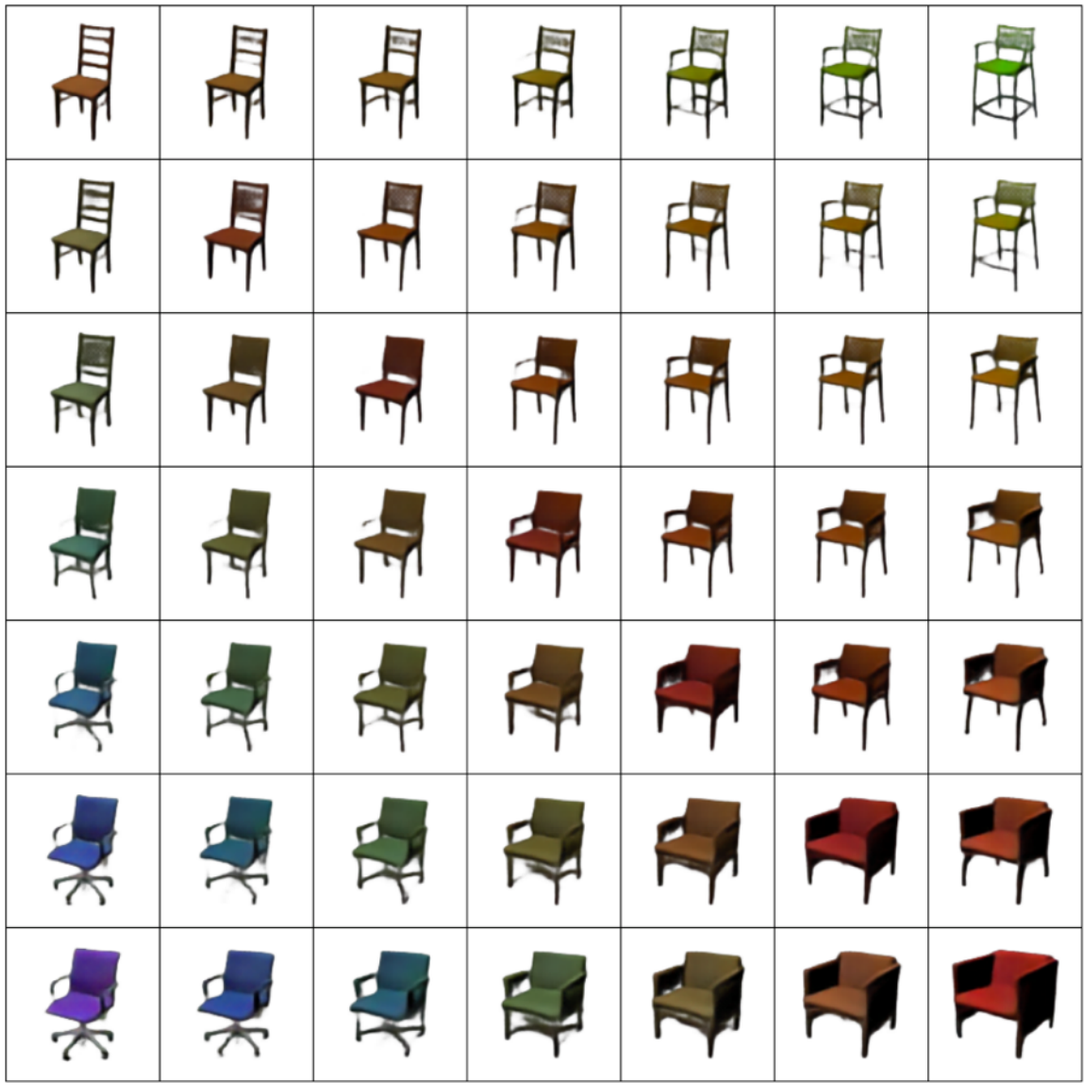
- So far you have seen that Convolutional Neural Networks work well for image classification, localization and etc…
- But does this hold for image generation ?
- Turns out yes !

Deconvolutional Architecture



Dosovitskiy et al 2014

Works really well



Dosovitskiy et al 2014

However

- In this case, the latent variables describing chairs, such as class id, view angle, transformation parameters were available.

- Which makes it strongly supervised and unrealistic in practice.

- Need to resort to generative models that learn to capture latent variables.

Problem Setup

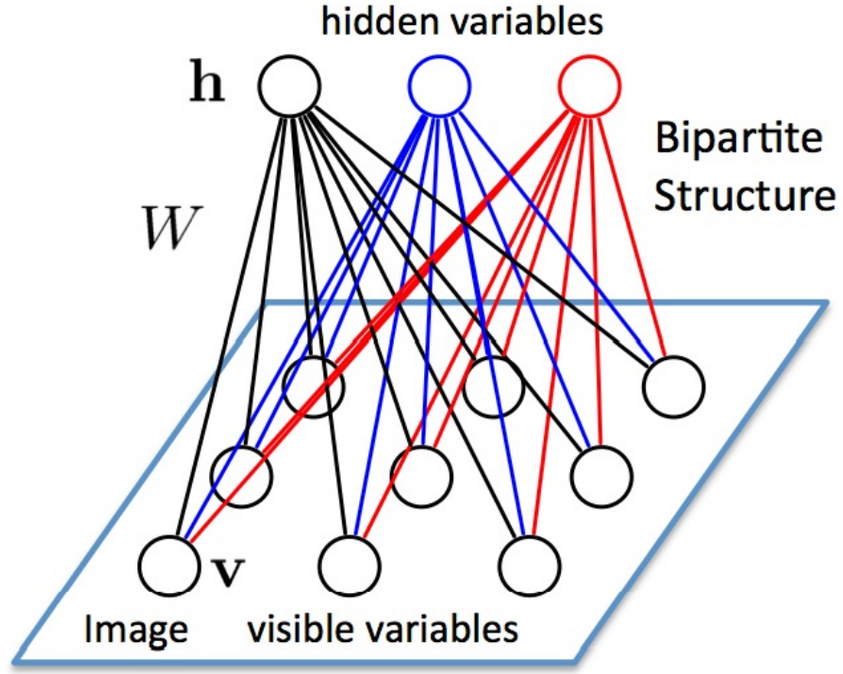
- You have a dataset of images $x = \{x_1, x_2, ..., x_n\}$
- There are latent variables $z = \{z_1, z_2, ..., z_k\}$ that model the data.
- A generative model is a parametrized joint distribution over variables $p(x, z|\theta)$
- Image generation is $p(x|z)$ where z is sampled from some distribution.

Generative Models

- Three main approaches:
- 1. Boltzmann Machines (undirected graphical models)
- 2. Variational Autoencoders (directed graphical models)
- 3. Generative Adversarial Networks (noise contrastive estimation)

Restricted Boltzmann Machines

- Undirected bipartite graphical model.
- $\mathbf{v} \in \{0,1\}^D$ observed visible units
- $\mathbf{h} \in \{0,1\}^K$ hidden binary units



$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z} exp(-E(\mathbf{v},\mathbf{h}))$$

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{v}^\top W \mathbf{h} - b^\top \mathbf{h} - a^\top \mathbf{v}$$

$$Z = \sum_{\mathbf{v}\in\{0,1\}^D} \sum_{\mathbf{h}\in\{0,1\}^K} exp(-E(\mathbf{v},\mathbf{h}))$$

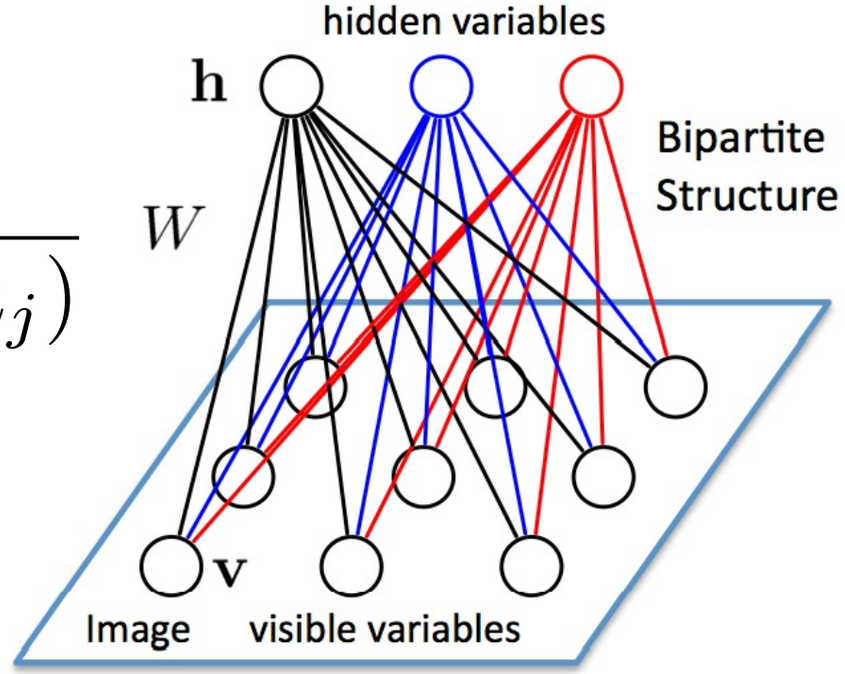
Hinton & Sejnowski,1986; Smolensky,1986

Restricted Boltzmann Machines

- Inferring the distribution of hidden variables is easy

$$P(\mathbf{h} \mid \mathbf{v}) = \prod_j P(h_j \mid \mathbf{v})$$

$$P(h_j = 1 \mid \mathbf{v}) = \frac{1}{1 + exp(-\mathbf{v}^\top W - a_j)}$$



- Similarly for visible variables

$$P(\mathbf{v} \mid \mathbf{h}) = \prod_i P(v_i \mid \mathbf{h})$$

$$P(v_i = 1 \mid \mathbf{h}) = \frac{1}{1 + exp(-W\mathbf{h} - b_i)}$$

Learning RBMs

- We want to learn parameters $\theta = \{W, a, b\}$

- Minimize negative log-likelihood objective

$$L(\theta) = -log(p(\mathbf{v})) = -log(\frac{1}{Z}\sum_{\mathbf{h}} exp(-E(\mathbf{v}, \mathbf{h})))$$

