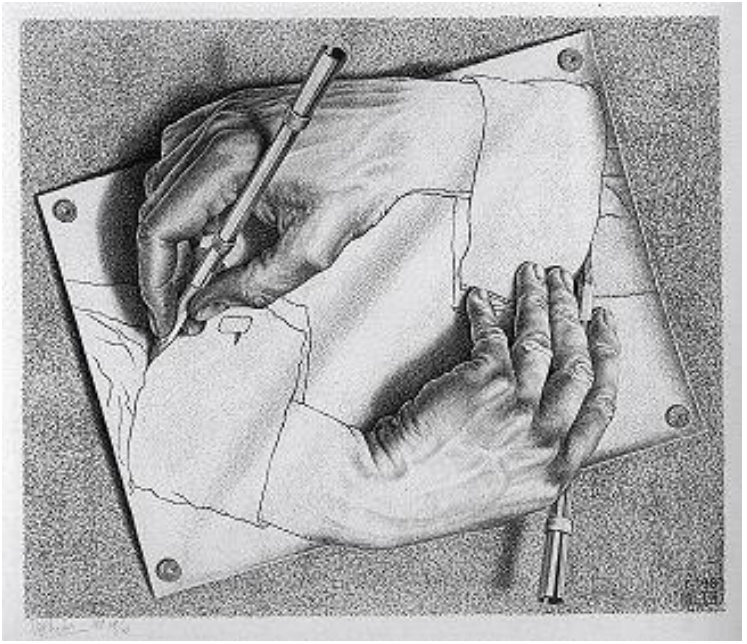
Advanced Generation Methods

Hsiao-Ching Chang, Ameya Patil, Anand Bhattad



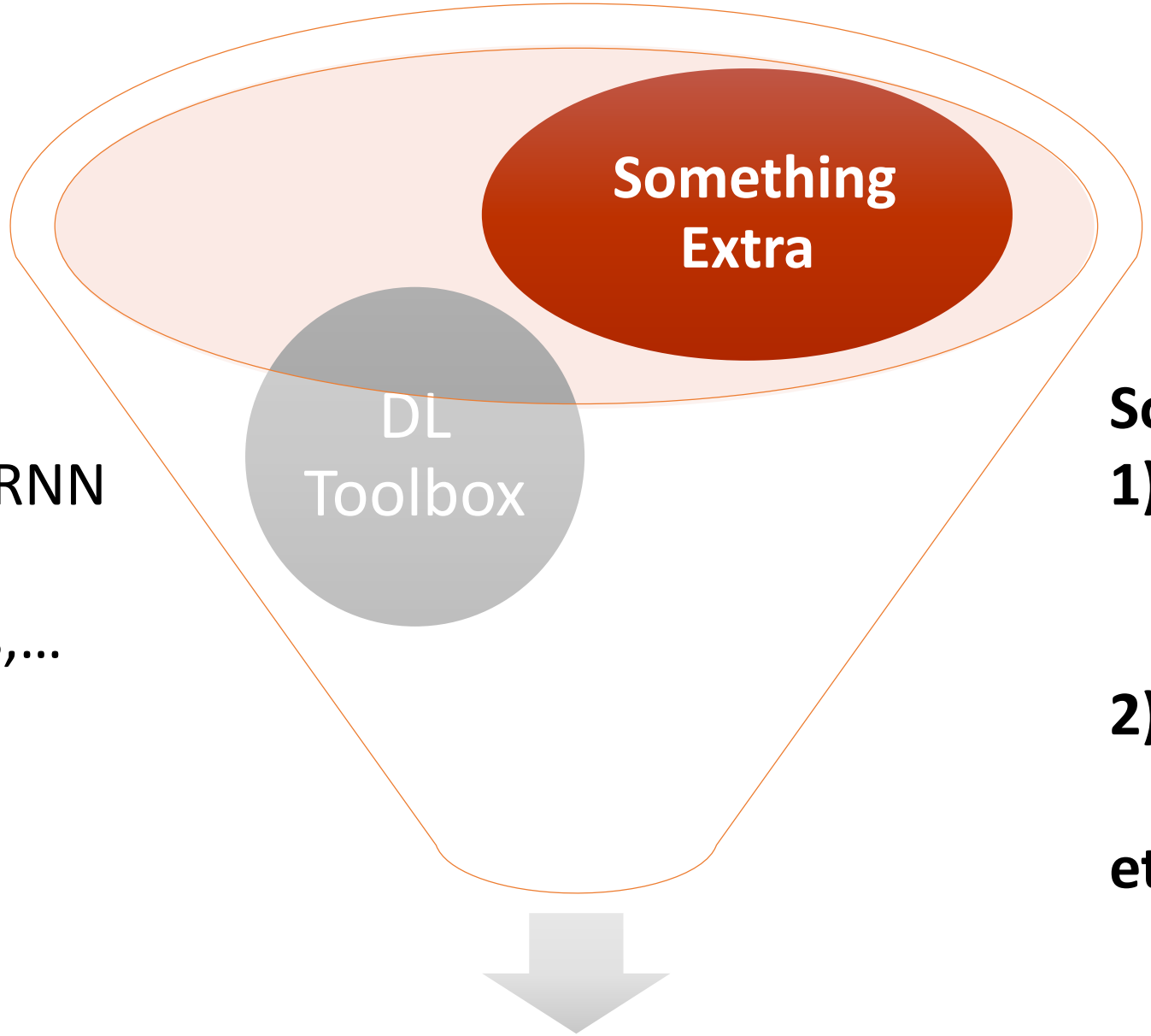
M. C. Escher, 1948

“What I cannot create, I do not understand.” —Richard Feynman

DL Toolbox:

1) DNN, CNN & RNN
2) Variational autoencoders,…
3) Adam, SGD, dropout, …

etc……



Something Extra:

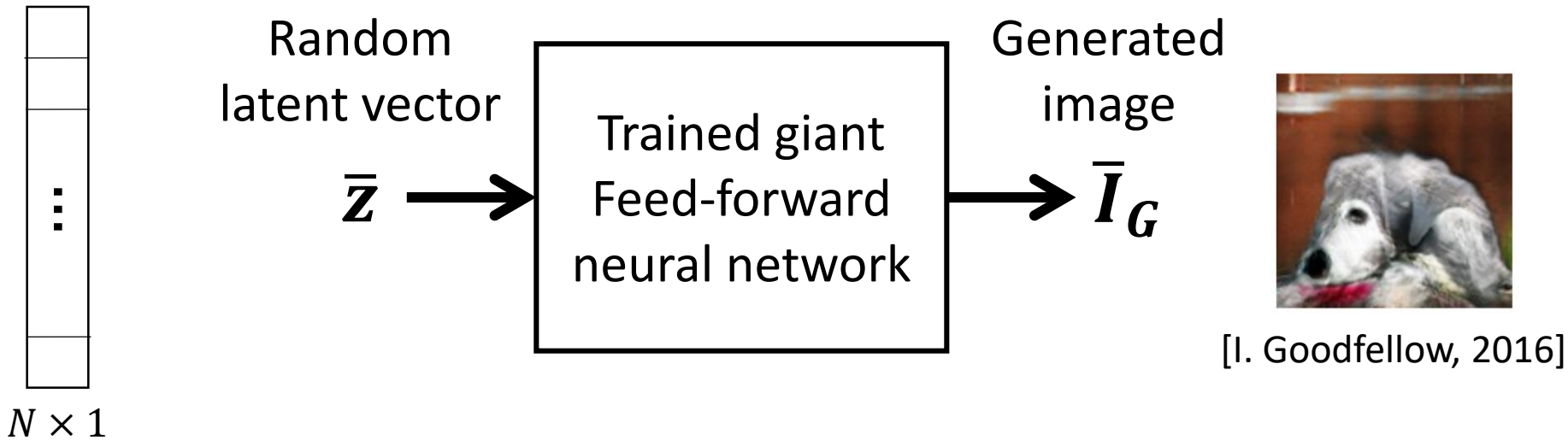
1) **Intuitions and insights into the problems**
2) **Ways of putting DL tools together**

etc……

Advanced Generation Methods

Image Generation: How Machines do it

The techniques that we learned: **GANs and VAEs**



[I. Goodfellow, 2016]

They attempt to generate image in one-shot projection !!

Image Generation: How We Do It

We pay **attention** on each subpart, we **iterate** in a **feedback** loop

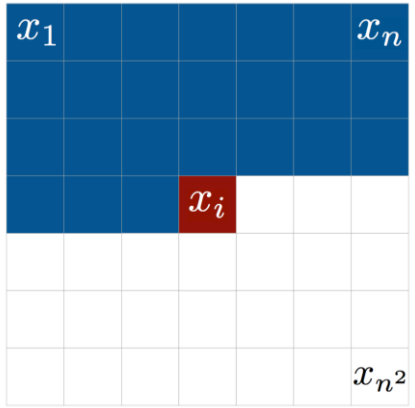
Can we teach machines to do the same ?

Image credit: Easy Drawing Tutorial 2016 App, Google Play

Advanced Generation Methods:

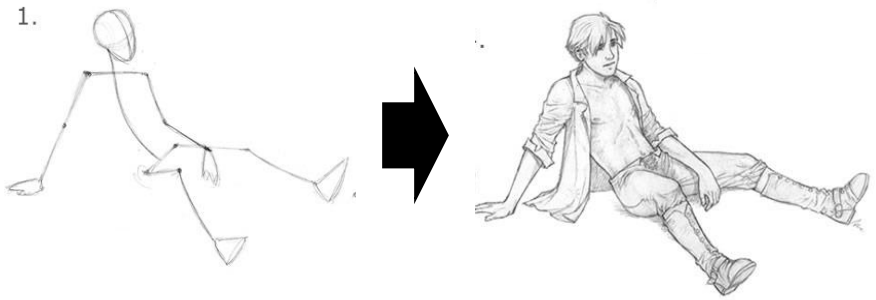
- Pixel-by-pixel generation:

A simple way to iterate, employ feedback and capture pixel dependencies



- Iterative attentive generation:

More advanced techniques involving iterative formation of an abstract schema



Pixel-by-pixel generation:

Outline

- Intuition
- Basic models
  - PixelRNN
  - PixelCNN
- Variants of PixelRNN and PixelCNN
  - Multi-scale version
  - Conditional image generation
  - Other recent improvements

Outline

- Intuition
- Basic models
  - PixelRNN
  - PixelCNN
- Variants of PixelRNN and PixelCNN
  - Multi-scale version
  - Conditional image generation
  - Other recent improvements

Intuition …(A customary CAT slide!)

How to include statistical dependencies over hundreds of pixels?

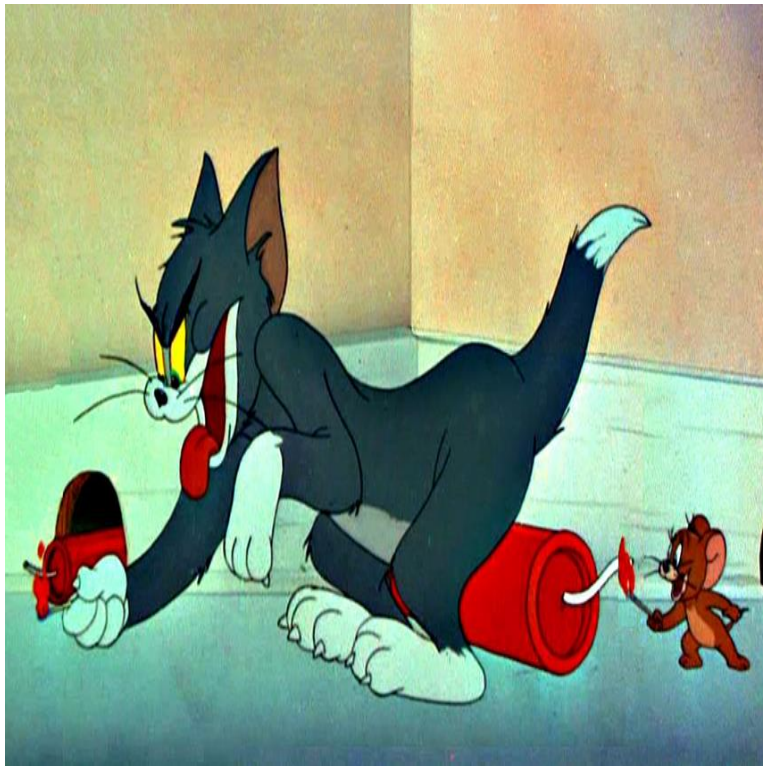
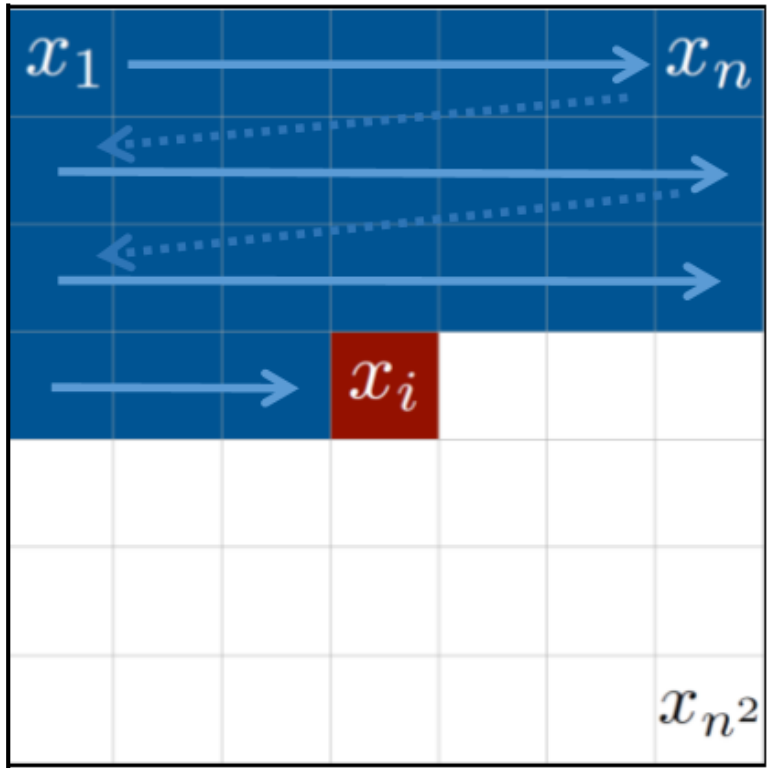


Image Source: http://wallpapers.ae/wp-content/uploads/2016/01/Tom-And-Jerry-Funny-Wallpaper.jpg

Intuition



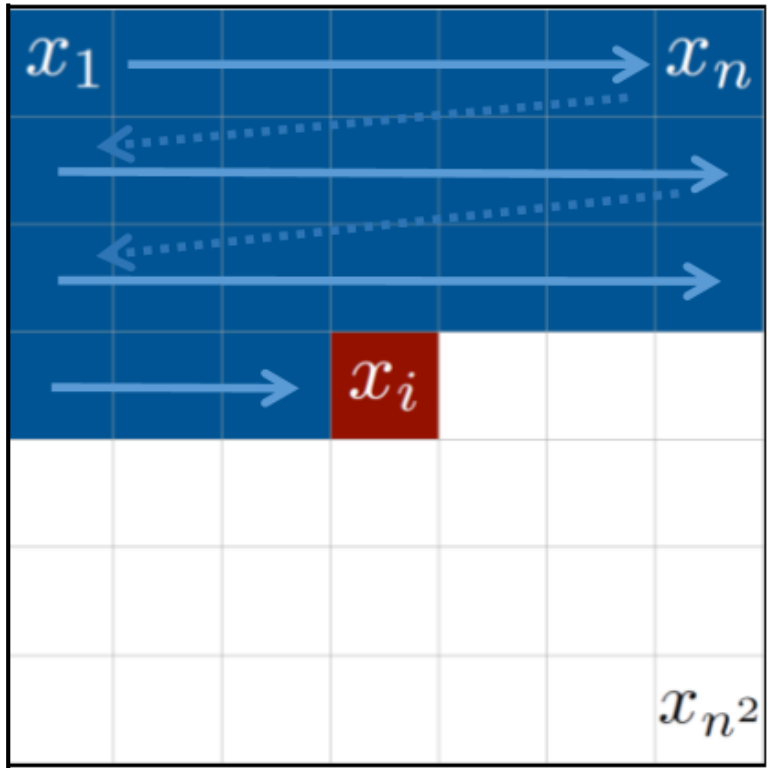
Pixel recurrent neural networks, ICML 2016

Intuition

$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_{n^2})$$

Bayes Theorem:

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$



Pixel recurrent neural networks, ICML 2016

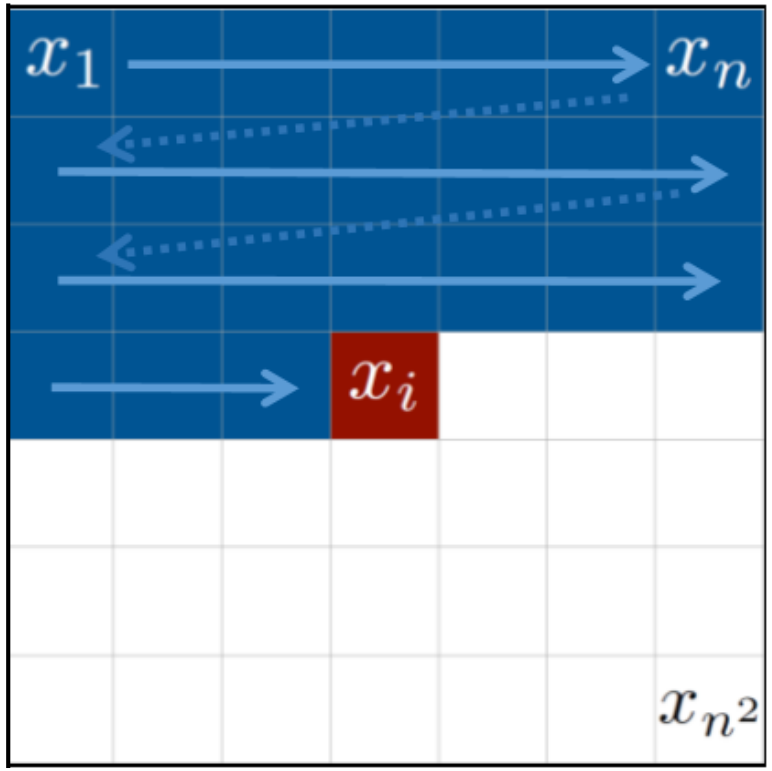
Intuition

$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_{n^2})$$

Bayes Theorem:

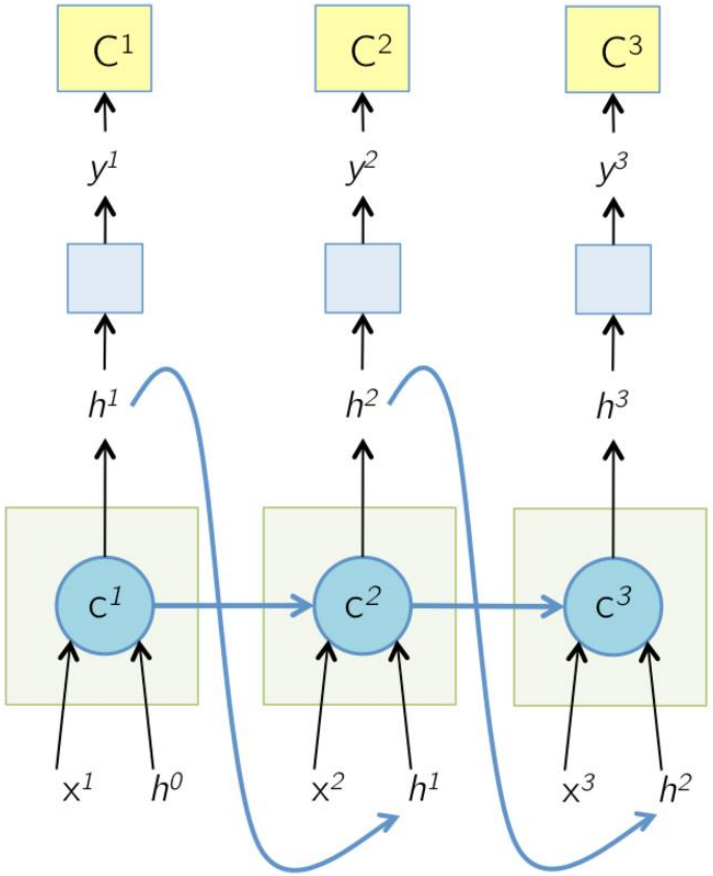
$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

A sequential model!