- Derivative of log-likelihood objective function

$$\frac{\partial L(\theta)}{\partial W_{ij}} = \mathbb{E}_{P_{data}}[v_i h_j] - \mathbb{E}_{P_\theta}[v_i h_j]$$

Hinton, 2000

Derivative of log-likelihood

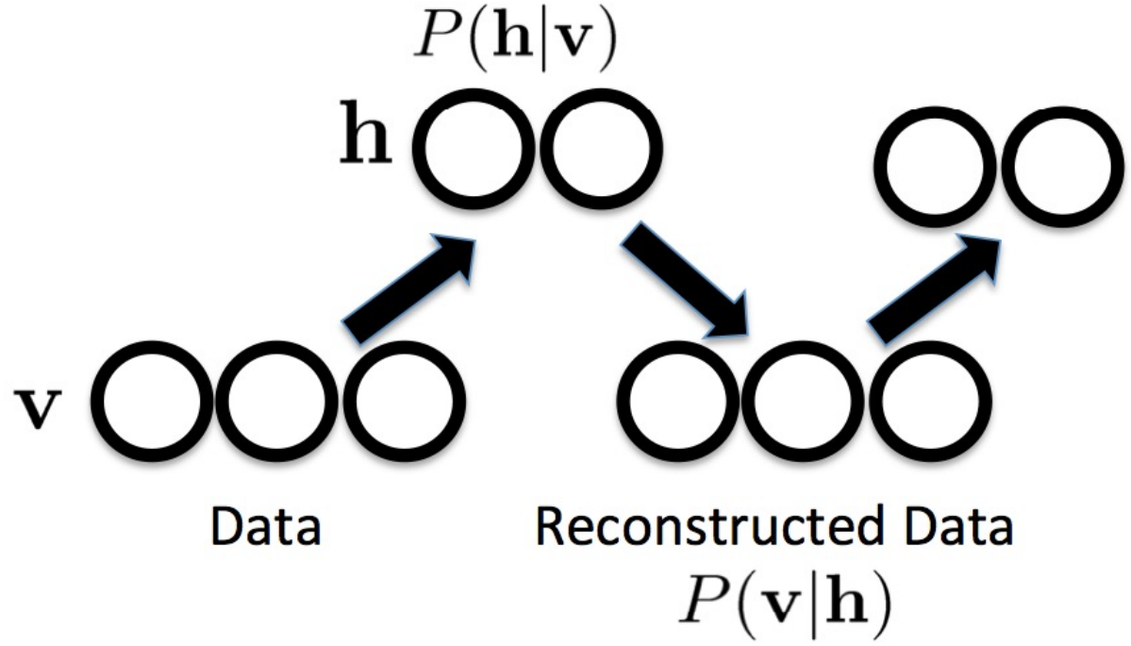
$$\frac{\partial L(\theta)}{\partial W_{ij}} = \mathbb{E}_{P_{data}}[v_i h_j] - \mathbb{E}_{P_\theta}[v_i h_j]$$

Easy to compute

intractable:
requires summing
over all possible configurations

$$\sum_{\mathbf{v}} \sum_{\mathbf{h}} v_i h_j P_\theta(\mathbf{vh})$$

Contrastive Divergence

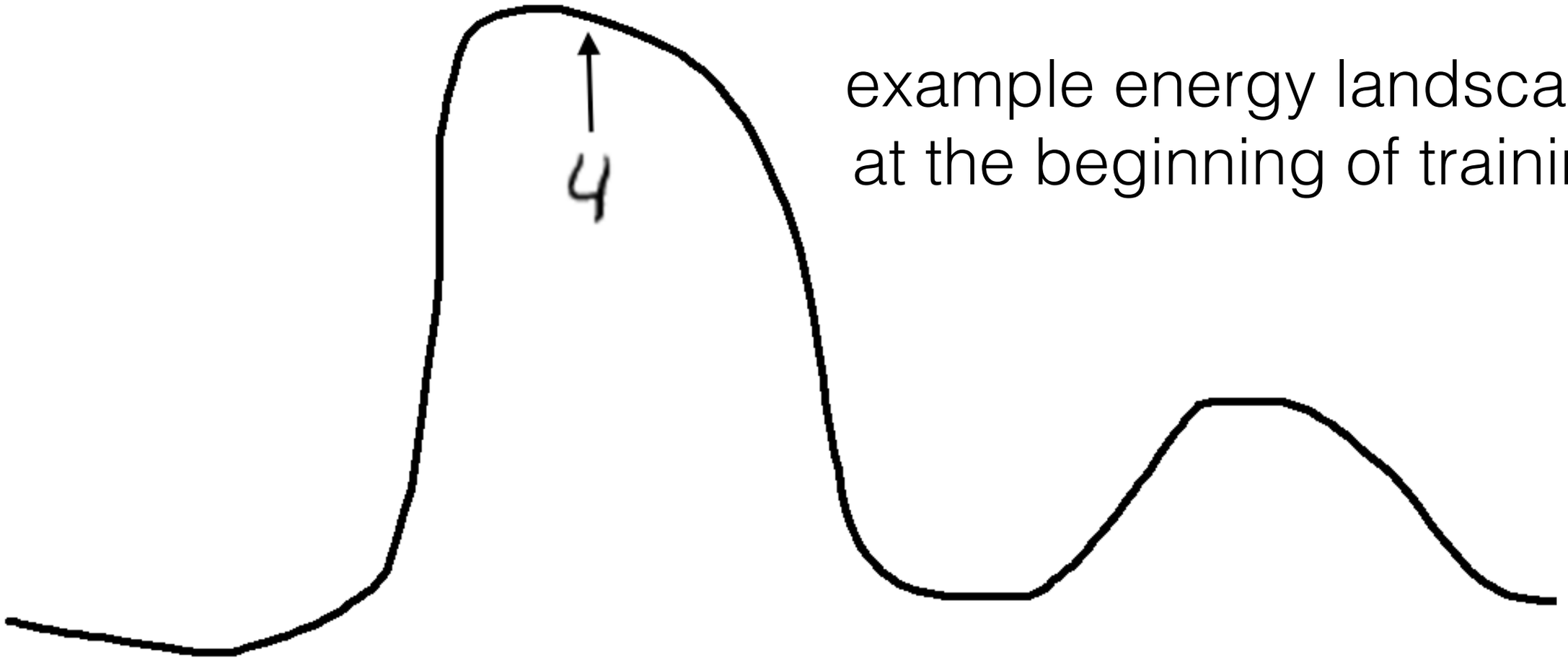


Idea: Replace the expectation by point estimates.

Initialize visible units to the training data.

For fixed number of steps run MCMC chain (Gibbs Sampler) to get dream samples of visible and hidden units.

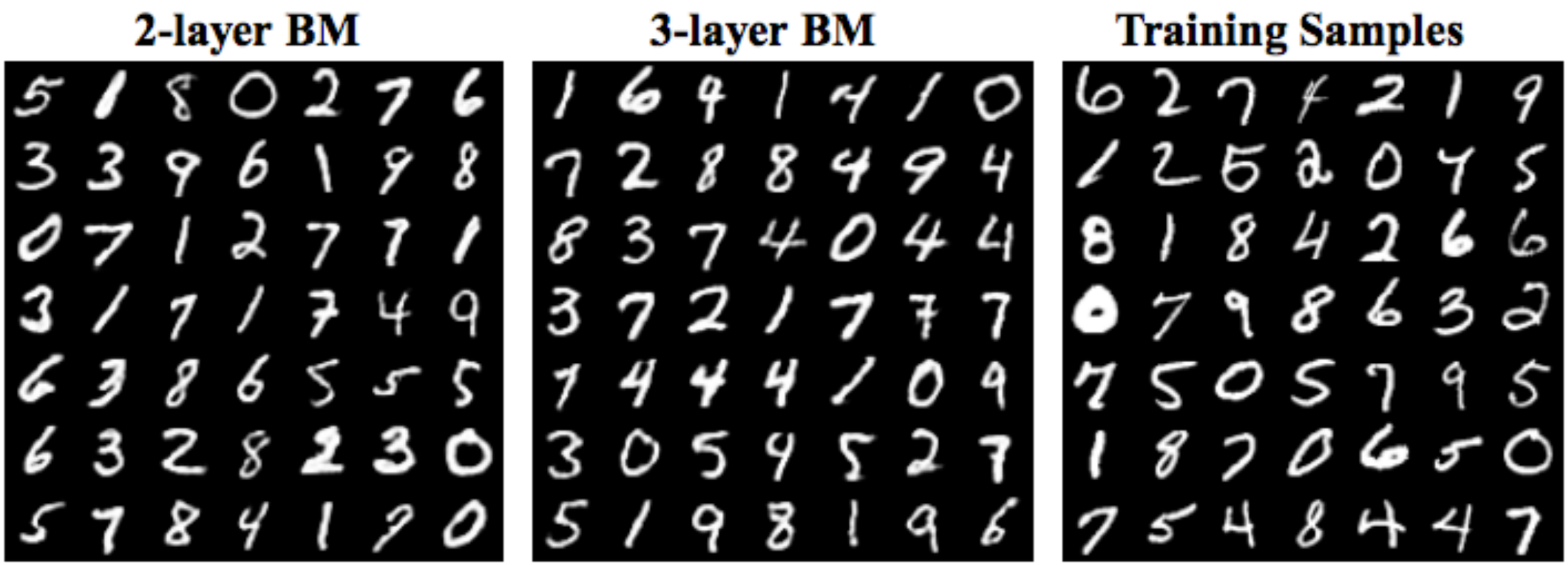
Contrastive Divergence



example energy landscape
at the beginning of training

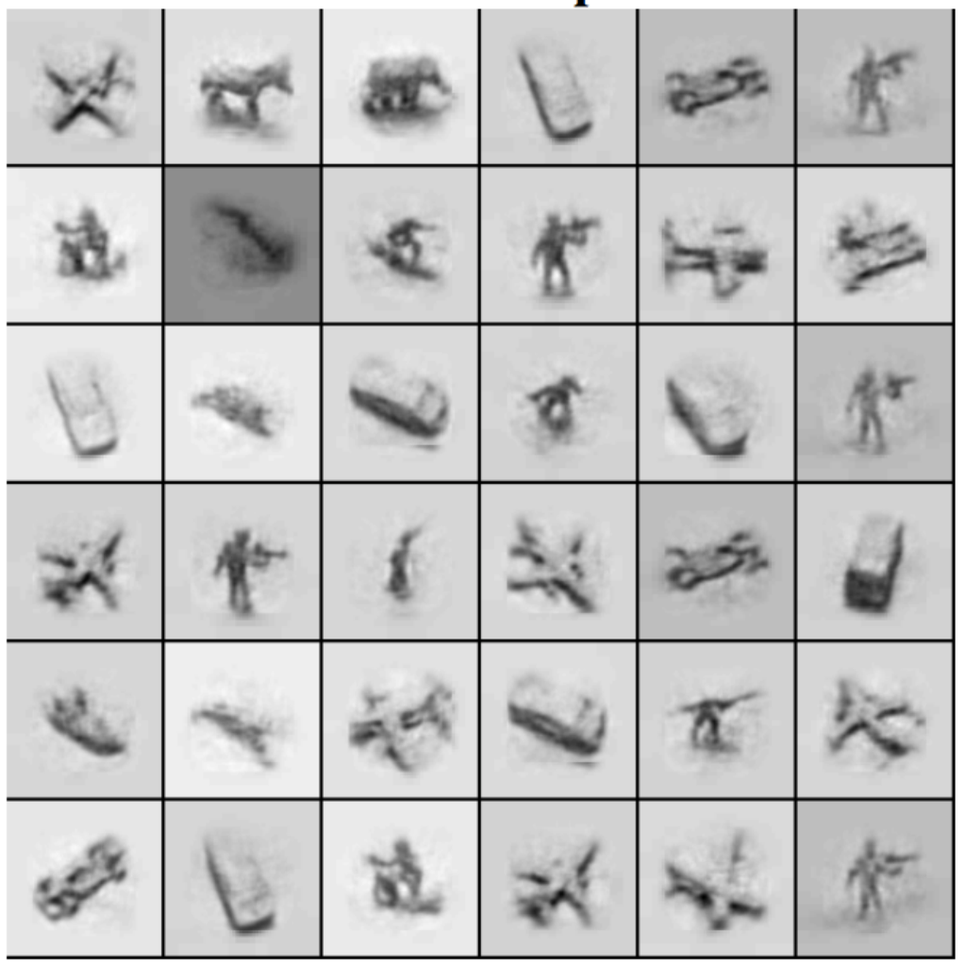
The goal is to minimize energy (or maximize likelihood) for training samples and maximize energy for randomly dreamed samples.

Generated MNIST Digits



Salakhutdinov & Hinton, 2009

Generating 3D objects



Salakhutdinov & Hinton, 2009

Issues with Boltzmann Machines

- Inference in Boltzmann Machines is computationally expensive.

- MCMC sampler creates an inner loop of training (for each example).

- Usually one sample is not enough to get good likelihood (except for the beginning of training).

- As the model gets sharper, gibbs sampler requires more steps to yield a better estimate of gradient. (learn more by studying MCMC methods).

Variational Inference

- But is it possible to avoid normalization constant ?

- What if instead of you learned some distribution $q_\phi(z|x)$ that approximates posterior $p(z|x)$

- ϕ is free to vary.

- This distribution is parametrized by neural network.

Variational Lower Bound

$$log(p(x)) >= \mathcal{L}(x) + \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x) \,\|\, p(z|x))$$

since KL divergence is non-negative maximizing first term (called variational lower bound) with minimize KL divergence

$$\mathcal{L}(x) = -\mathrm{D}_{\mathrm{KL}}(q_\phi(z|x) \,\|\, p(z)) + \mathbb{E}_{q_\phi(z|x)}\Big[log(p(x|z))\Big]$$

regularization term: distance between learned distribution q and fixed prior p

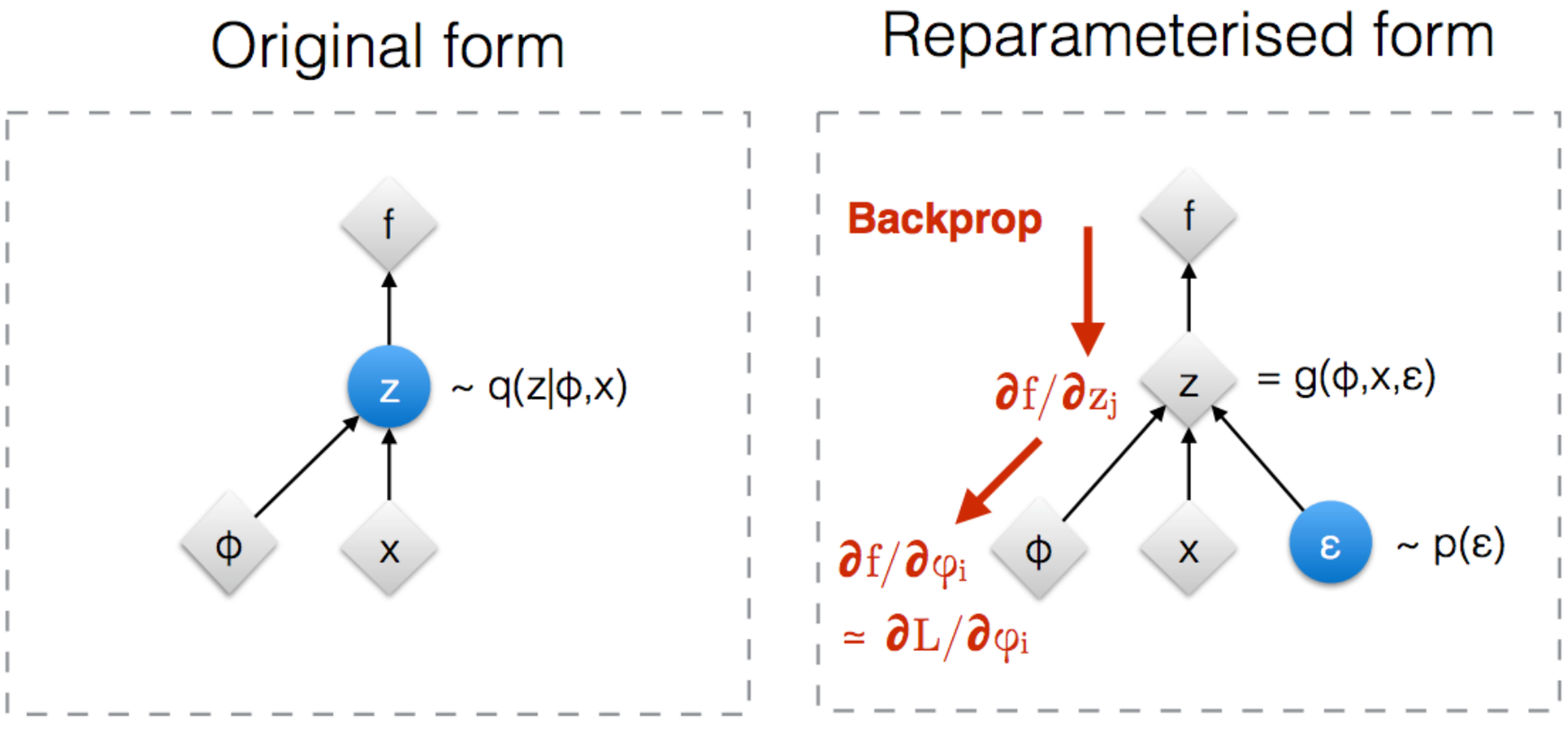
$$q_\phi(z|x) = N(\mu, \sigma^2) \quad p(z) = N(0, 1)$$

reconstruction term

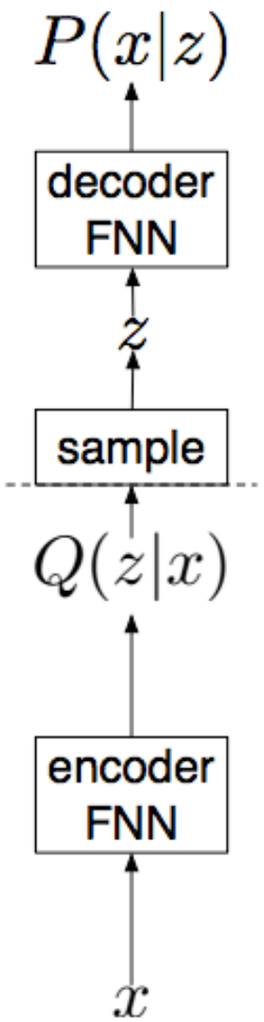
Kingma & Welling, 2013/14 & Rezende & Mohamed, 2014