Pixel recurrent neural networks, ICML 2016

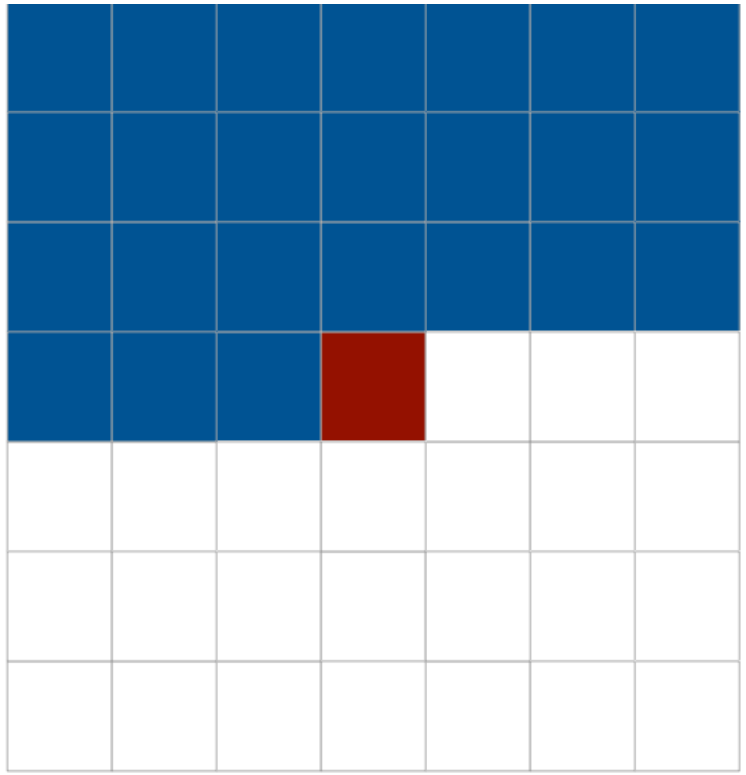
LSTM



http://arunmallya.github.io/writeups/nn/lstm/index.html#/6

Intuition

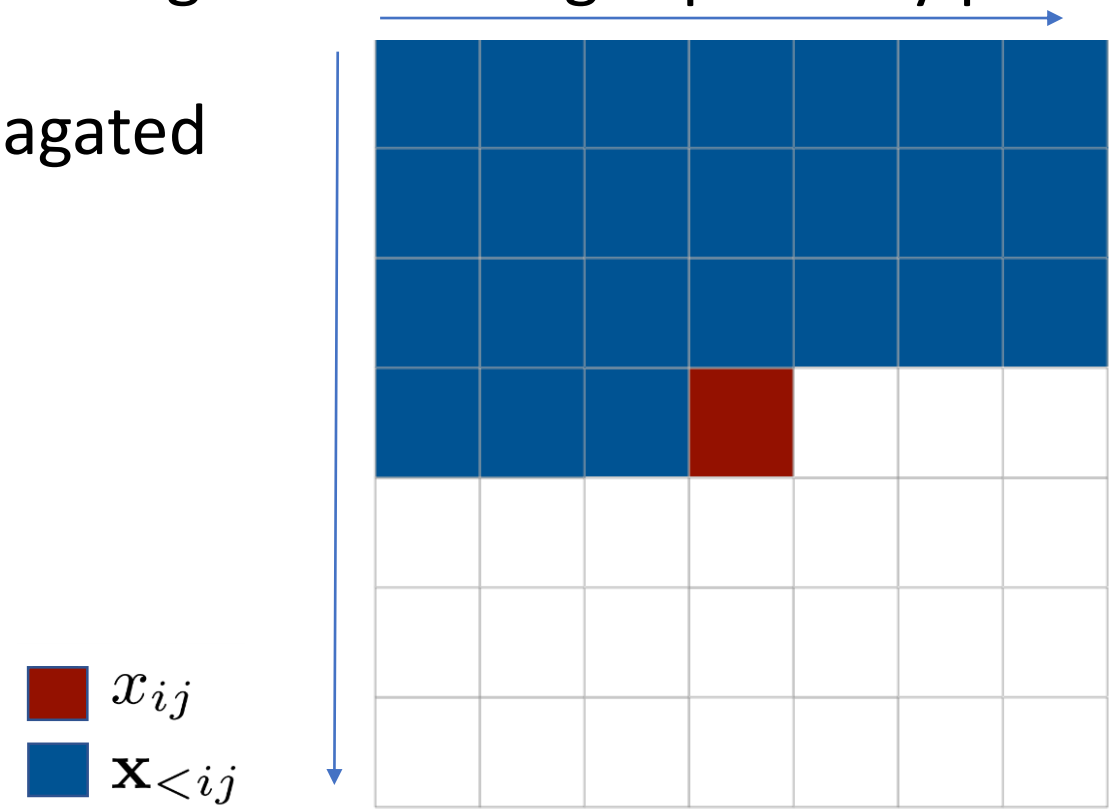
- Question: Can we use plain-LSTM to generate images pixels by pixels?



Pixel recurrent neural networks, ICML 2016

Intuition

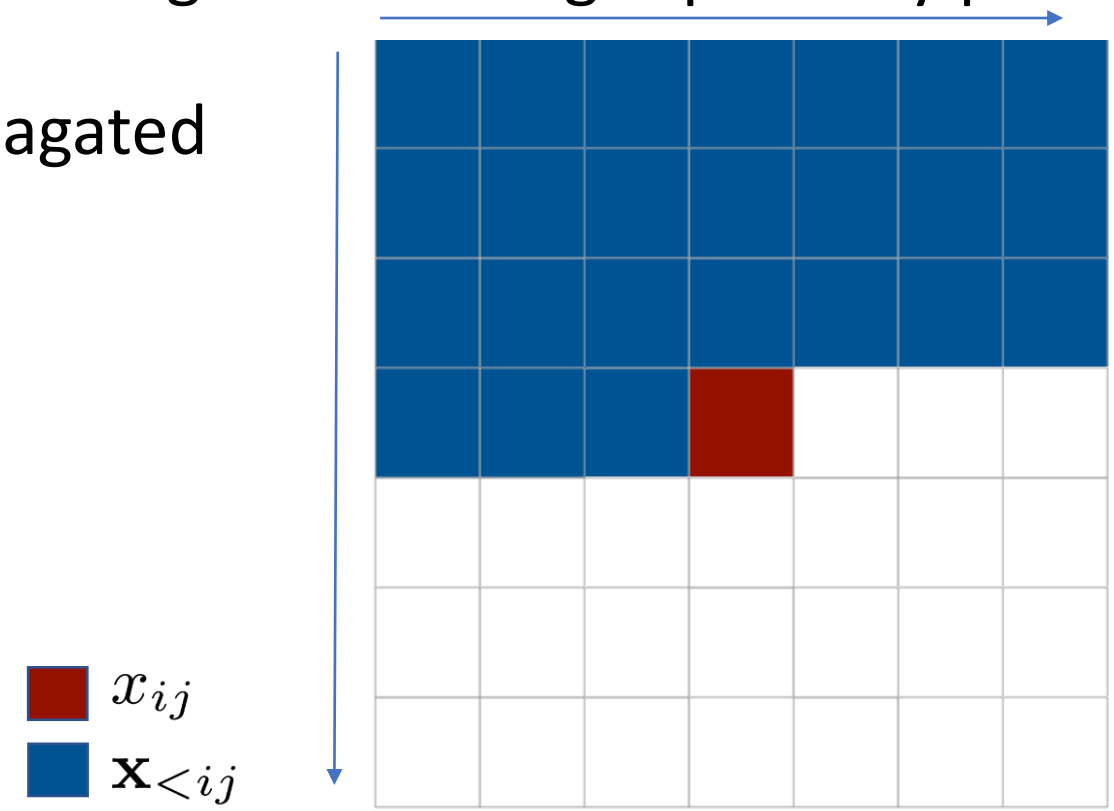
- Question: Can we use plain-LSTM to generate images pixels by pixels?
- Ensure information is well propagated in two dimensions



Pixel recurrent neural networks, ICML 2016

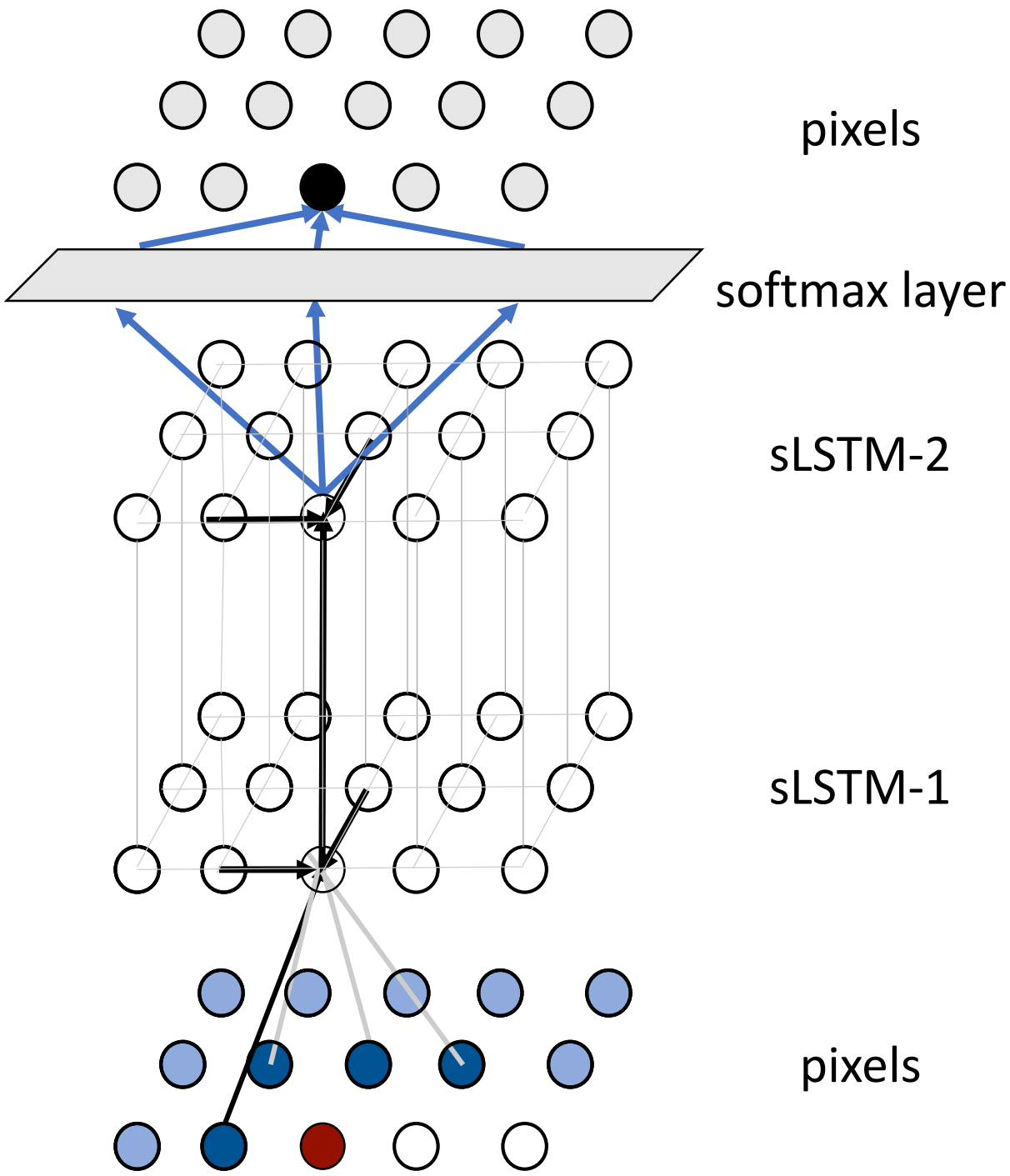
Intuition

- Question: Can we use plain-LSTM to generate images pixels by pixels?
- Ensure information is well propagated in two dimensions
- spatial LSTM (sLSTM)



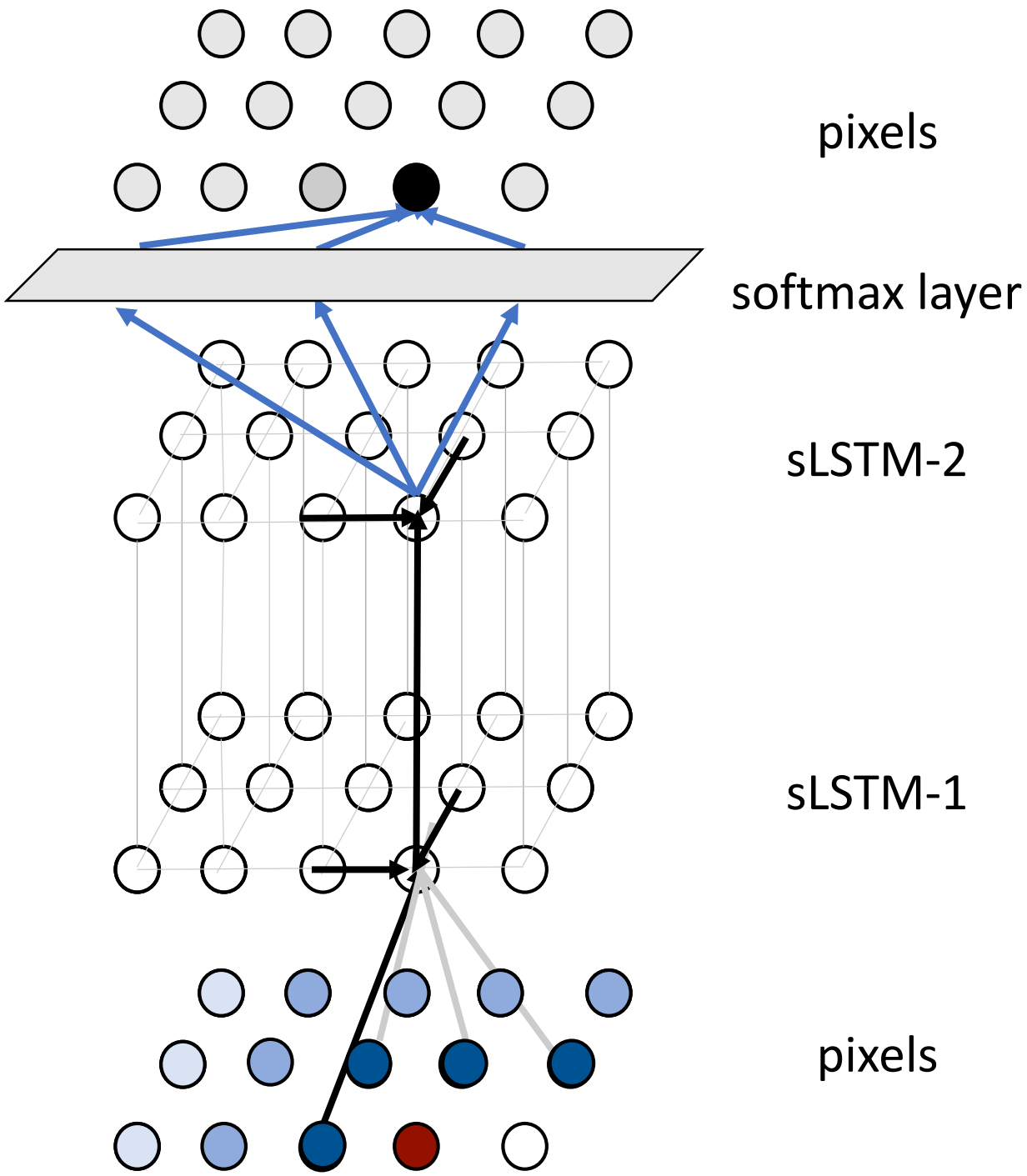
Pixel recurrent neural networks, ICML 2016

Spatial LSTM



Adapted from: Generative image modeling using spatial LSTM. Theis & Bethge, 2015

Spatial LSTM



Adapted from: Generative image modeling using spatial LSTM. Theis & Bethge, 2015

Details about Soft Max

- Treat pixels as discrete variables:
  - To estimate a pixel value, do classification in every channel (256 classes indicating pixel values 0-255)
  - Implemented with a final softmax layer

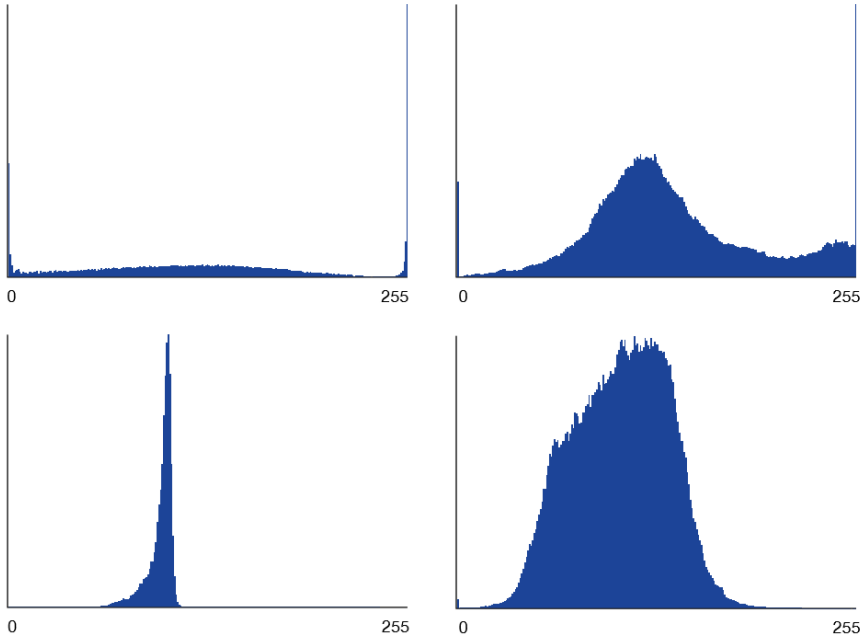
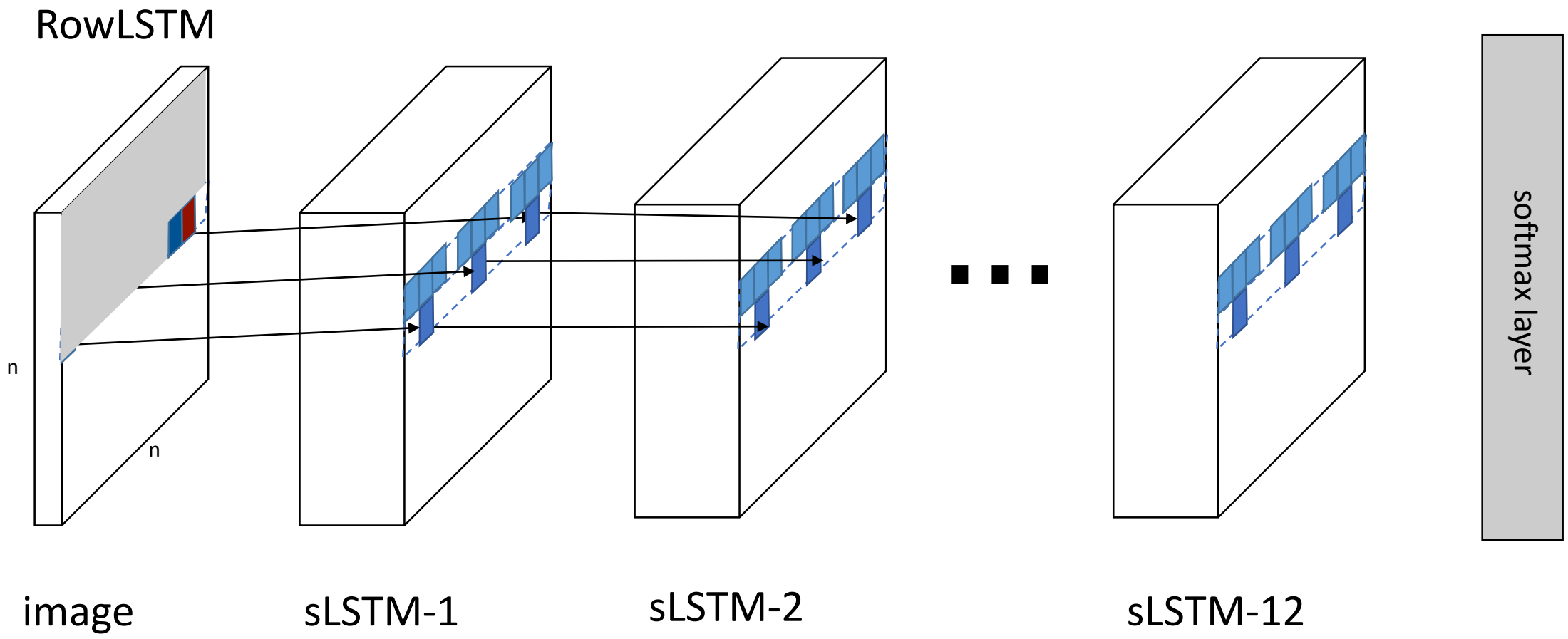


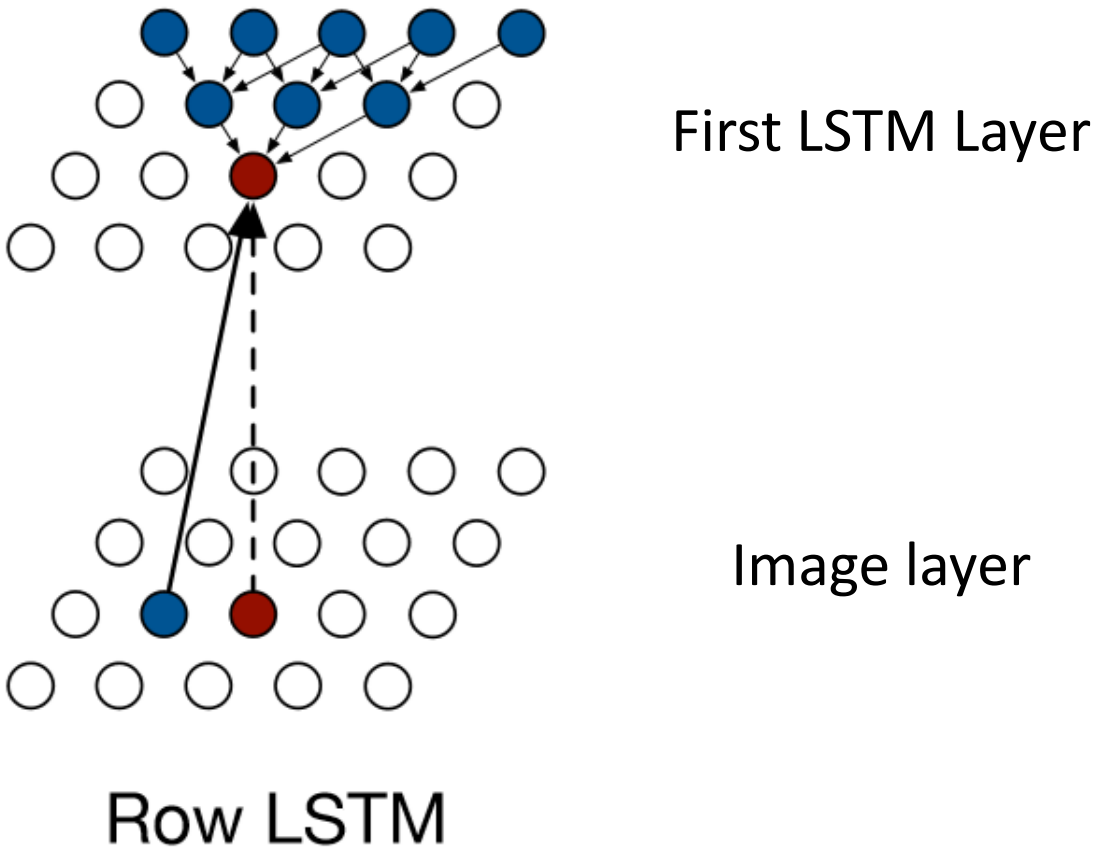
Figure: Example softmax outputs in the final layer, representing probability distribution over 256 classes.

Figure from: Oord et al.

PixelRNN: A specific Multidimensional LSTM



Pixel recurrent neural networks, ICML 2016



Pixel recurrent neural networks, ICML 2016

PixelRNN: A specific Multidimensional LSTM

Diagonal LSTM

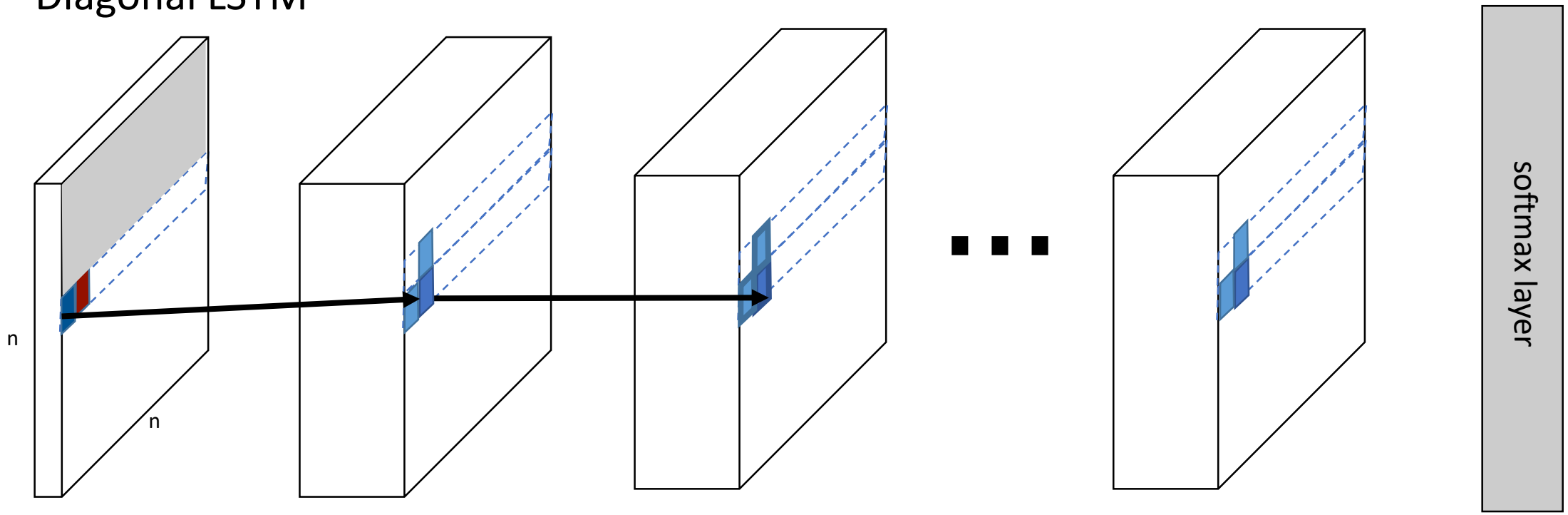
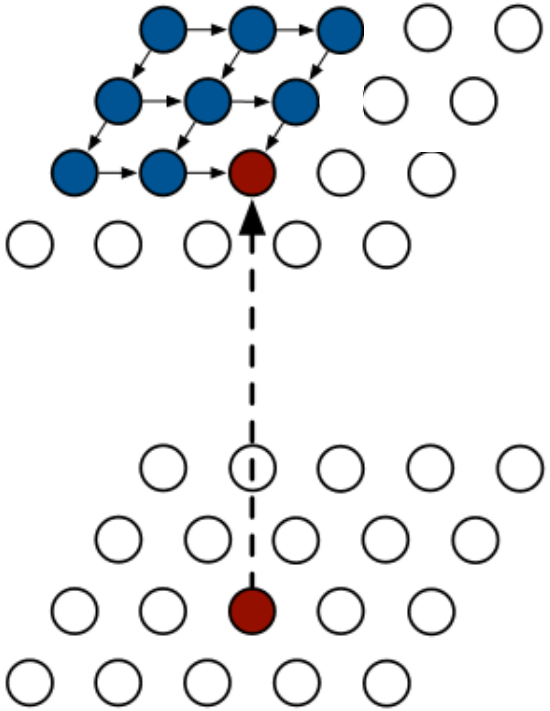


image sLSTM-1 sLSTM-2 sLSTM-12

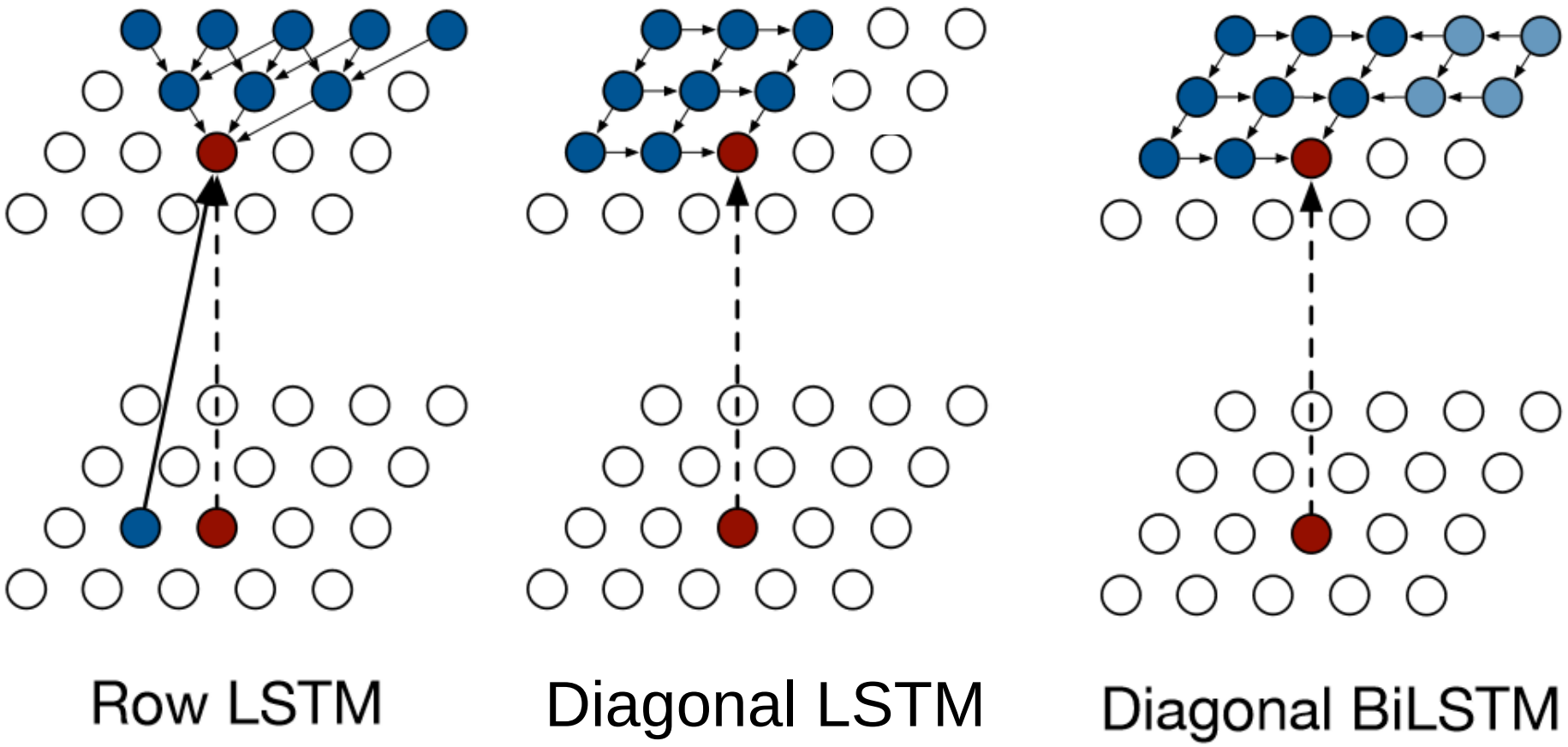
[Pixel recurrent neural networks](), ICML 2016