Reparametrization trick

- Still can't backpropagate though $\mathbb{E}_{q_\phi(z|x)}\Big[log(p(x|z))\Big]$
- Instead of sampling from $q_\phi(z|x) = N(\mu, \sigma^2)$ sample from $\mu + \sigma^2 * N(0,1)$



Variational Autoencoder (VAE)



$$\mathcal{L}(x) = -\mathrm{D}_{\mathrm{KL}}(q_\phi(z|x) \,\|\, p(z)) + \mathbb{E}_{\mu+\sigma^2 * N(0,1)}\Big[log(p(x|z))\Big]$$

$$q_\phi(z|x) = N(\mu, \sigma^2)$$

Trained by backpropagation
as simple as auto-encoders

Convolutional/Deconvolutional VAE trained on faces

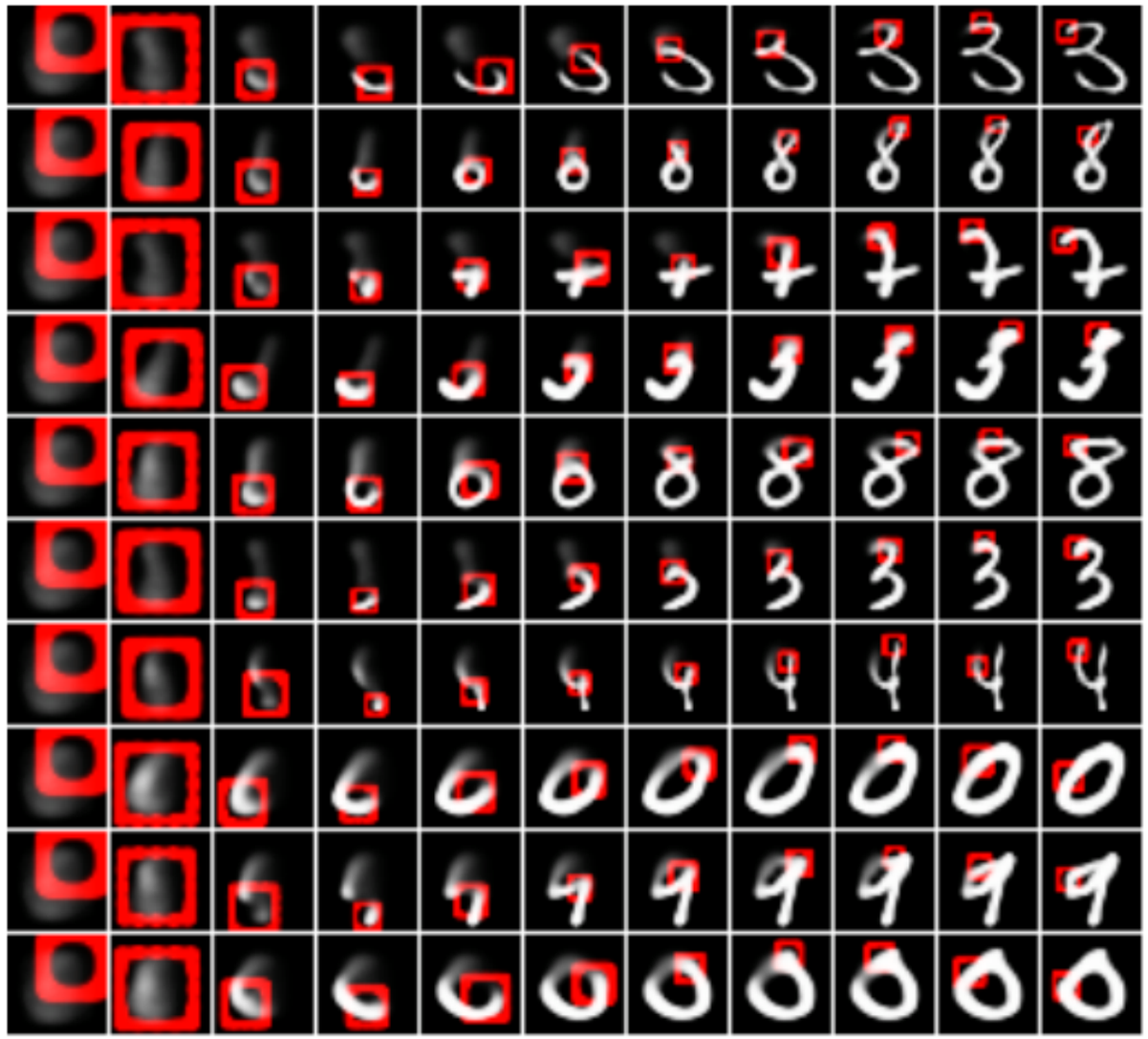
trained by
A. Radford
2014

DRAW: Deep Recurrent Attentive Writer

- What if we augment encoder and decoder with recurrent neural networks

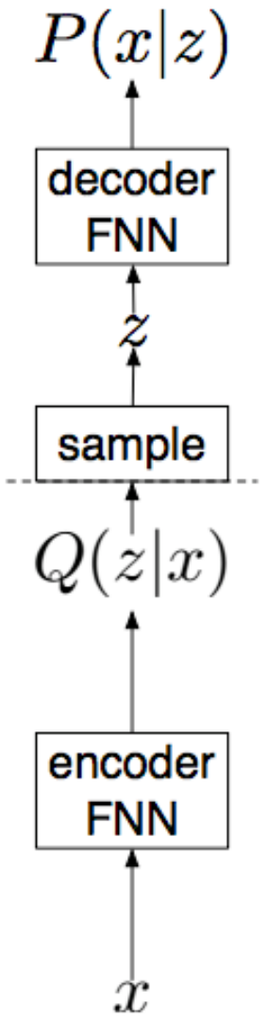
- Inference and generation defined by sequential process.

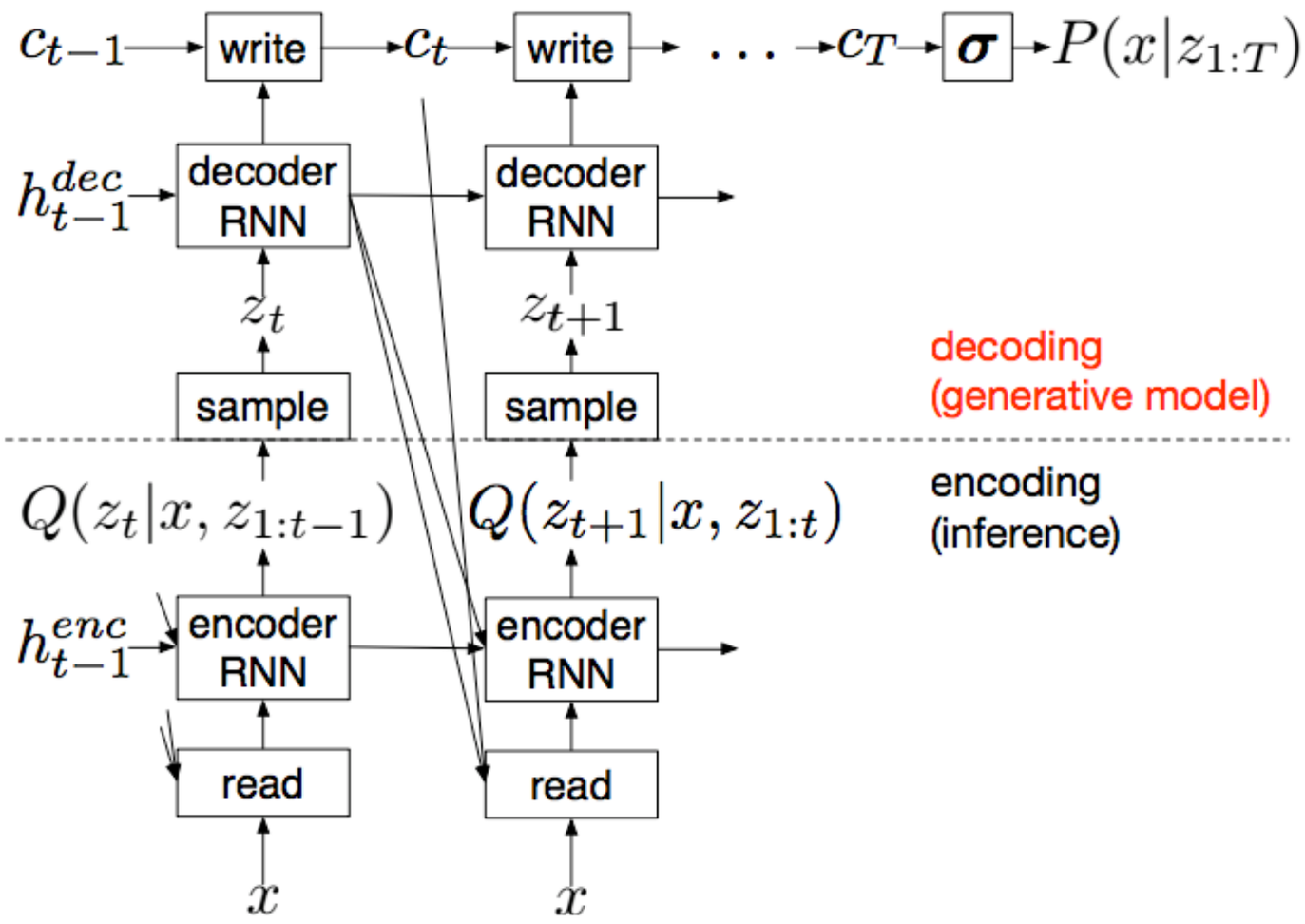
- Adds “attention” mechanism over a static input to define a sequential process



Gregor et. al. 2015

Vanilla VAE & DRAW





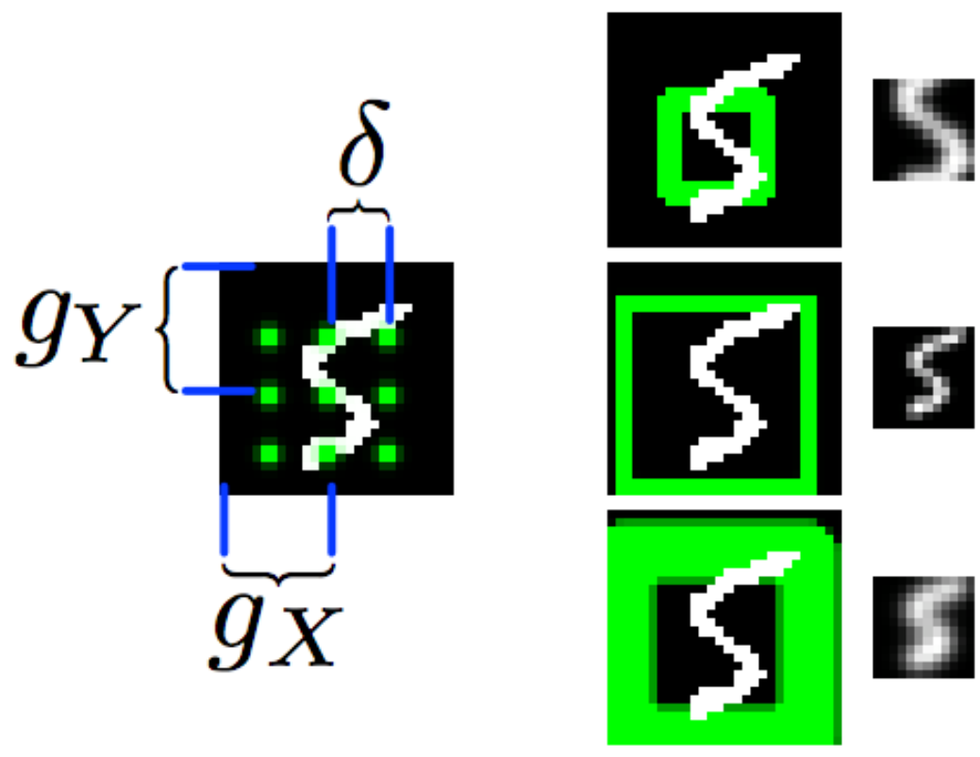
Variational Autoencoder

DRAW

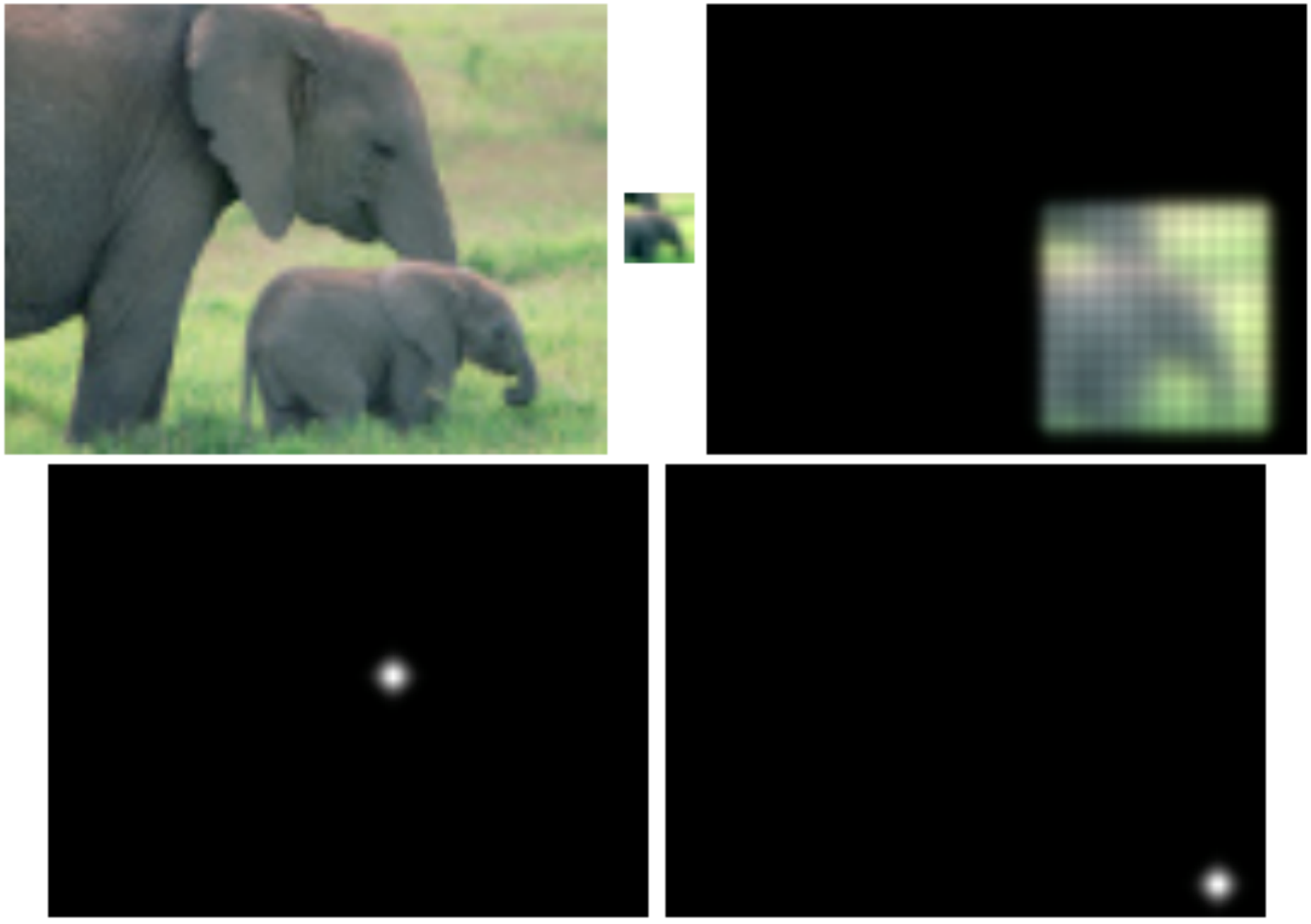
DRAW Attention Mechanism

- At each timestep model generates the coordinates and zoom of the patch it is going to read/write

- Based on that model creates grid of gaussian filters that read/write specific sub-patches.