Diagonal LSTM

Pixel recurrent neural networks, ICML 2016

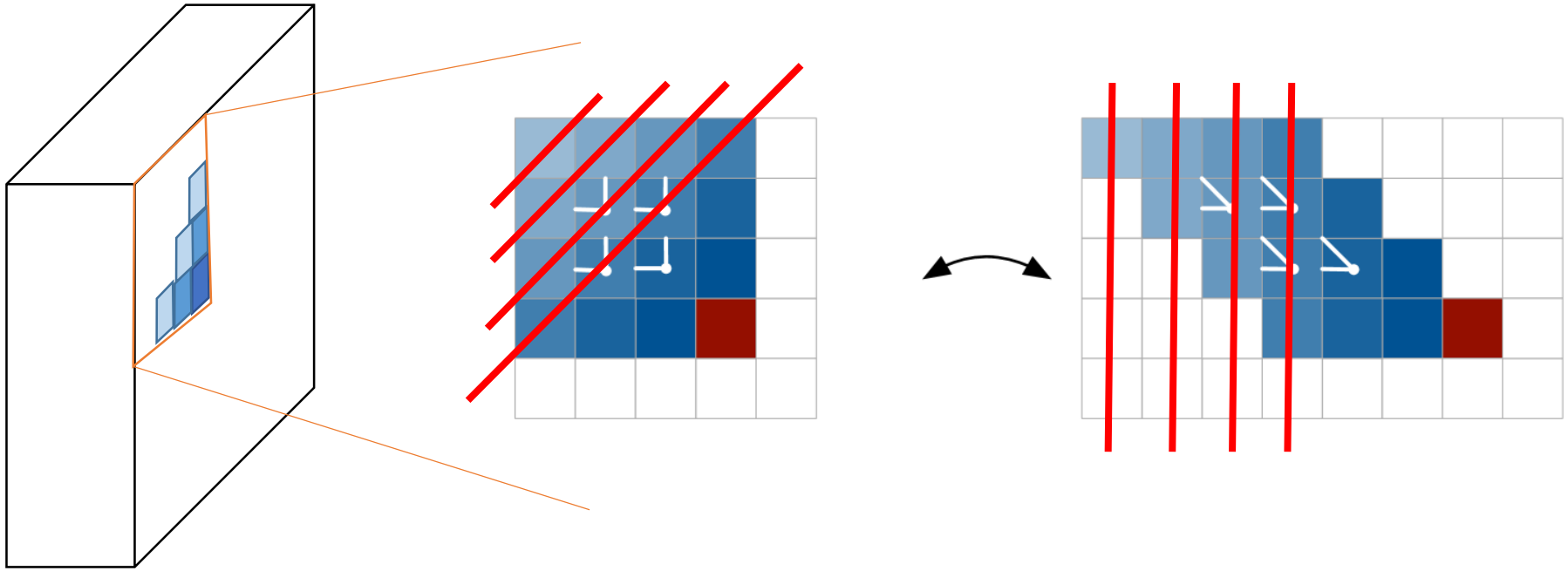
Receptive Field



Pixel recurrent neural networks, ICML 2016

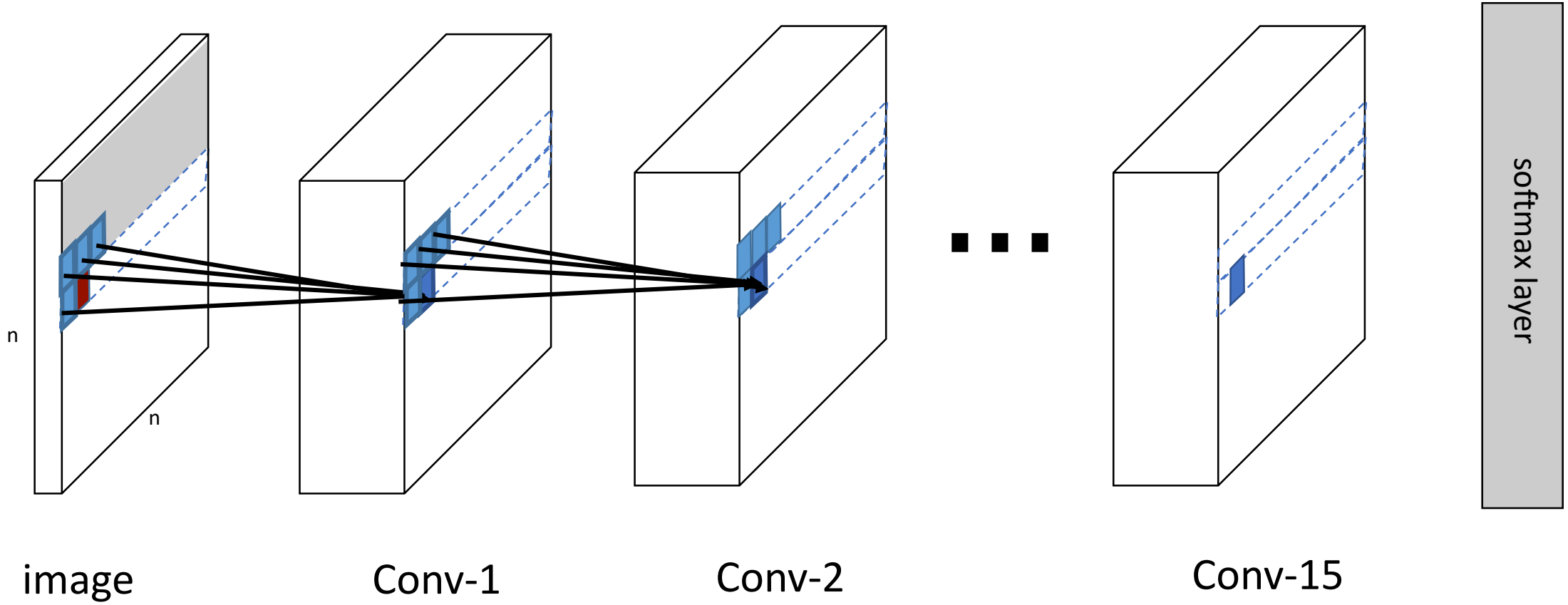
Diagonal LSTM

- To optimize, we skew the feature maps so it can be parallelized



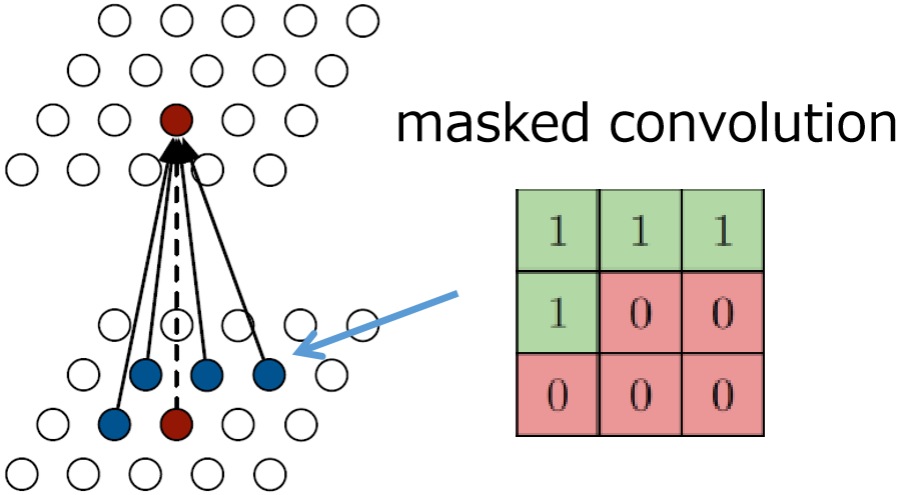
Pixel recurrent neural networks, ICML 2016

PixelCNN



PixelCNN

- 2D convolution on previous layer
- Apply masks so a pixel does not see future pixels (in sequential order)



Pixel recurrent neural networks, ICML 2016

Comparison

PixelCNN	PixelRNN – Row LSTM	PixelRNN – Diagonal BiLSTM
Full dependency field	Triangular receptive field	Full dependency field
Fastest	Slow	Slowest
Worst log-likelihood	-	Best log-likelihood

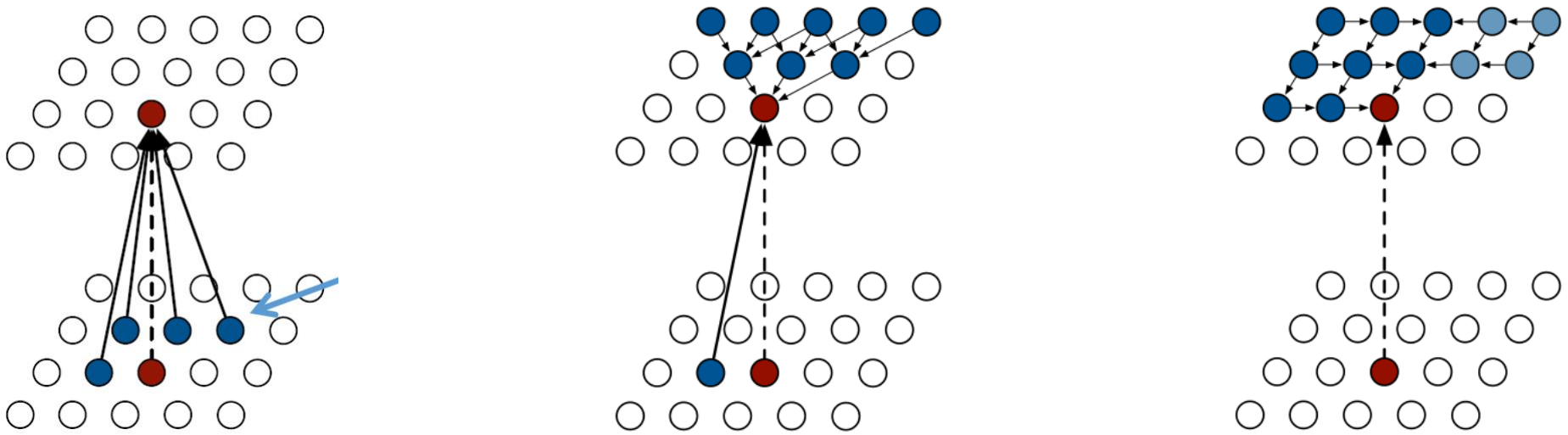
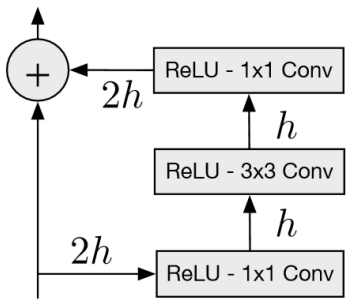


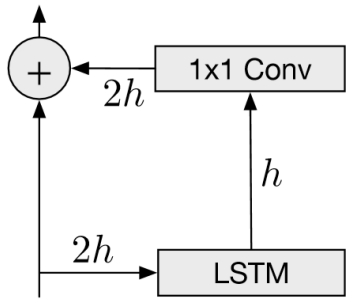
Figure from: Oord et al.

Architecture

- Residual connections

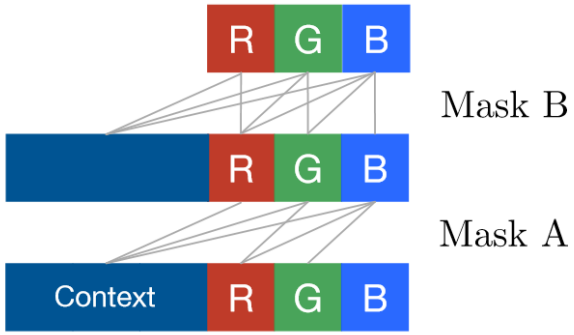


PixelCNN



PixelRNN

- Channel masks
  - Sequential order: R→G→B
  - Used in input-to-state convolutions
  - Two types of masks:



- Channels are connected to themselves
- Used in all other subsequent layers

- Channels are **not** connected to themselves
- Only used in first layer

Figure from: [Oord et al.]

Architecture

PixelCNN	Row LSTM	Diagonal BiLSTM
7×7 conv mask A		
Multiple residual blocks: (see fig 5)		
Conv 3×3 mask B	Row LSTM i-s: 3×1 mask B s-s: 3×1 no mask	Diagonal BiLSTM i-s: 1×1 mask B s-s: 1×2 no mask
ReLU followed by 1×1 conv, mask B (2 layers)		
256-way Softmax for each RGB color (Natural images) or Sigmoid (MNIST)		

Figure from: Oord et al.

Results

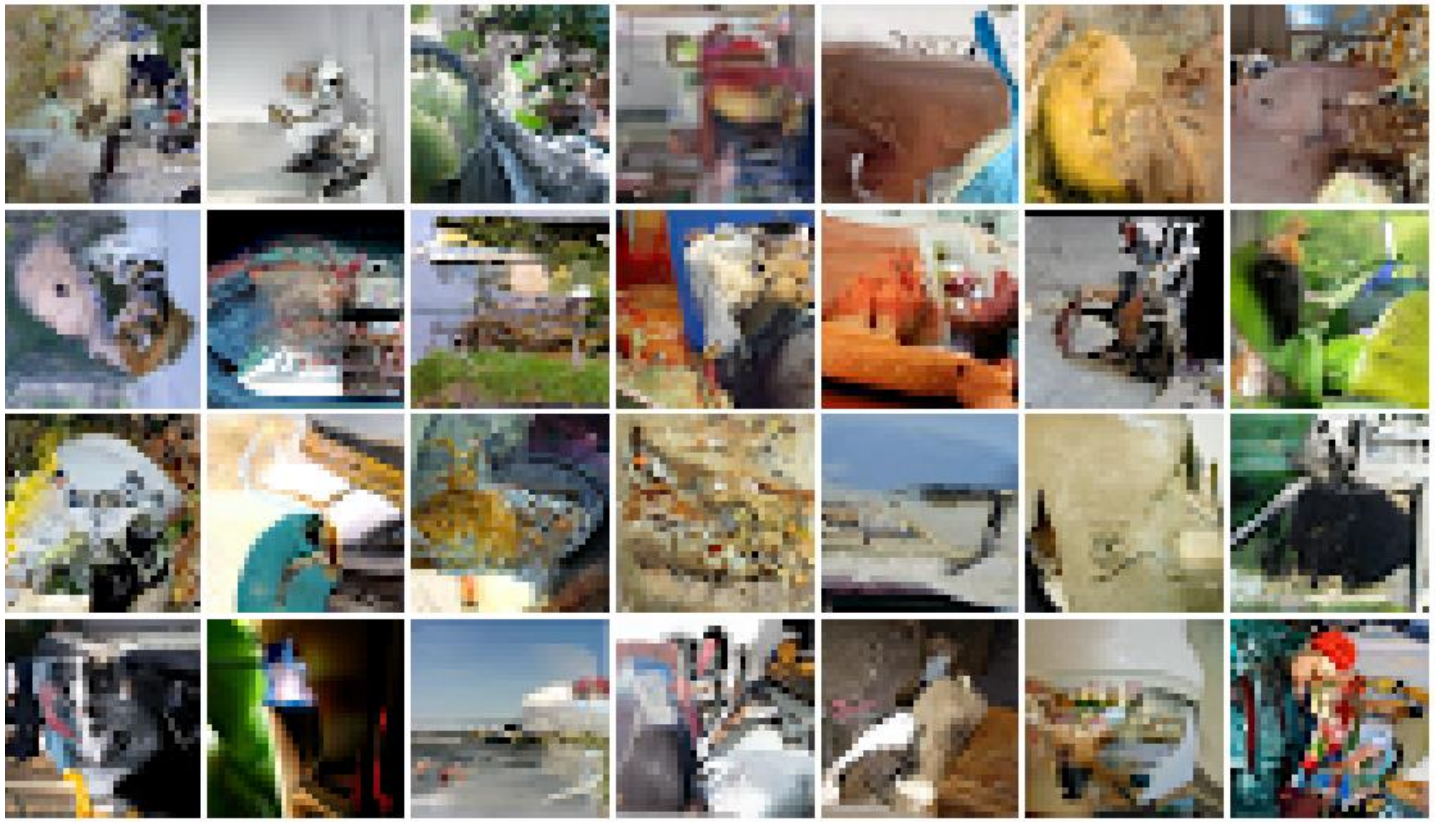


Figure: 32x32 ImageNet results from Diagonal BiLSTM model.

Figure from: Oord et al.

Outline

- Intuition
- Basic models
  - PixelCNN
  - PixelRNN
- Variants of PixelRNN and PixelCNN
  - Multi-scale version
  - Conditional image generation
  - Other recent improvements

Multi-scale PixelRNN

- Take subsampled pixels as additional input pixels
- Can capture better global information (visually more coherent)
- Performance is similar to normal one

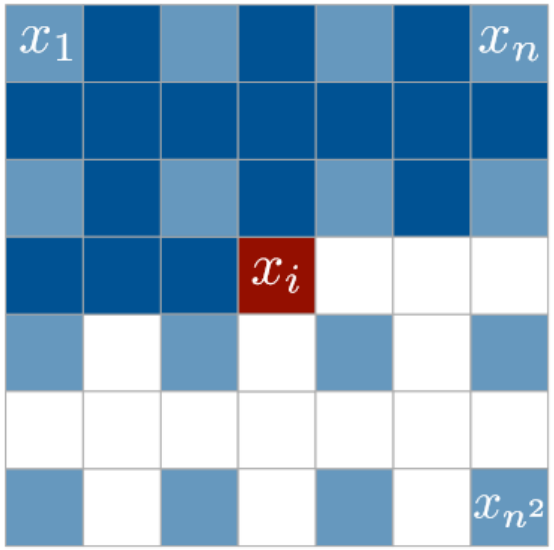


Figure from: Oord et al.

Multi-scale PixelRNN

Figure: 64x64 ImageNet results from normal Diagonal BiLSTM model (left) and multi-scale model (right).

Figure from: Oord et al.

Conditional Image Generation

- Given a high-level **image description vector $\boldsymbol{h}$**

$$p(\mathbf{x}) = p(x_1, x_2, ..., x_{n^2})$$

$$\downarrow$$

$$p(\mathbf{x}|\mathbf{h}) = p(x_1, x_2, ..., x_{n^2}|\mathbf{h})$$

Conditional image generation with pixelcnn decoders. NIPS 2016

Conditional Image Generation

- $\boldsymbol{h}$ is location-independent
  - For example,
    - One-hot encoding representing a specific class
    - Latent representation embedding
  - Model joint probability conditioned on $\boldsymbol{h}$

additional input

$$p(\mathbf{x}|\mathbf{h}) = \prod_{i=1}^{n^2} p(x_i|x_1, ..., x_{i-1}, \mathbf{h})$$

$$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x} + V_{k,f}^T \mathbf{h}) \odot \sigma(W_{k,g} * \mathbf{x} + V_{k,g}^T \mathbf{h})$$

dot product

dot product

Conditional image generation with pixelcnn decoders. NIPS 2016

Conditional Image Generation

- $\boldsymbol{h}$ is location-dependent
  - $\boldsymbol{h}$ contains both object and location information
  - Use an additional deconvolutional neural network to estimate $\boldsymbol{s} = m(\boldsymbol{h})$, where $\boldsymbol{s}$ has same size as images

$$p(\mathbf{x}|\mathbf{h}) = \prod_{i=1}^{n^2} p(x_i|x_1, ..., x_{i-1}, \mathbf{h})$$

$$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x} + V_{k,f} * \mathbf{s}) \odot \sigma(W_{k,g} * \mathbf{x} + V_{k,g} * \mathbf{s})$$

1x1 convolution 1x1 convolution

Conditional image generation with pixelcnn decoders. NIPS 2016

Results

African elephant

Sandbar

Figure from: Oord et al.

Other Recent Improvements

- Gated PixelCNN (Oord et al.)
  - Improve PixelCNN by removing blind spots and replacing ReLU units
- PixelCNN++ (Salimans et al.)
  - Improve PixelCNN by optimization techniques
- Video Pixel Networks (Kalchbrenner et al.)
  - Extend the work to 4 dimension

Comparison with GANs and VAEs

Autoregressive models (PixelRNNs, PixelCNNs)	GAN	VAE
• Simple and stable training process (e.g. softmax loss) • Best log likelihoods so far	• Sharpest images	• Easy to relate image with low-dimensional latent variables
• Inefficient during sampling • Don't easily provide simple low-dimensional codes for images	• Difficult to optimize due to unstable training dynamics	• Tends to have blurry outputs
		

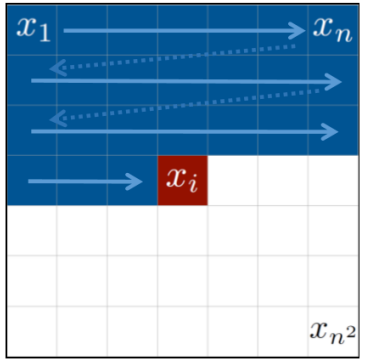
Credit: https://openai.com/blog/generative-models/, Oord et al. and Larsen et al.

Iterative Attentive Generation:

What We Saw Previously

Pixel-by-pixel generation:

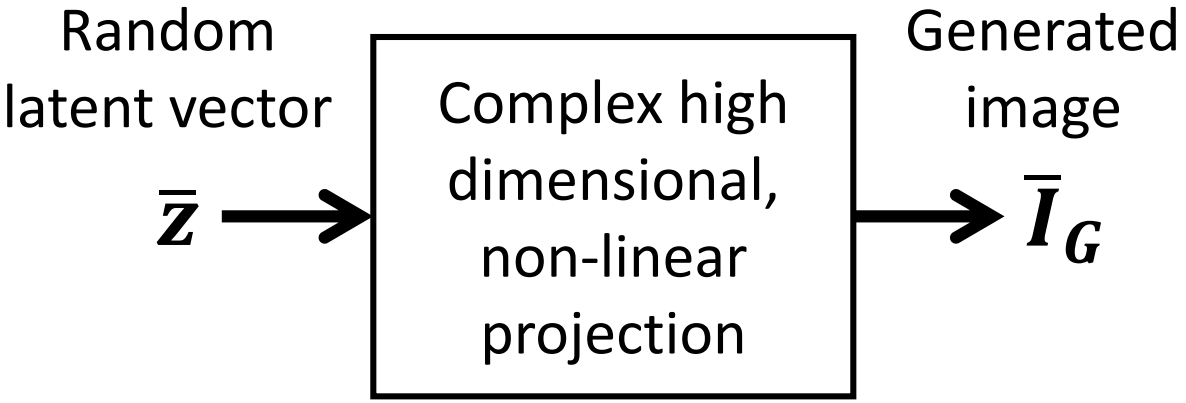
Inference decisions at the pixel-level



Generation using VAEs and GANs:

Employ one-shot generation

- Limited for highly complex scenes
- Alignments of objects should be considered only after objects are generated



The Human Way

Attention

Focus on a subpart at a time

Iteration

Use multiple strokes
Eraser

Feedback

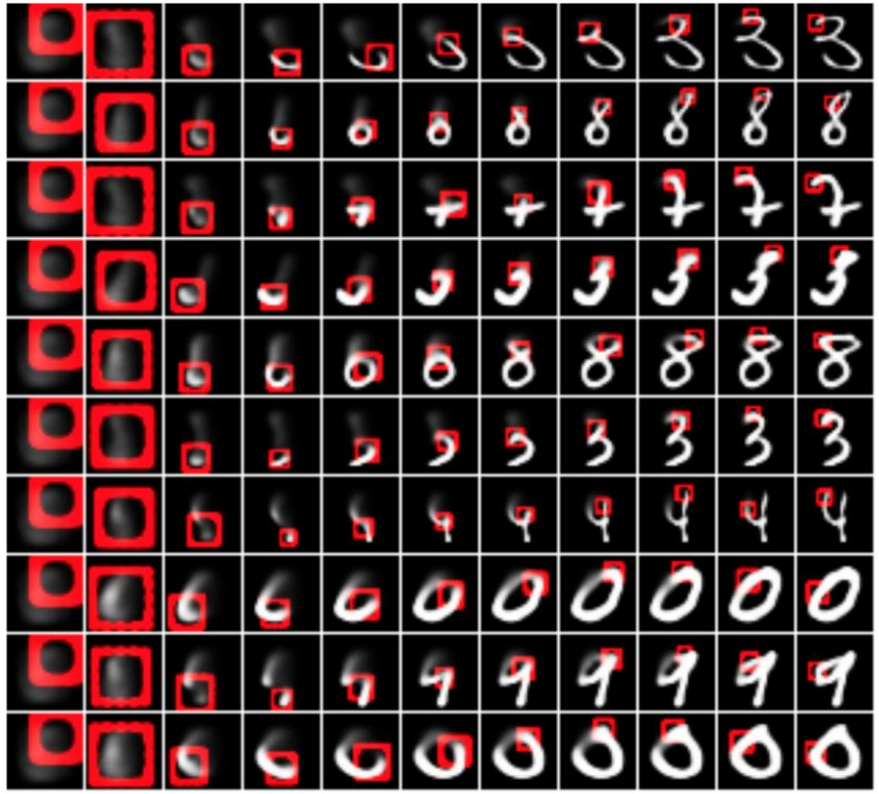
Draw, take a look and then see what to draw next

Enabling machines to employ above attributes
Iterative Attentive Generation using deep learning models

DRAW: A Recurrent Neural Network For Image Generation

May 2015

Karol Gregor KAROLG@GOOGLE.COM
Ivo Danihelka DANIHELKA@GOOGLE.COM
Alex Graves GRAVESA@GOOGLE.COM
Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Daan Wierstra WIERSTRA@GOOGLE.COM
Google DeepMind

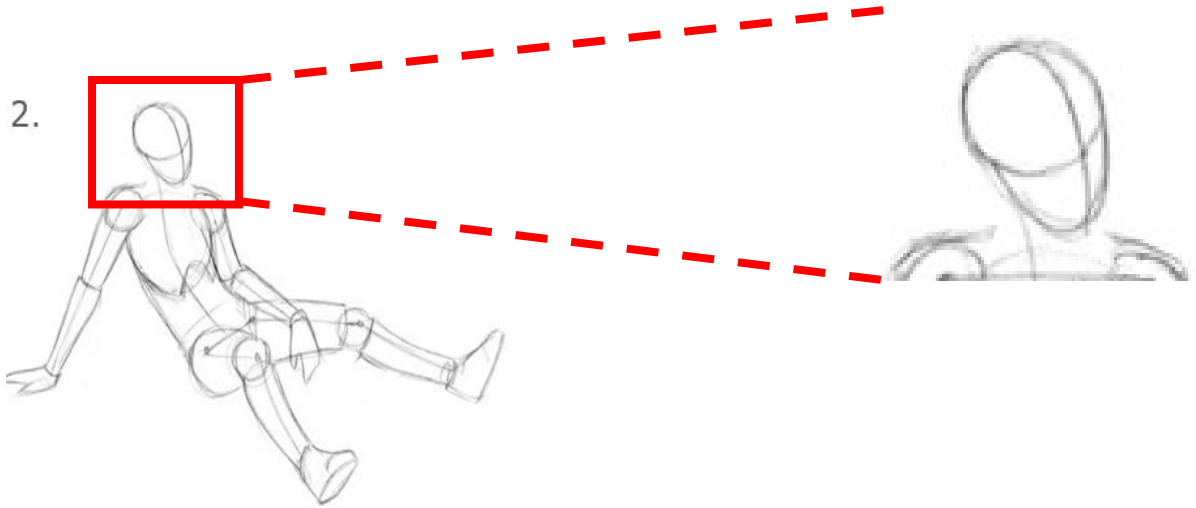


Time ⟶

How can a machine employ “Attention”?

What is attention ?

Ability to focus on a part of an image, either to understand it or to modify it



Original image
500×500

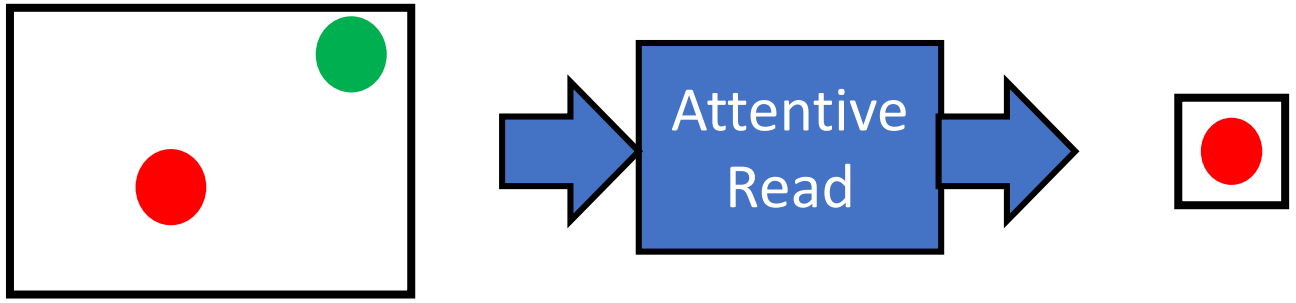
Glimpse
60×60

For machine, it is a process of “***Glimpse***” extraction

Employing Attention:

Attentive read: Reading a particular glimpse from an image

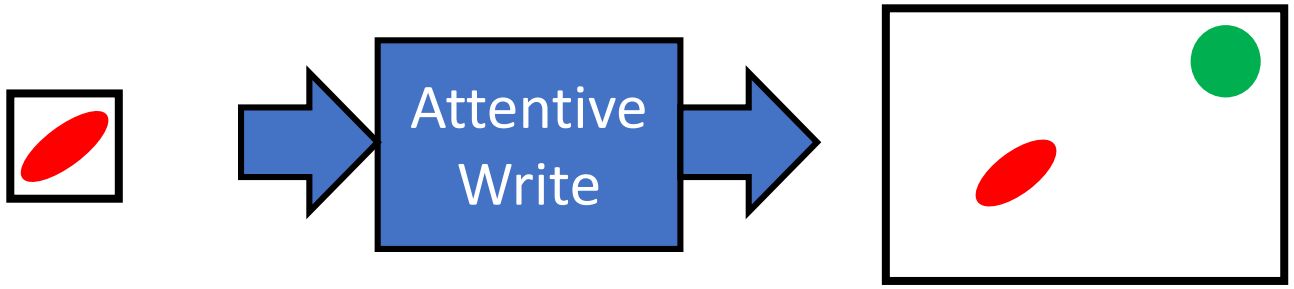
Input Image:
$A \times B$



Extracted glimpse:
$N \times N$

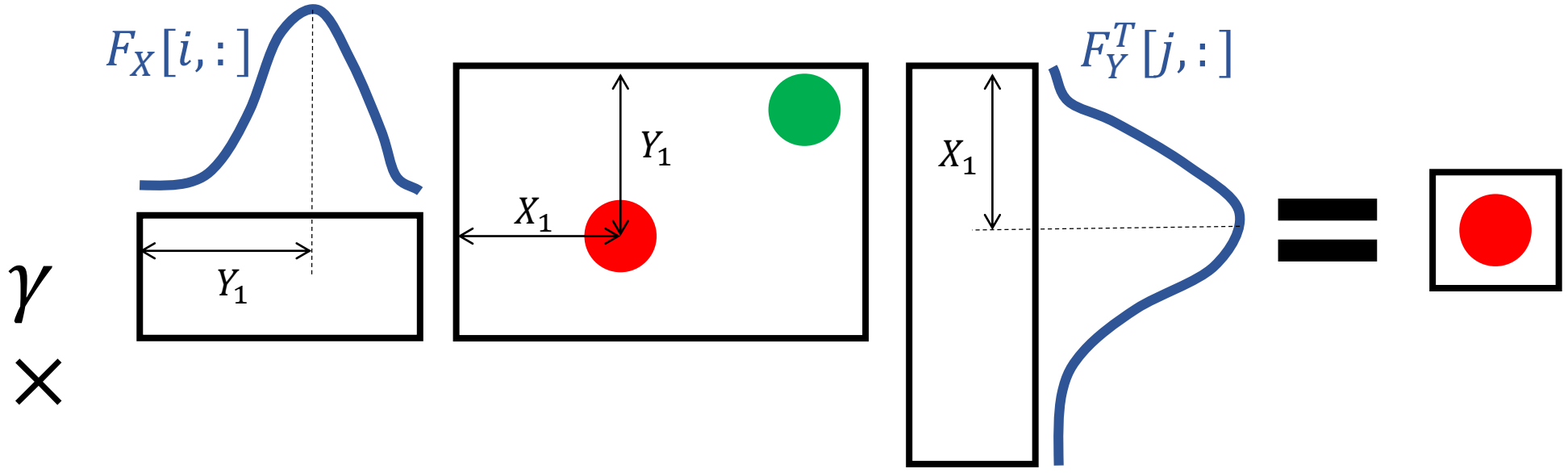
Attentive write: Writing the modified glimpse to the image

Modified glimpse:
$N \times N$



Updated image:
$A \times B$

Read Attention



Filter F_X: $N \times A$

Image: $A \times B$

Filter F_Y^T: $B \times N$

Glimpse: $N \times N$

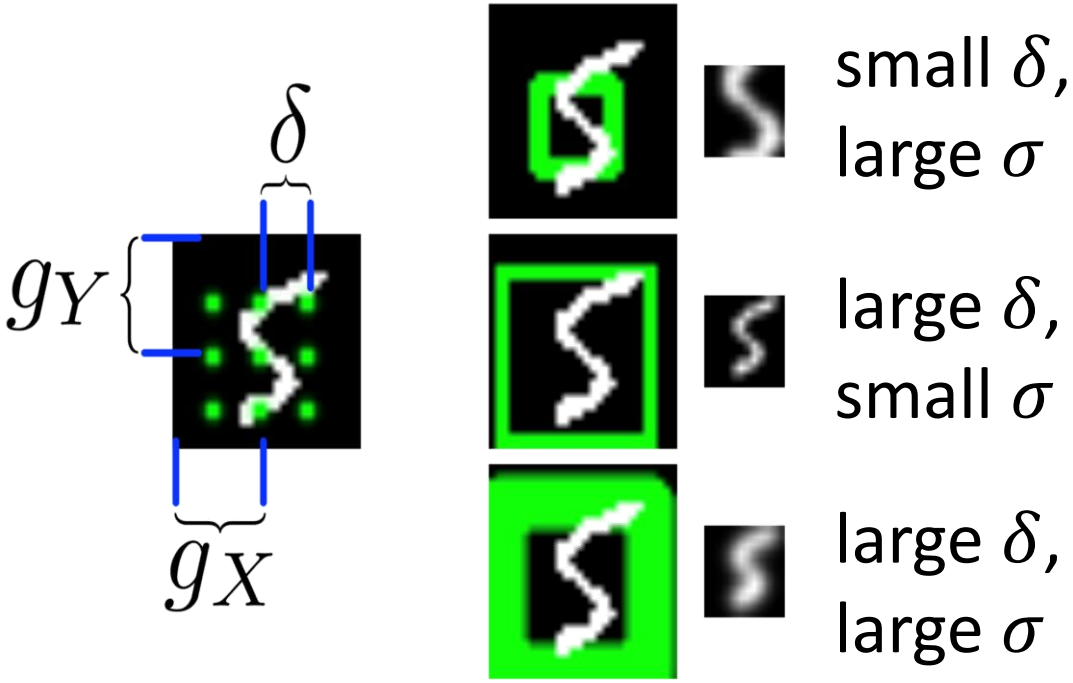
Filter definitions:

$$F_X[i,a] = \frac{1}{Z_X} \exp\left(-\frac{(a-\mu_X^i)^2}{2\sigma^2}\right)$$

$$F_Y[j,b] = \frac{1}{Z_Y} \exp\left(-\frac{(b-\mu_Y^j)^2}{2\sigma^2}\right)$$

Choosing filter parameters

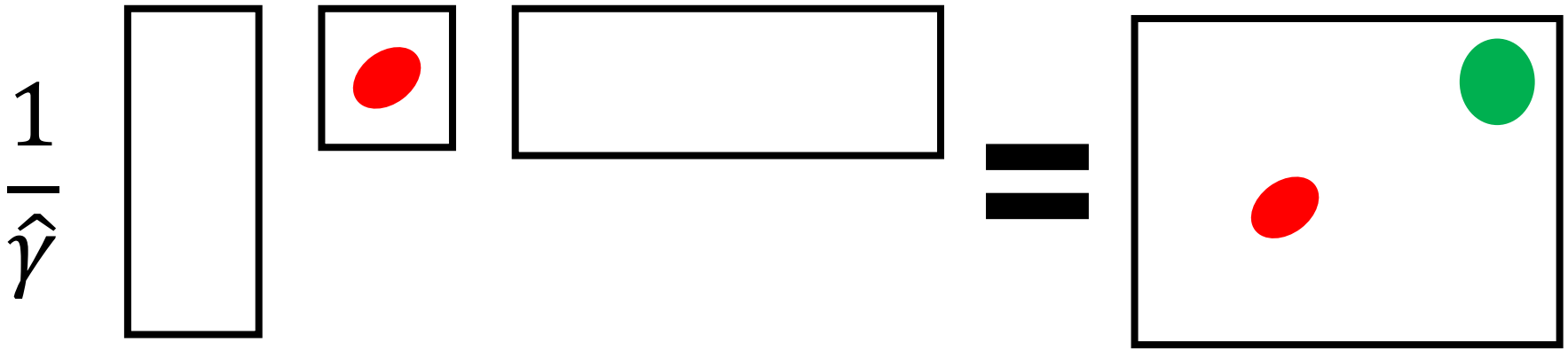
N=12



$$\mu_X^i = g_X + (i - N/2 - 0.5)\,\delta$$
$$\mu_Y^j = g_Y + (j - N/2 - 0.5)\,\delta$$

g_X, g_Y, γ, δ are learned. Details about it later….

Write Attention:



Filter Y^T: $A \times N$

Glimpse: $N \times N$

Filter X: $N \times B$

Updated image $A \times B$

Again, we use Gaussian filters in the exact same setting.
Write filters are distinct from read filters

Machines trying to generate images in a manner similar to us

Attention

Done!

Iteration

?

Feedback

?

How can machines employ “Iterations”?

Machines actually iterate all the time…!

But for image generation, you need something to iterate upon….

A canvas !!!

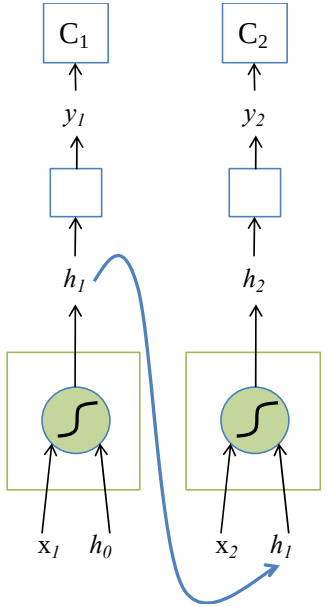
It is a matrix c, typically of the same size that of the original image

Write attention operation updates parts in c

Final generated image = $f(c)$

How can machines employ “Feedback”?

Remember RNNs ..!!

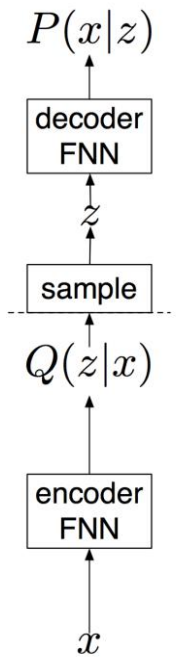


RNNs

(Figure courtesy: Arun Mallya)

So, why don’t combine these two..!

And we want to generate images..!



Variational Autoencoders

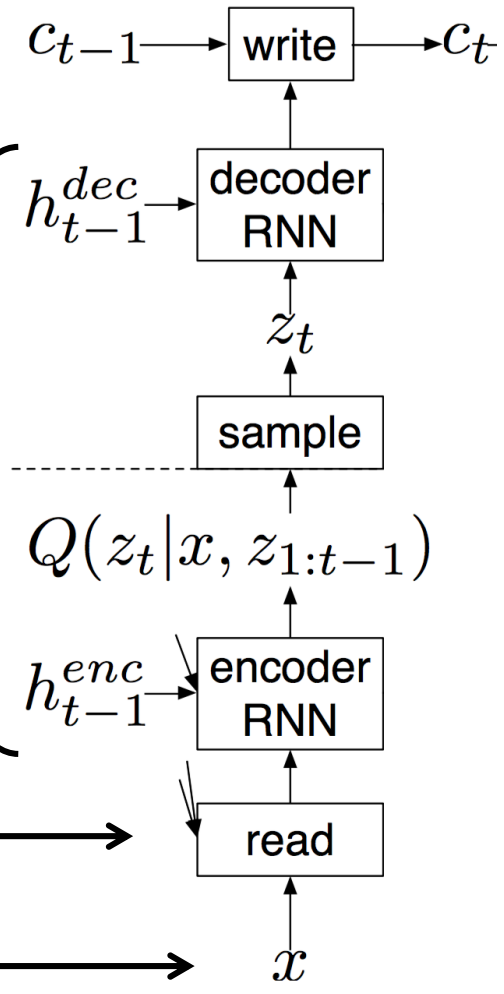
Putting it all together: Ta Da…!!