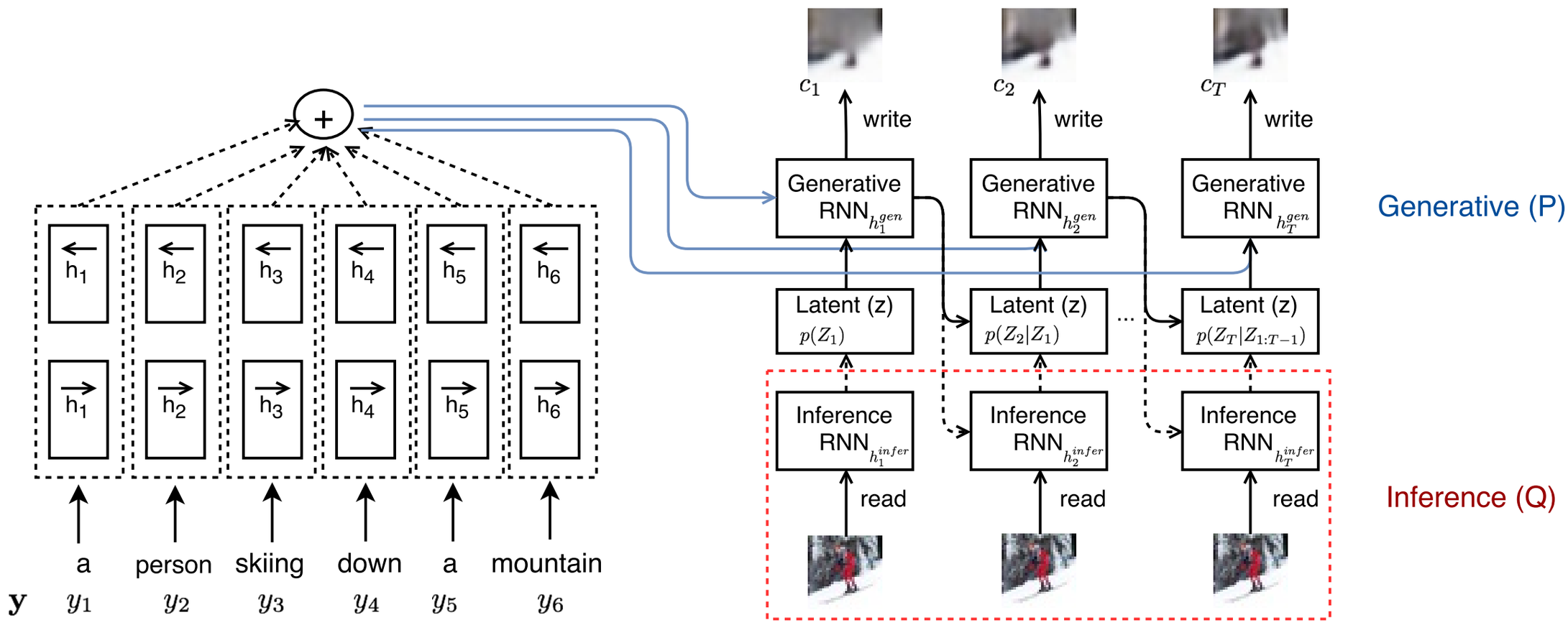


DRAW Attention Mechanism



Generating House Numbers

Generating Images from Captions



Model is trained to maximize variational lower bound

$$\mathcal{L} = \mathbb{E}_{Q(Z_{1:T}\,|\,\mathbf{y},\mathbf{x})}\left[\log p(\mathbf{x}\,|\,\mathbf{y}, Z_{1:T}) - \sum_{t=2}^{T} \mathrm{D_{KL}}\left(Q(Z_t\,|\,Z_{1:t-1}, \mathbf{y}, \mathbf{x}) \,\|\, P(Z_t\,|\,Z_{1:t-1}, \mathbf{y})\right)\right] - \mathrm{D_{KL}}\left(Q(Z_1\,|\,\mathbf{x}) \,\|\, P(Z_1)\right)$$

Mansimov et. al. 2015/16

Novel Compositions

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

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

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

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

Issues with Variational Autoencoders

- In practice, I found that simple feedforward encoder/decoders don't work well for challenging and diverse datasets.

- Except if you pre-train good classifier (say on ImageNet) and initialize encoder weights.

- Otherwise need to resort to more powerful encoder-decoder architectures like DRAW, diffusion models and etc.

- And/or more advanced inference techniques like importance weighted autoencoders, normalizing flows, HMC, etc.

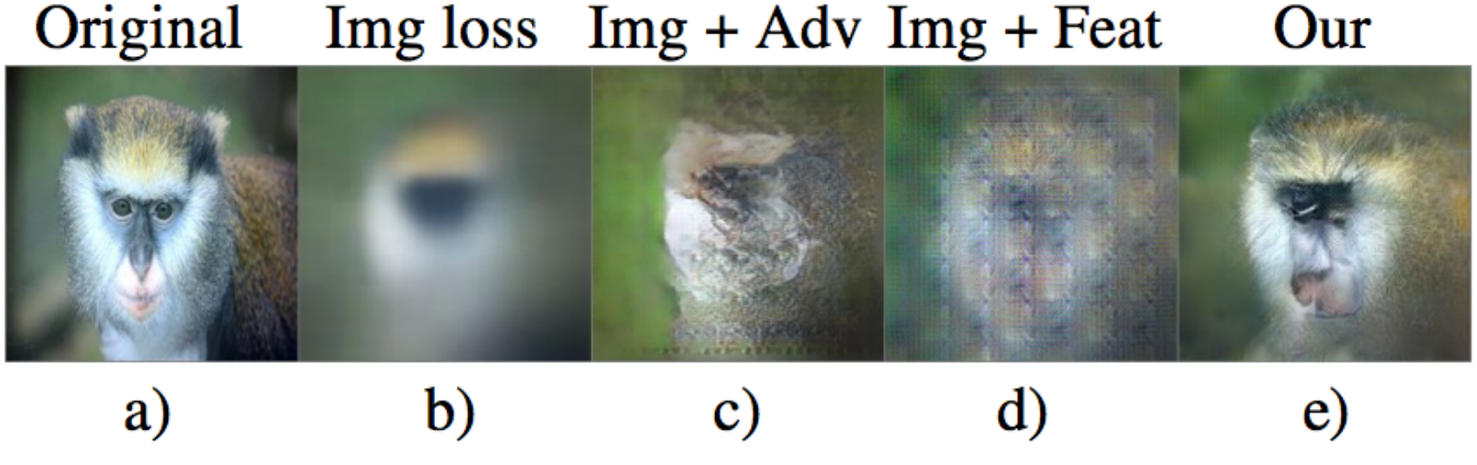
More issues with reconstruction term.

Simple pixel-wise reconstruction error is a bad cost function.

Simply shifting pixels can create a huge difference.

It starts averaging pixels over likely locations which leads to blurry output.

Ideally cost function should be more location invariant. Currently researchers trying to create better loss functions.



Dosovitskiy & Brox 2016

Generative Adversarial Networks

- Another way of avoiding normalization term and reconstruction loss.

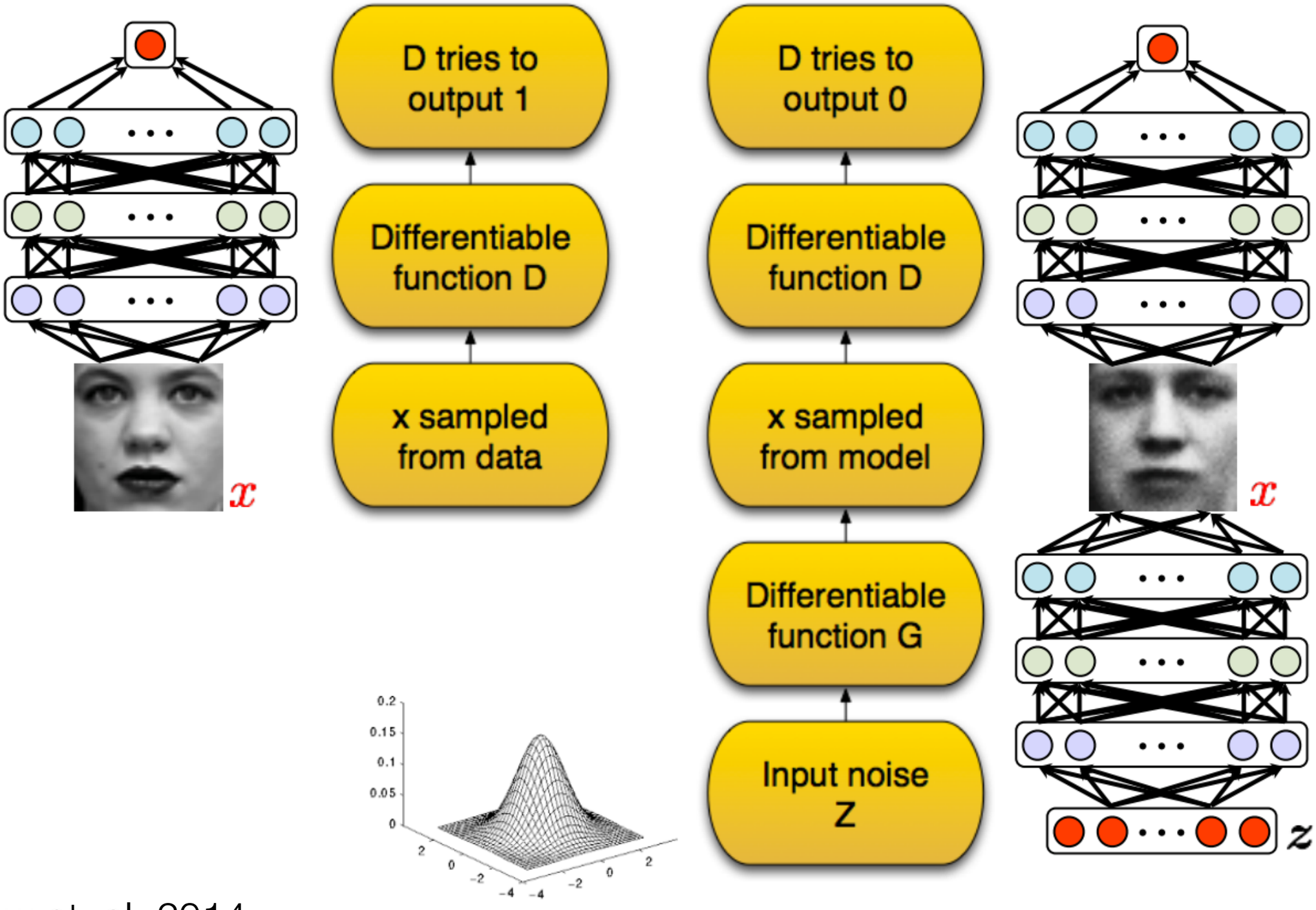
- Frame image generation as a game between two players
  - Discriminator **D**
  - Generator **G**

- **G** tries to trick **D** by generating samples that are hard for **D** to distinguish from data.