Attentive write and canvas update:

Conventional Variational autoencoder (operating on ***glimpse***)

Attentive read:

Input image

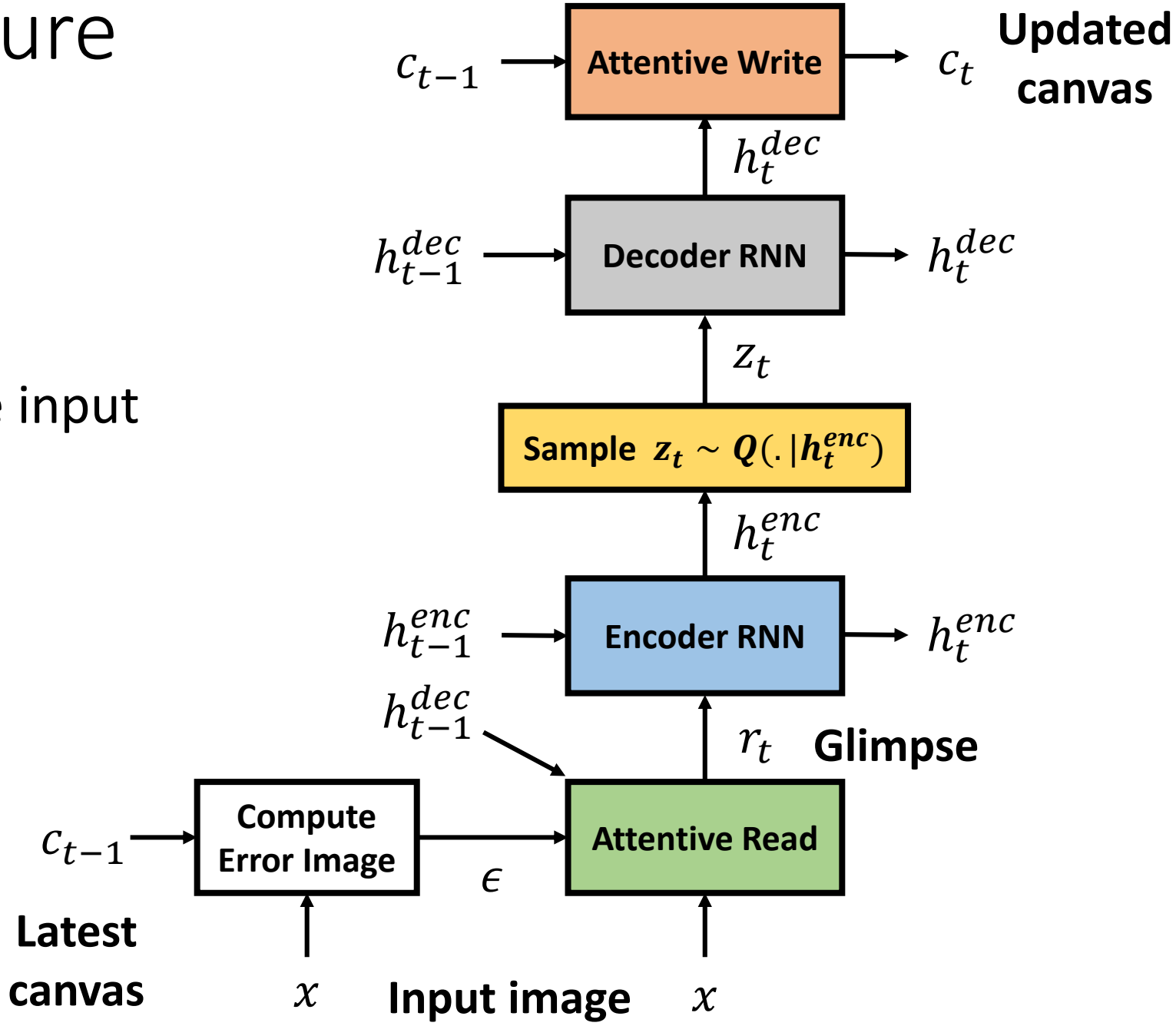


RNN-based feedback and loop:

Training Procedure

Goal in training:
Learn to reconstruct the input image x in T iteration…

So, for every iteration t:



Training Procedure

Loss function:

$$\underbrace{-\log D(x|c_T)}_{\text{Reconstruction loss}} + \underbrace{\sum_{t=1}^{T} KL\big(Q(Z_t|h_t^{enc})||P(Z_t)\big)}_{\text{Regularization (for each iteration)}}$$

Recall: VAE lecture

Typically, $P(Z_t) \sim N(0, \mathrm{I})$ and $Q(Z_t|h_t^{enc}) \sim N\big(\mu_t(h_t^{enc}), \sigma_t^2(h_t^{enc})\big)$

Training Procedure

Given input image, compute the feedforward path for T iterations

↓

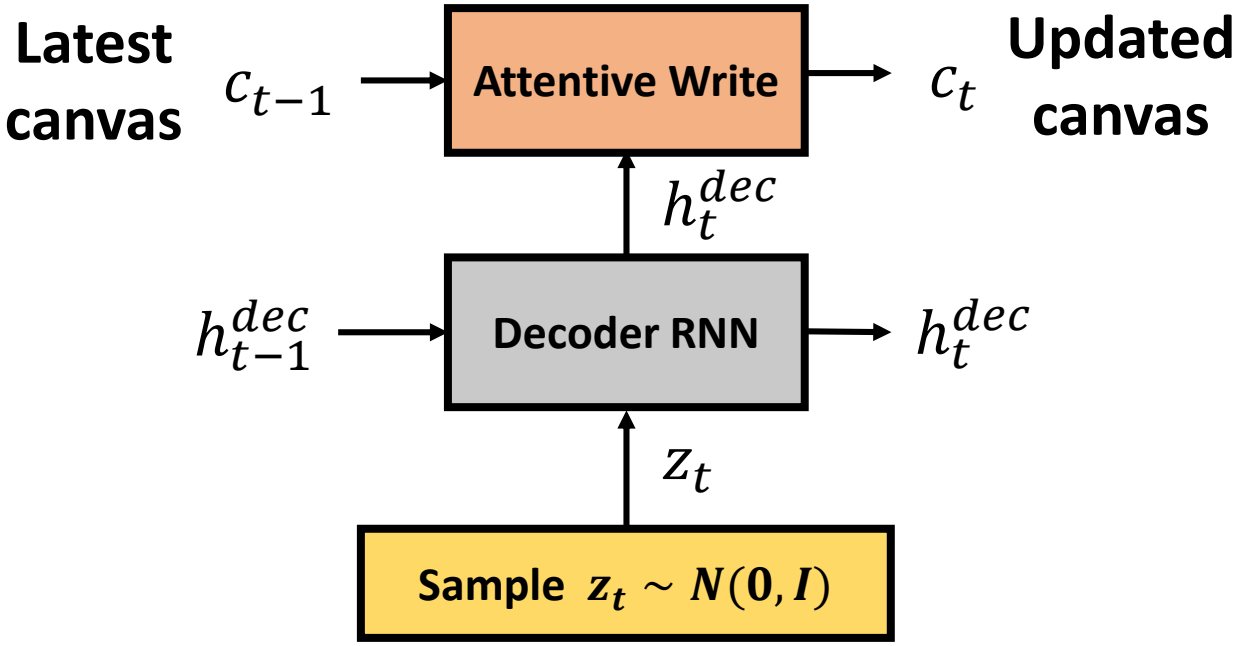
Compute the final loss function

↓

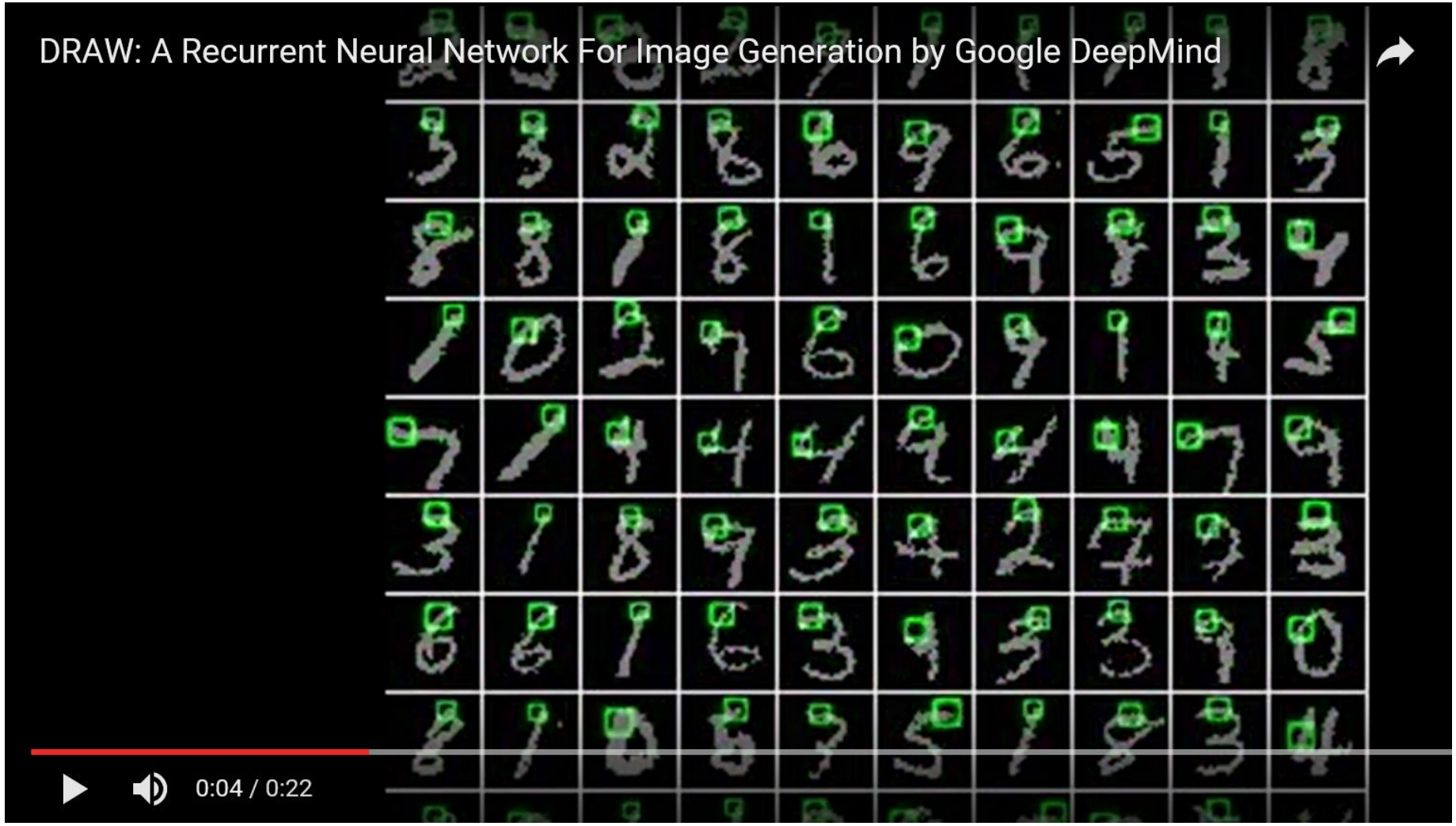
Compute the gradients and propagate them back
(recall: reparametrization trick in VAEs)

Image Generation During Test:

No encoder is used while generating images, just like VAEs

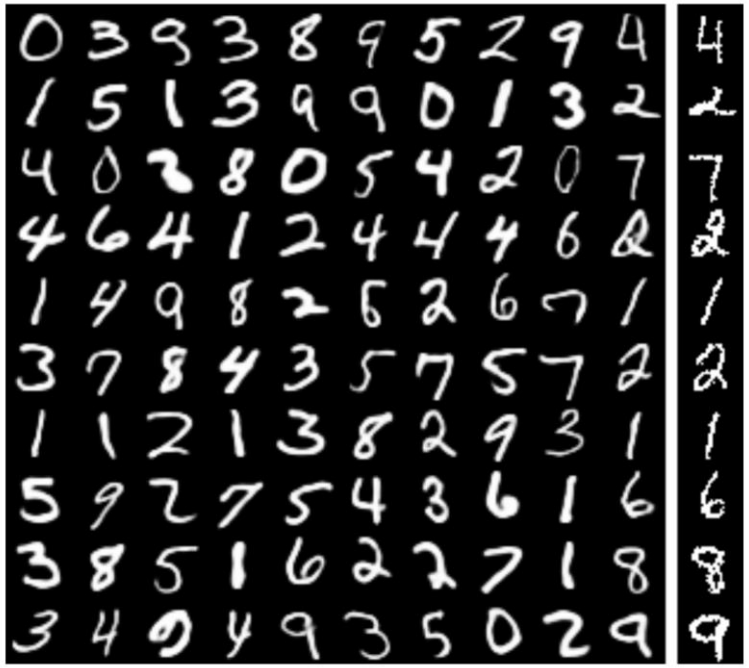


Let’s See How it Performs:



Final Results: Generated images

Task	#glimpses	LSTM #h	#z	Read Size	Write Size
MNIST Model	64	256	100	2×2	5×5
SVHN Model	32	800	100	12×12	12×12
CIFAR Model	64	400	200	5×5	5×5

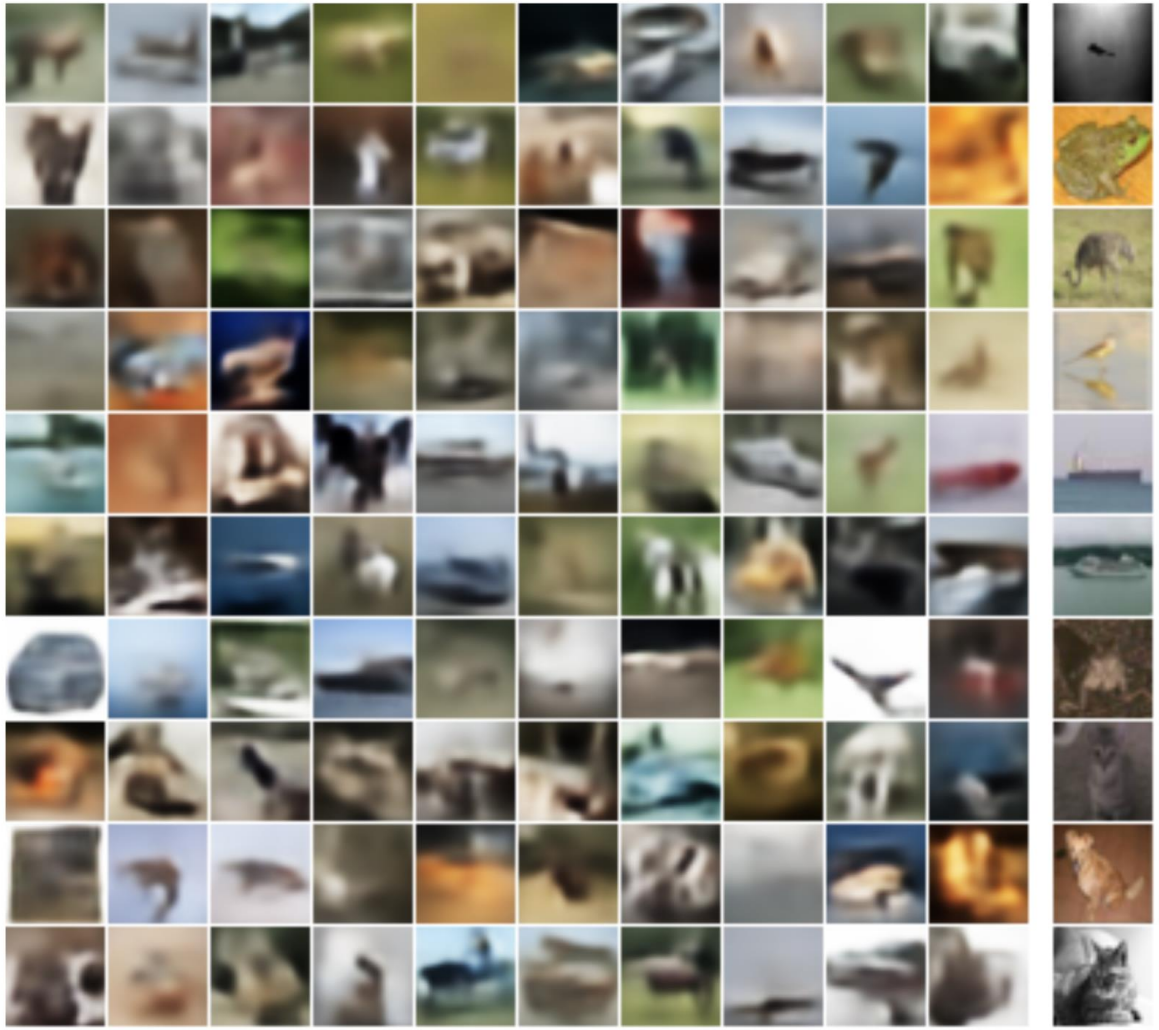


MNIST

SVHN

[DRAW: a recurrent neural network for image generation](). ICML 2015

CFAR-10 Generation



The images still seem somewhat blur

DRAW: a recurrent neural network for image generation. ICML 2015

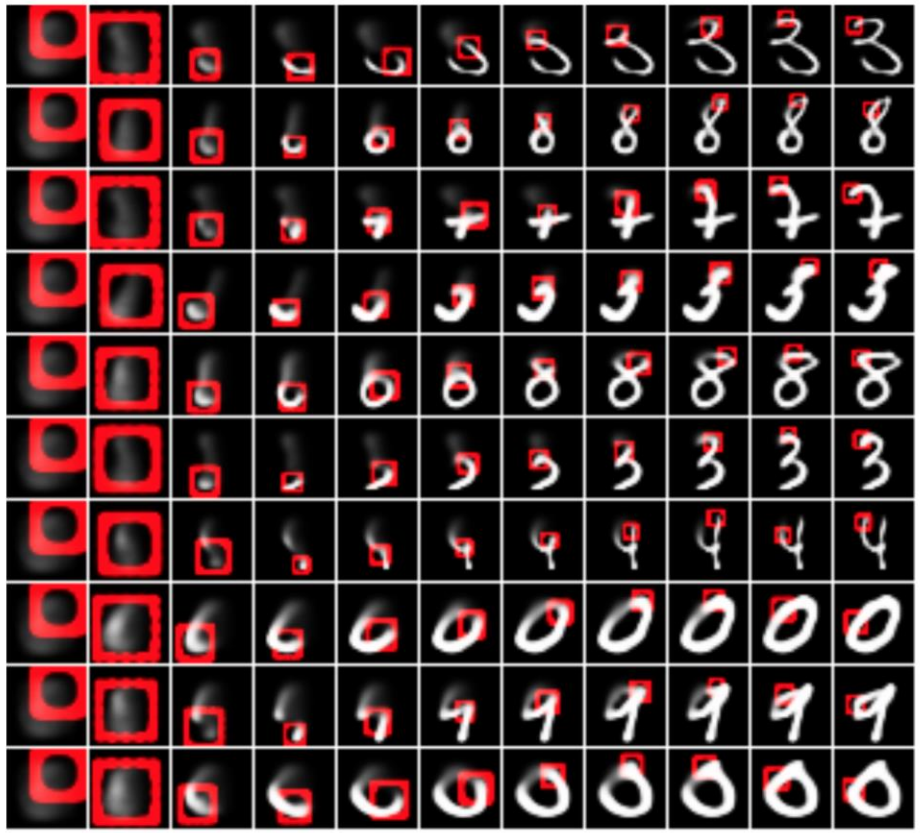
Final Results:

The red square indicates “glimpse” used for attentive write in canvas

Without attention

Time ⟶

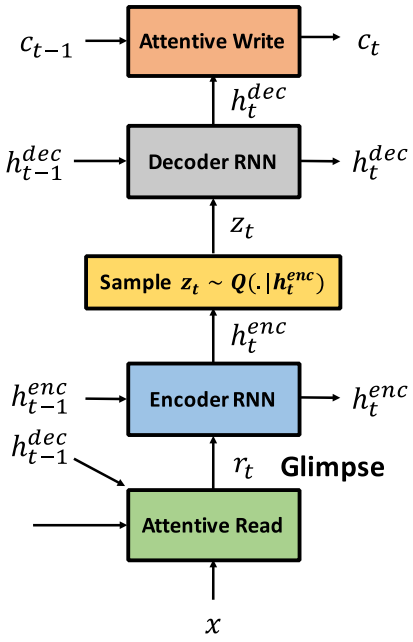
With attention



Time ⟶

[DRAW: a recurrent neural network for image generation](). ICML 2015

So, now we have a machine that can employ "***attention***", "***iteration***" and "***feedback***"



(Lossy) Image Compression

Improve it and use it for

One Shot Generalization

Image Generation to Compression:

How humans generate images ..!

Some bipedal entity

A human-like being

A young boy sitting, has some hair, wearing shirt, shoes…..

Final exact image:

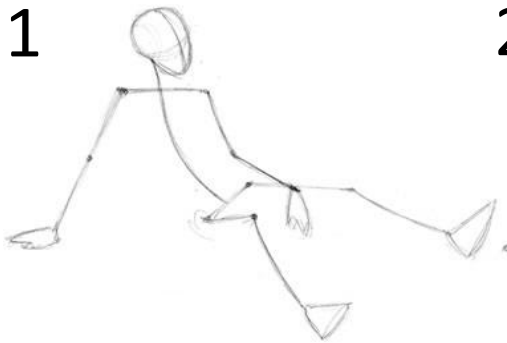
Conceptual hierarchy

A way to “conceptually” compress images

“Conceptual” Lossy Compression

Given 4, can a machine infer 1, 2 and 3 ?

1



2

3

4

1 can also mean:

2 can also mean:

3 can also mean:

Actual Image:

Ultimate (lossy) compression: Store only ***absolutely essential information (concept)***. Let the deep generative model generate ***the details***…

Image credits: https://s-media-cache-ak0.pinimg.com/236x/86/f2/12/86f212a460b81fb5d12fe3d7460aa0e0.jpg, https://goo.gl/images/afnkkg

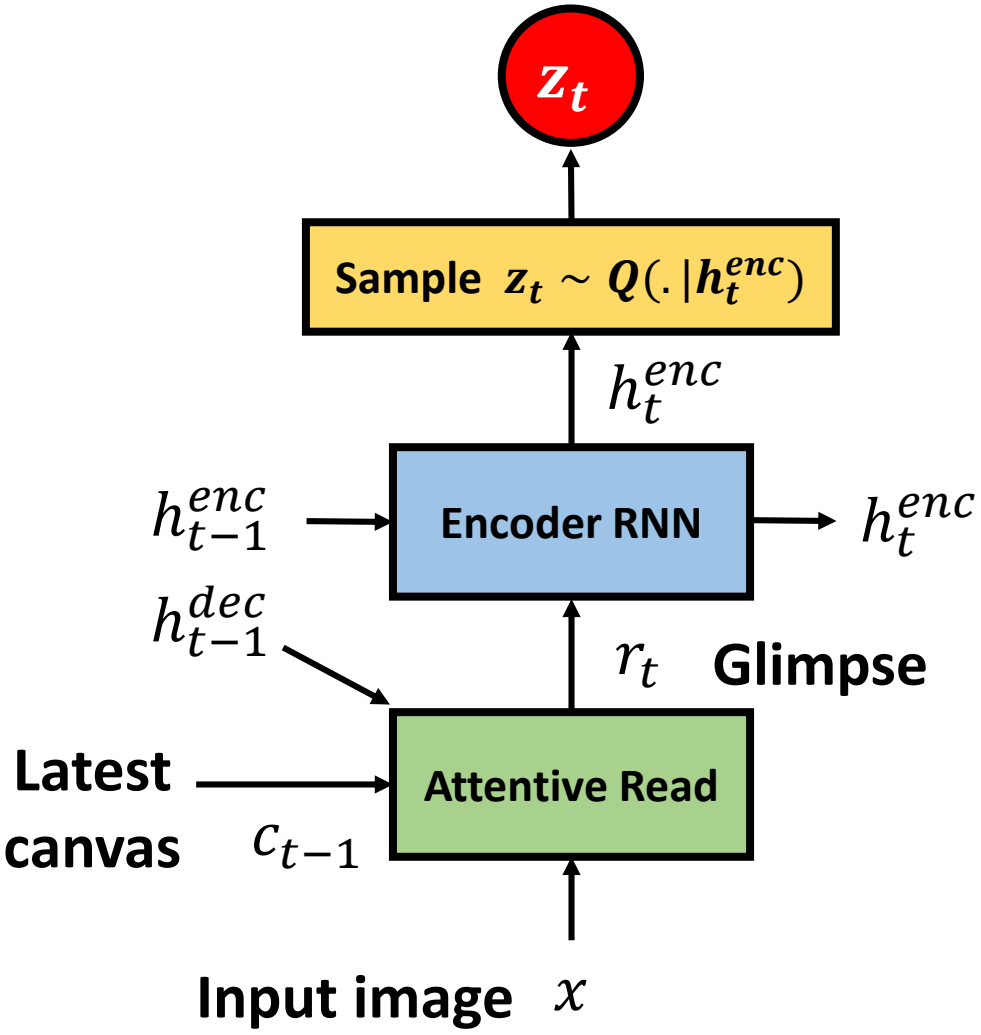
Towards Conceptual Compression

Karol Gregor KAROLG@GOOGLE.COM
Frederic Besse FBESSE@GOOGLE.COM
Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Ivo Danihelka DANIHELKA@GOOGLE.COM
Daan Wierstra WIERSTRA@GOOGLE.COM
Google DeepMind, London, United Kingdom

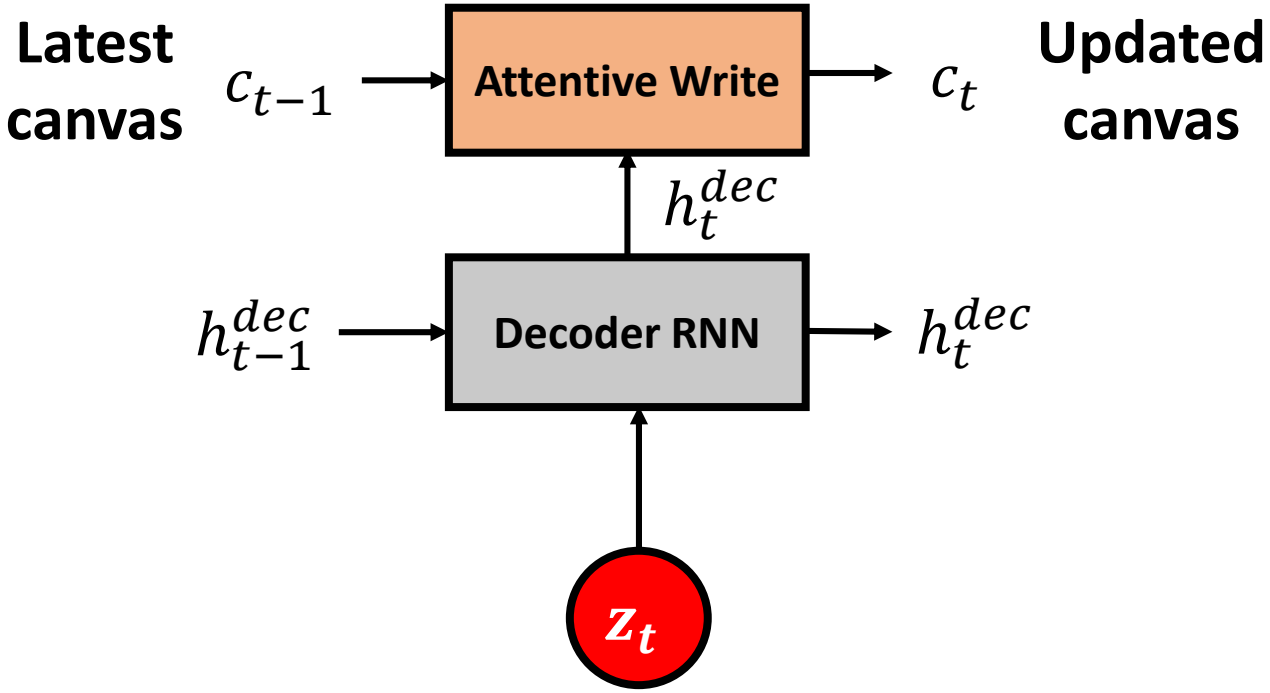
April 2015

Compression

Deep Generative Model: Recall



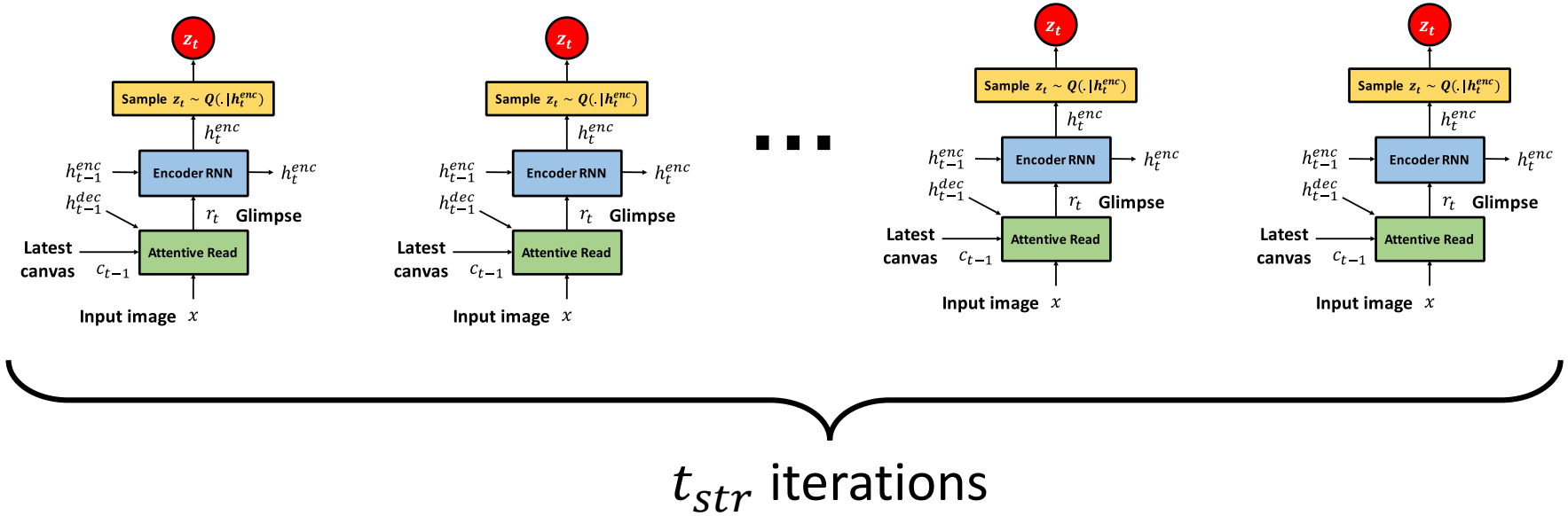
Inference Model



Generative Model

Compression: Basic Idea

- Train a deep generative model for T iterations
- Given a new image x, run the model for first $t_{str} < T$ iterations

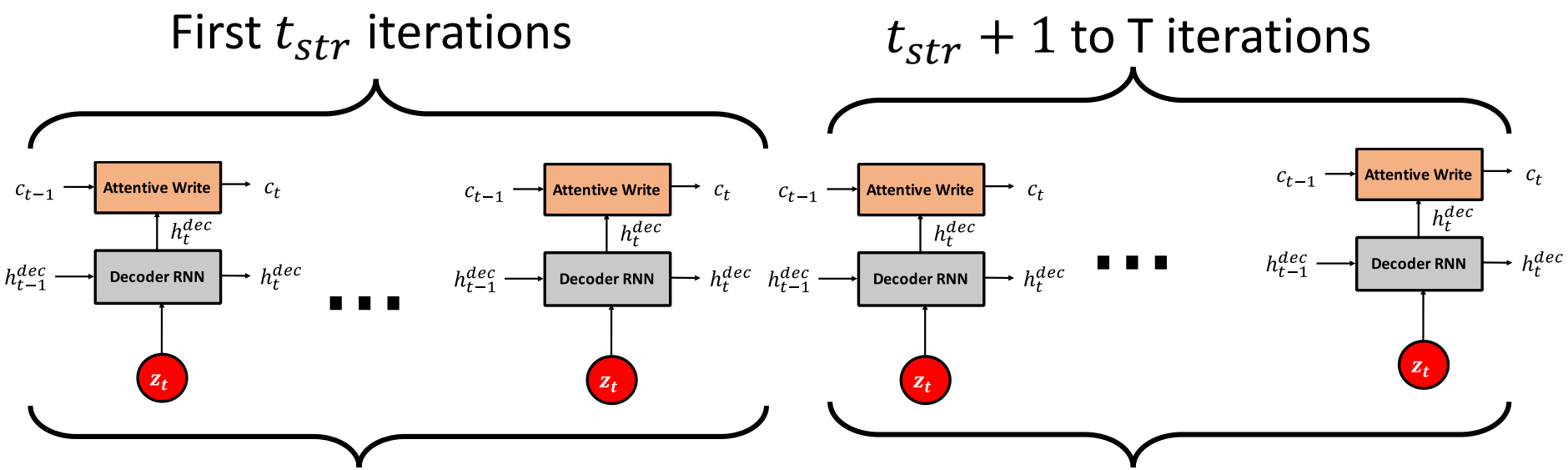


Store the means of inferred posterior latent distributions: $z_1, \ldots, z_{t_{str}}$

These $z_1, \ldots, z_{t_{str}}$ represent the compressed image

Reconstruction from Compressed Image:

- Start with $z_1, \ldots, z_{t_{str}}$
- Run first $t_{str} < T$ iterations of the generative model "deterministically"
- Run $t_{str} + 1$ to T iterations while sampling those latent variables from learned prior $p\left(z_t \middle| h_{t-1}^d\right)$



Final reconstructed image

z_ts are stored

$z_t \sim p(z|h_{t-1}^d)$

Note: The prior here depends upon h_{t-1}^d

Some More Details:

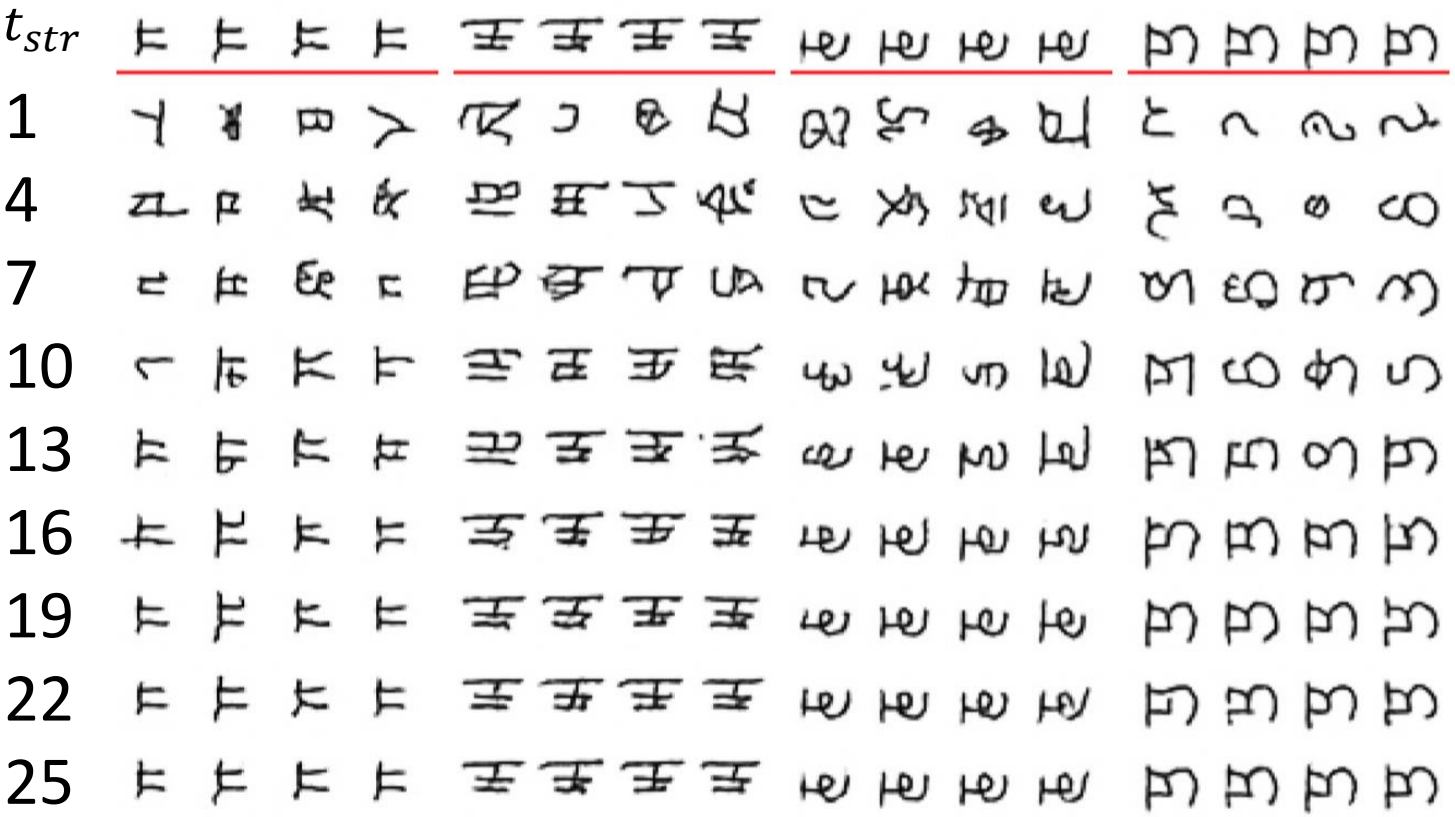
- Training procedure and training loss are same as we saw in the case of DRAW work

$$L = \beta L^x + \sum_{t=1}^{T} L_t^z$$

Parameter β controls emphasis on "pixel-level reconstruction"

- The latent variables z_1 to $z_{t_{str}}$ can be further compressed by discretization and arithmetic coding etc.