- **D** tries not to be tricked by **G** by trying to discriminate between real data and fake data.

Goodfellow et. al. 2014

Generative Adversarial Networks



Goodfellow et. al. 2014

Generative Adversarial Networks

- Minimax value function

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Generator pushes down

Discriminator pushes up

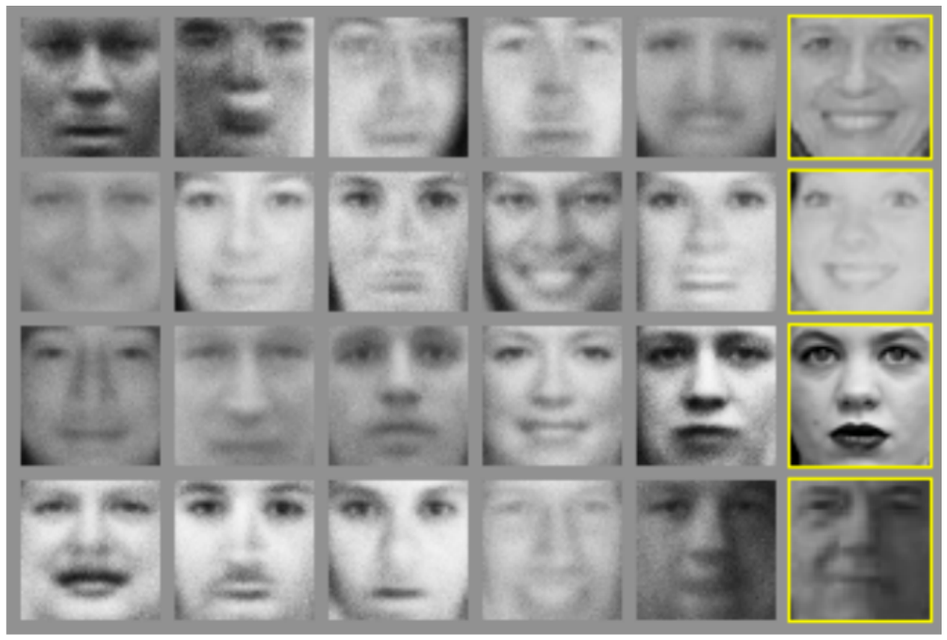
Discriminator's ability to recognize data as being real

Discriminator's ability to recognize generator samples as being fake

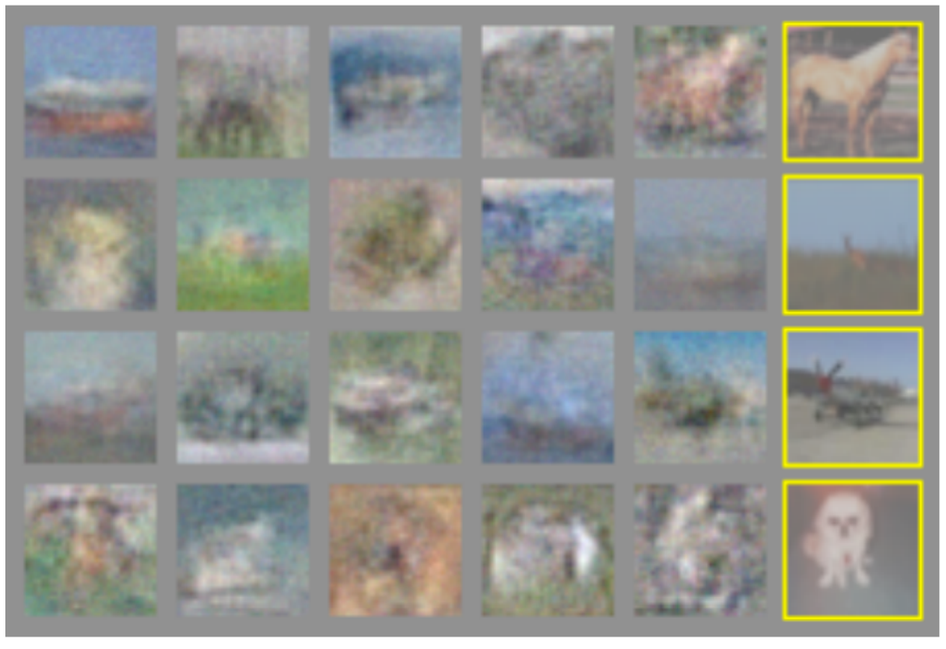
Goodfellow et. al. 2014

Generated Samples

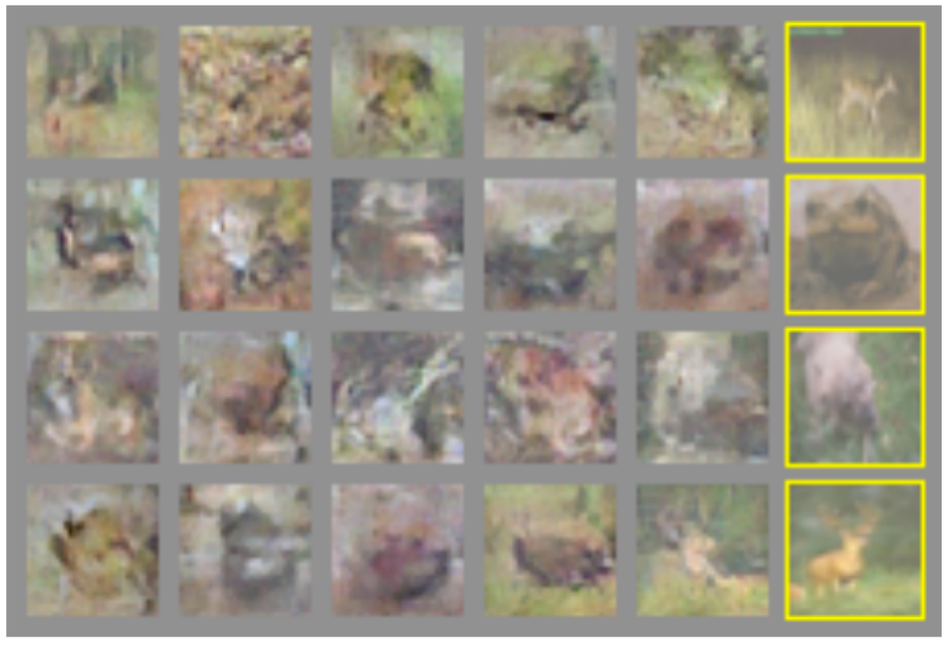
MNIST



TFD



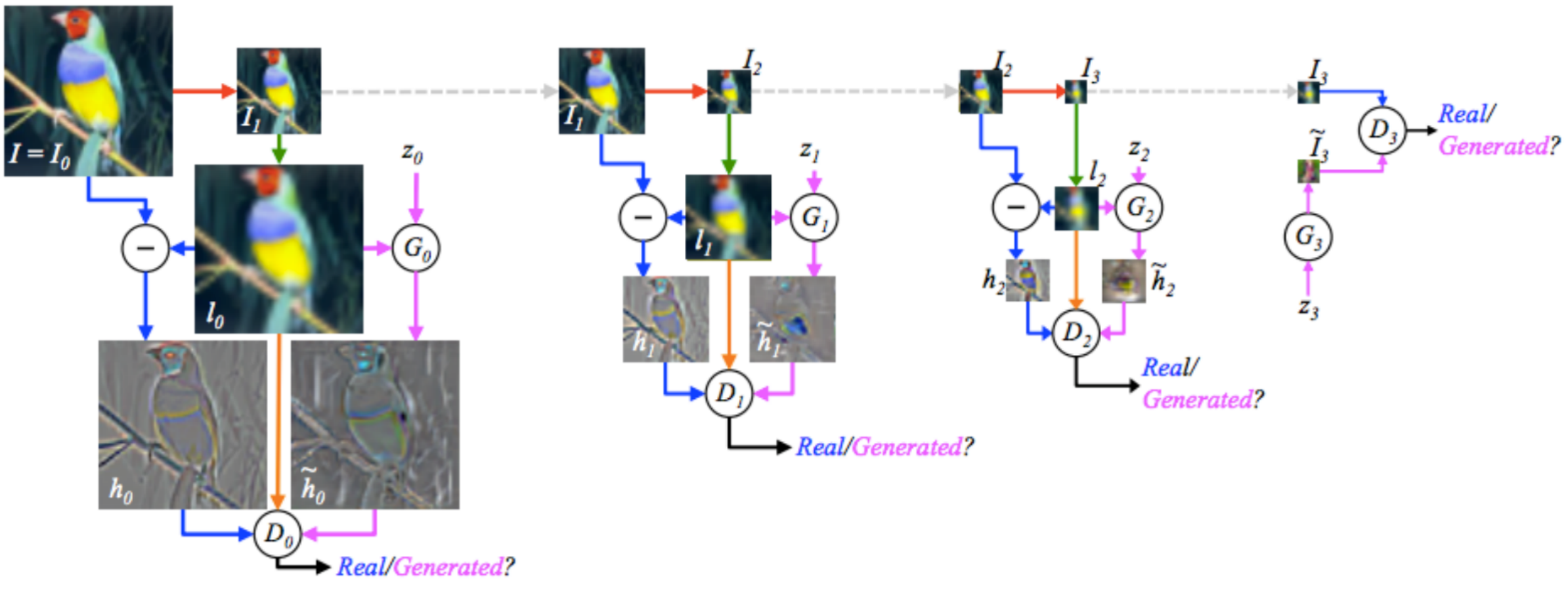
CIFAR-10 (fully connected)



CIFAR-10 (convolutional)

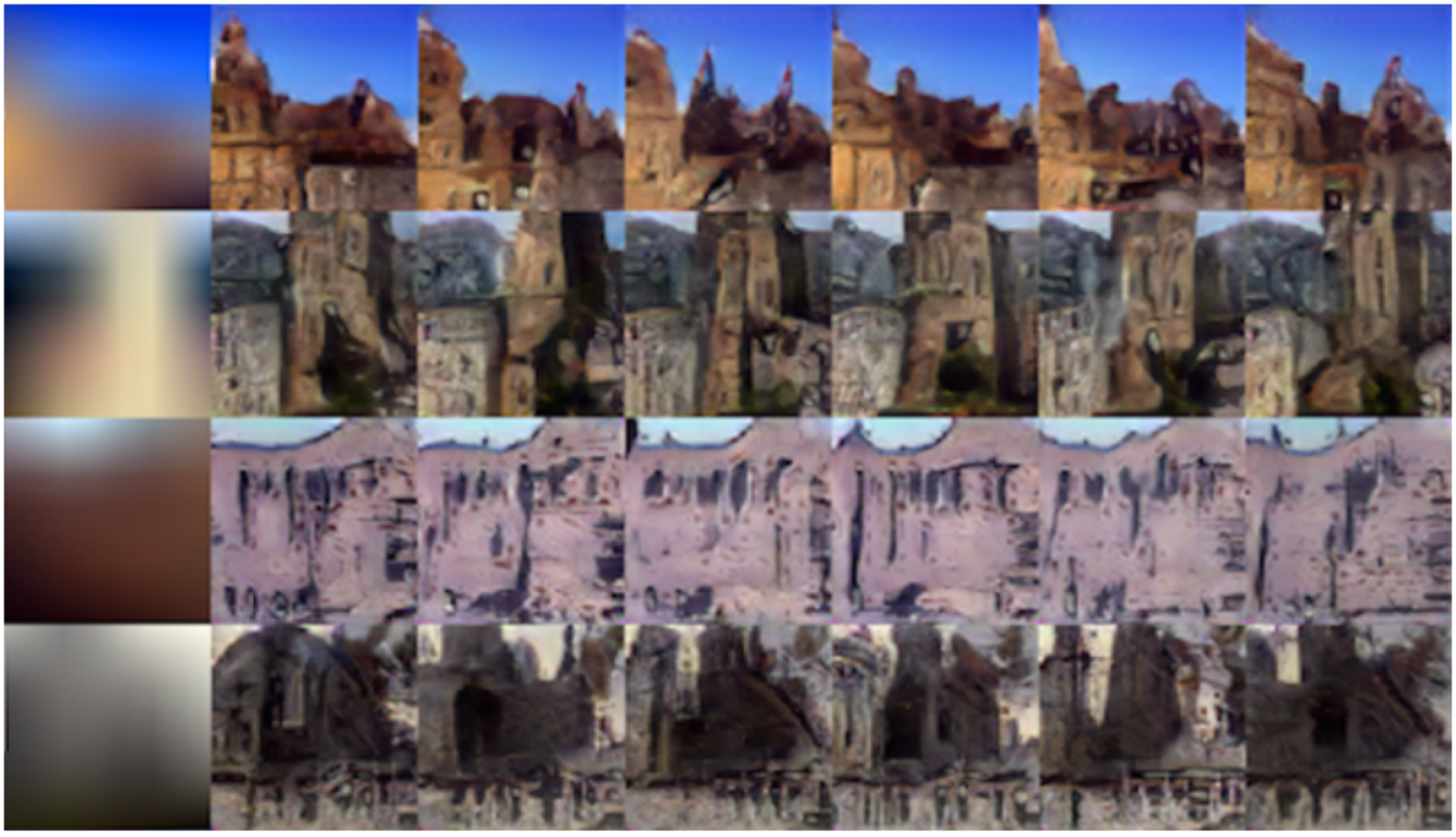
Laplacian Pyramid of Generative Adversarial Networks

- Instead of generating whole image, break down the image generation into several steps.



Denton & Chintala et. al. 2015

Example of course-to-fine generation on validation images



Generated Samples

40% of samples mistaken by humans as real.

Stabilizing training of GANs

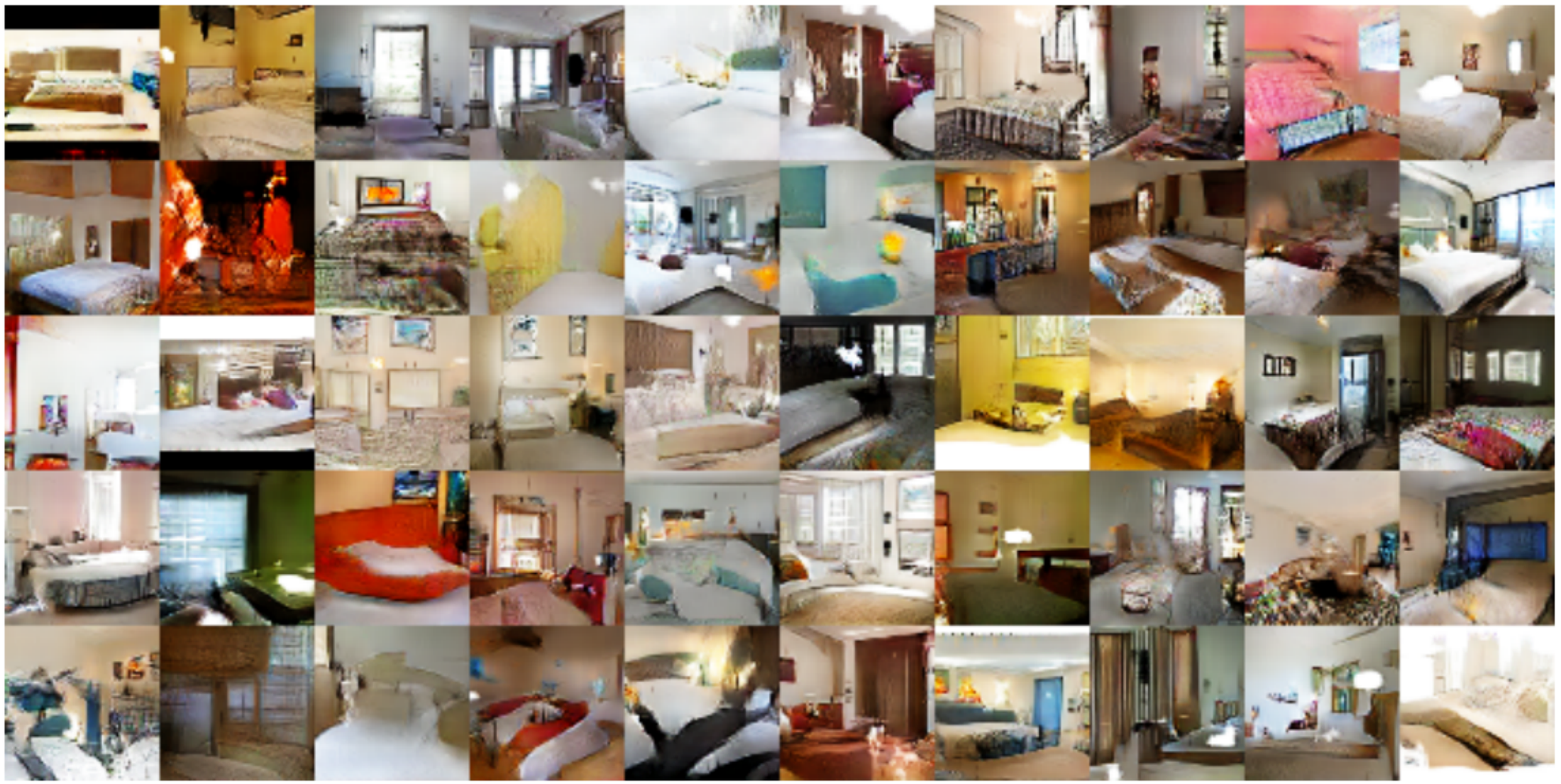
- GANs are unstable and hard to train.
- Discriminator can learn very fast to discriminate between real and fake data and generator might not catch up.
- Generator might get stuck at specific model of data distribution repeat the same textures.

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Radford et. al. 2015/16

Generated Samples



Radford et. al. 2015/16

Issues with Generative Adversarial Networks

- Still with those tricks, training can be unstable and need to monitor samples carefully.

- Also end-to-end backpropogation though GAN framework doesn't work for non-continuous data like text.

Conclusions

- There many more interesting extensions and modifications of these models that I haven't mentioned.

- Each generative model has its pros and cons and there is no clear winner.

- There is a potential to create a hybrid of different approaches.

- Or create completely new approaches from scratch.