Reconstruction from “conceptually compressed”



Reconstructed with prior variance (WPV)

$$z_t \sim p(z|h_{t-1}^d)$$

Towards conceptual compression. NIPS 2016

Results: Lossy Compression

JPEG

JPEG2000

conv-DRAW, WPV

conv-DRAW, WoPV

bits per image: 153

JPEG

JPEG2000

conv-DRAW, WPV

conv-DRAW, WoPV

bits per image: 307

Towards conceptual compression. NIPS 2016

Results: Lossy Compression

JPEG

JPEG2000

conv-DRAW, WPV

conv-DRAW, WoPV

bits per image: 460

When latent variables are sampled from prior distribution, the reconstructed images are sharper but have artifacts

Towards conceptual compression. NIPS 2016

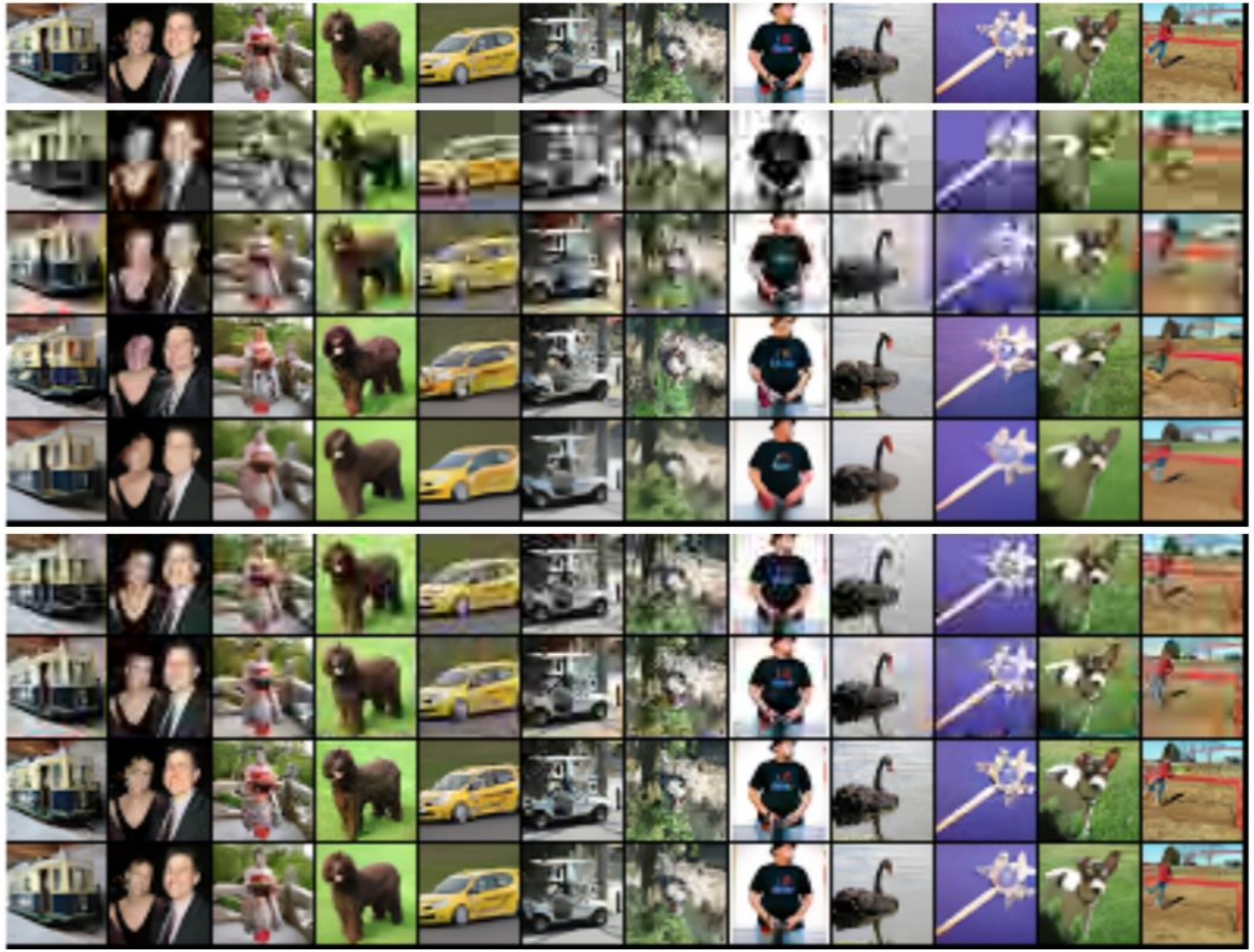
Results: Lossy Compression

JPEG

JPEG2000

conv-DRAW, WPV

conv-DRAW, WoPV



bits per image: 612

JPEG

JPEG2000

conv-DRAW, WPV

conv-DRAW, WoPV

bits per image: 1228

Towards conceptual compression. NIPS 2016

Results: Image Generation

β: weighing parameter for reconstruction loss

$\beta = 0.4$

$\beta = 0.5$

$\beta = 0.6$

$\beta = 0.8$

$\beta = 1$

blur

Sharper

Lower the β, lesser the emphasis on pixel-level details, and more on learning the latent representative structure

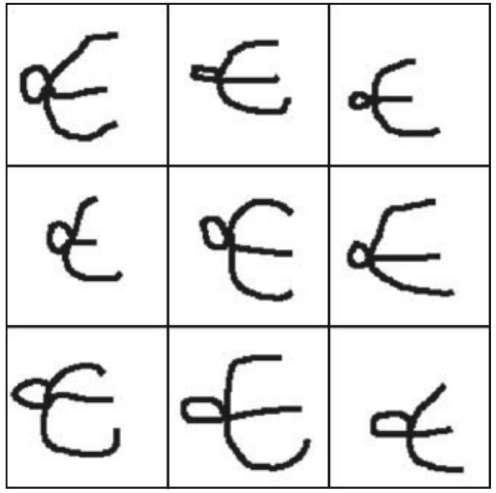
Towards conceptual compression. NIPS 2016

One-shot Generalization

Train your machine to obtain the final parameters…!!

Now, show a new example, that is **not a part of training set**

Then, can the machines generate following examples... ?



Humans have this ability of one-shot generalization

Image credits: Lake et al., science 2015

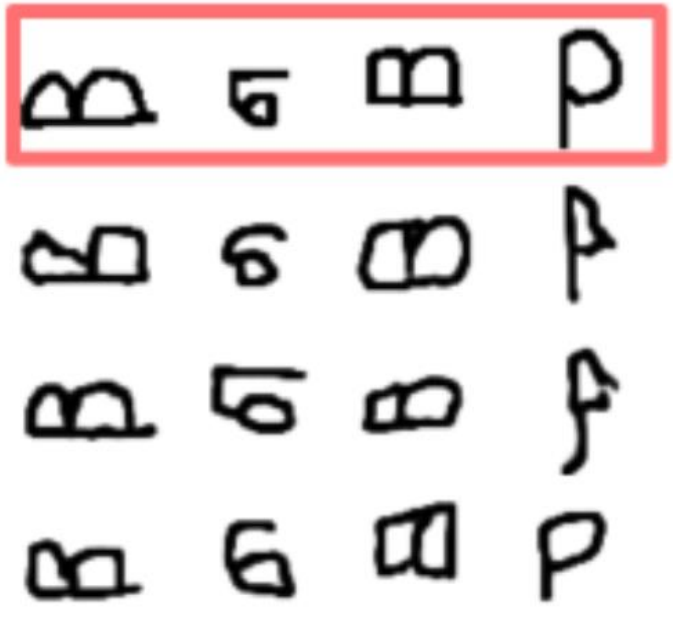
One-Shot Generalization in Deep Generative Models

Danilo J. Rezende* DANILOR@GOOGLE.COM
Shakir Mohamed* SHAKIR@GOOGLE.COM
Ivo Danihelka DANIHELKA@GOOGLE.COM
Karol Gregor KAROLG@GOOGLE.COM
Daan Wierstra WIERSTRA@GOOGLE.COM
Google DeepMind, London

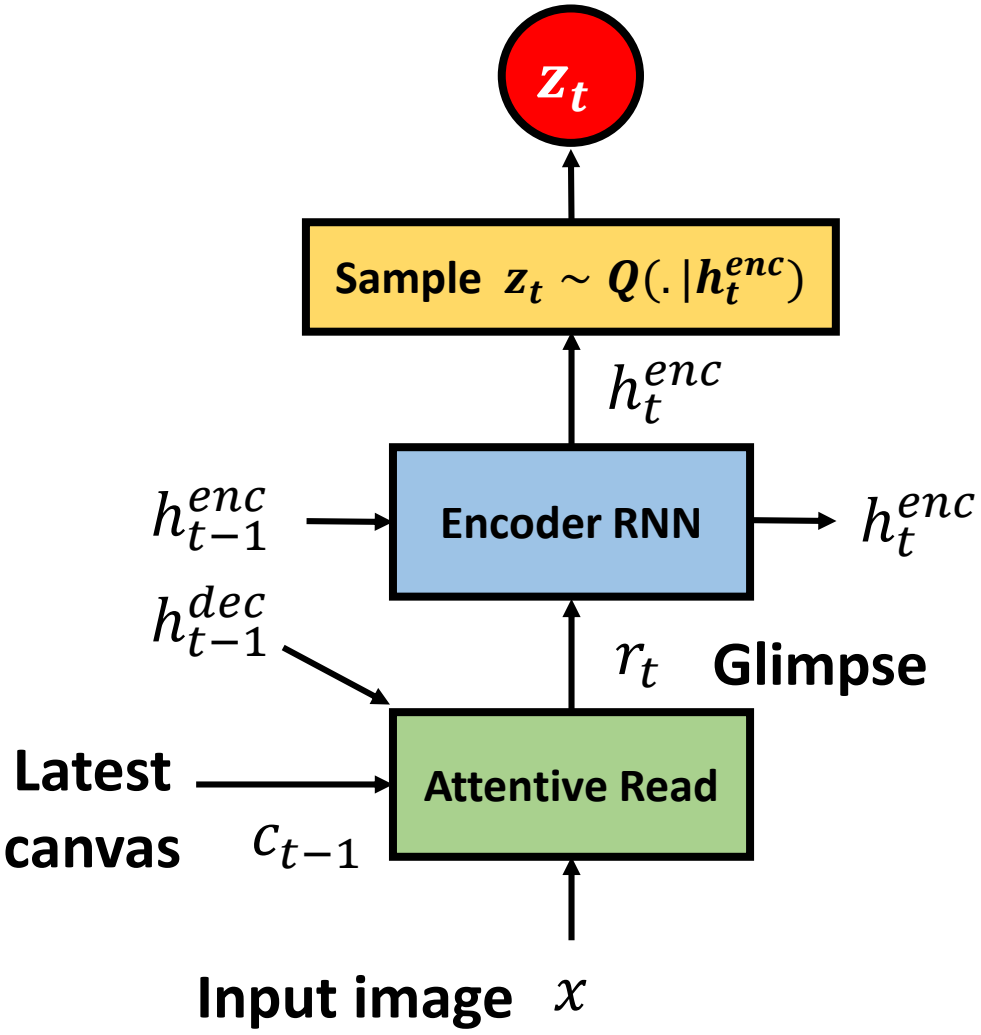
May 2016

Given one new example

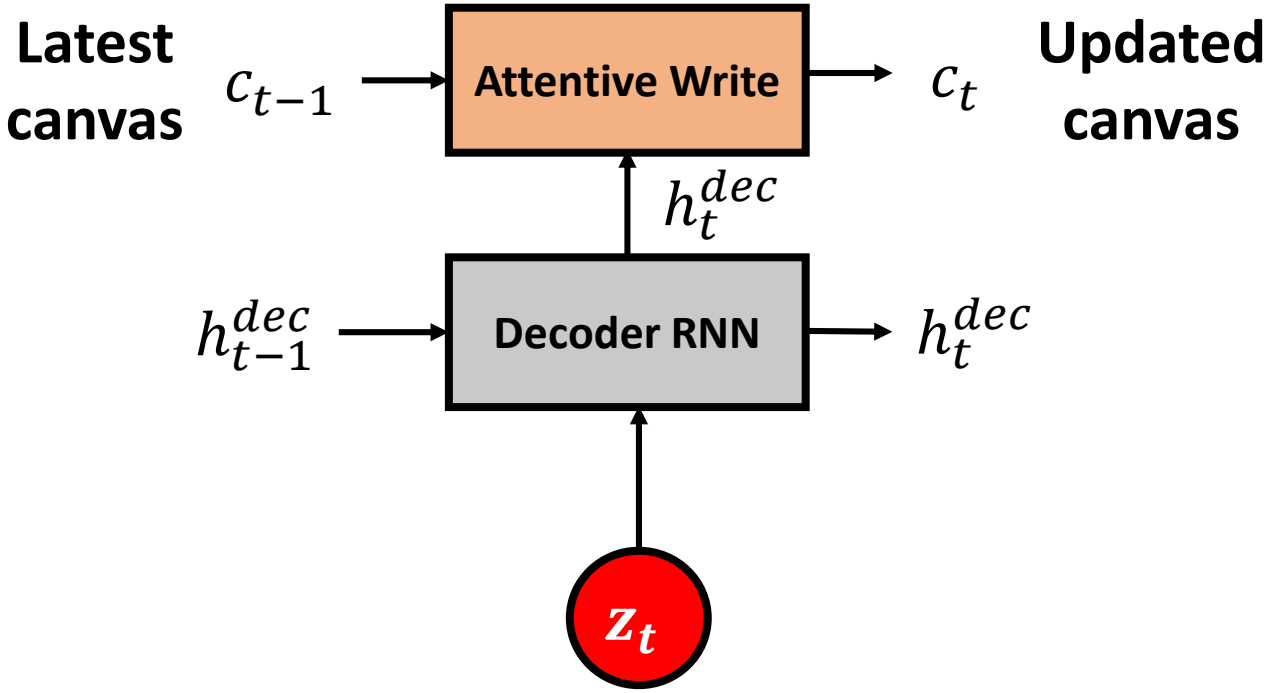
Examples generated by the model



Deep Generative Model: Recall

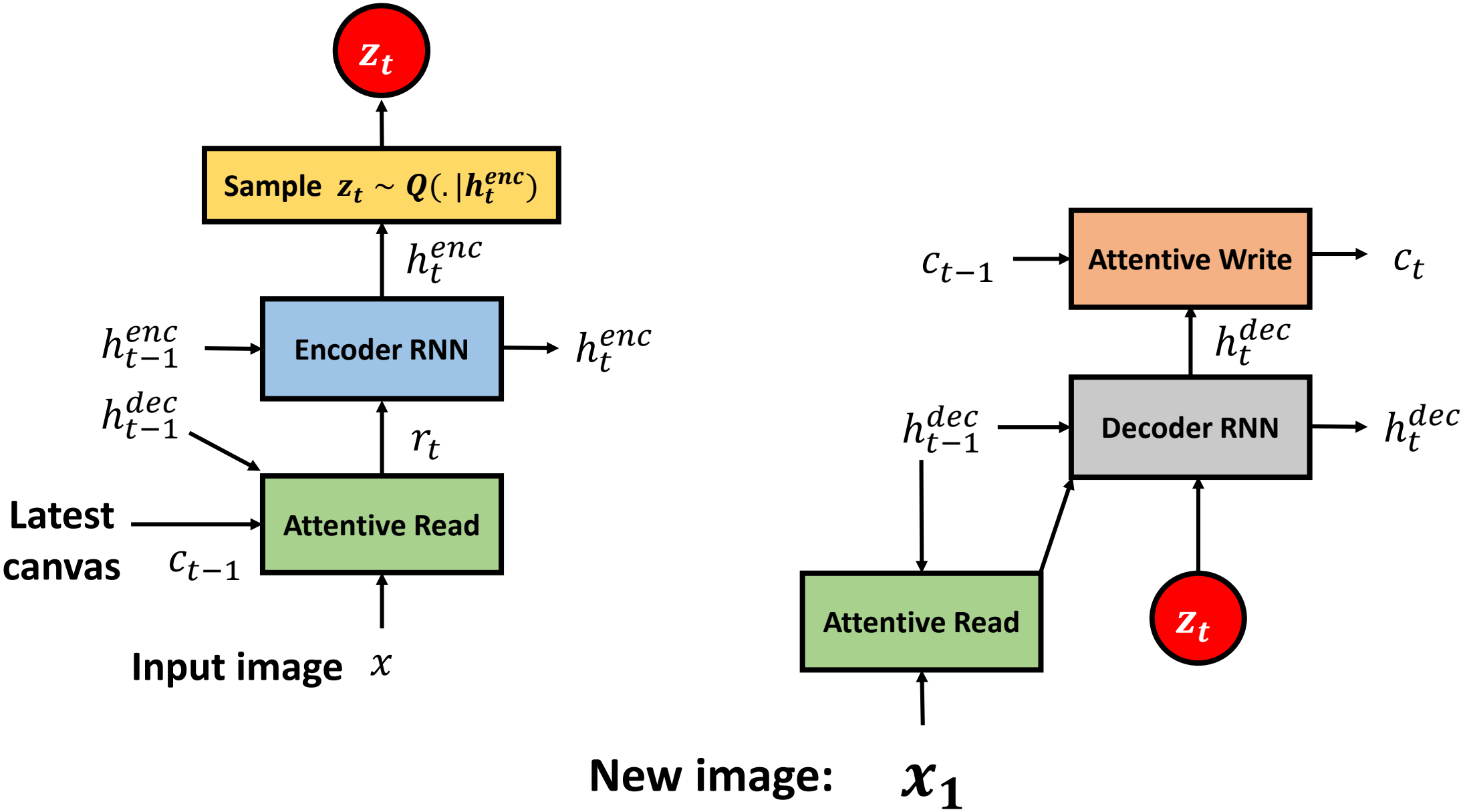


Inference Model



Generative Model

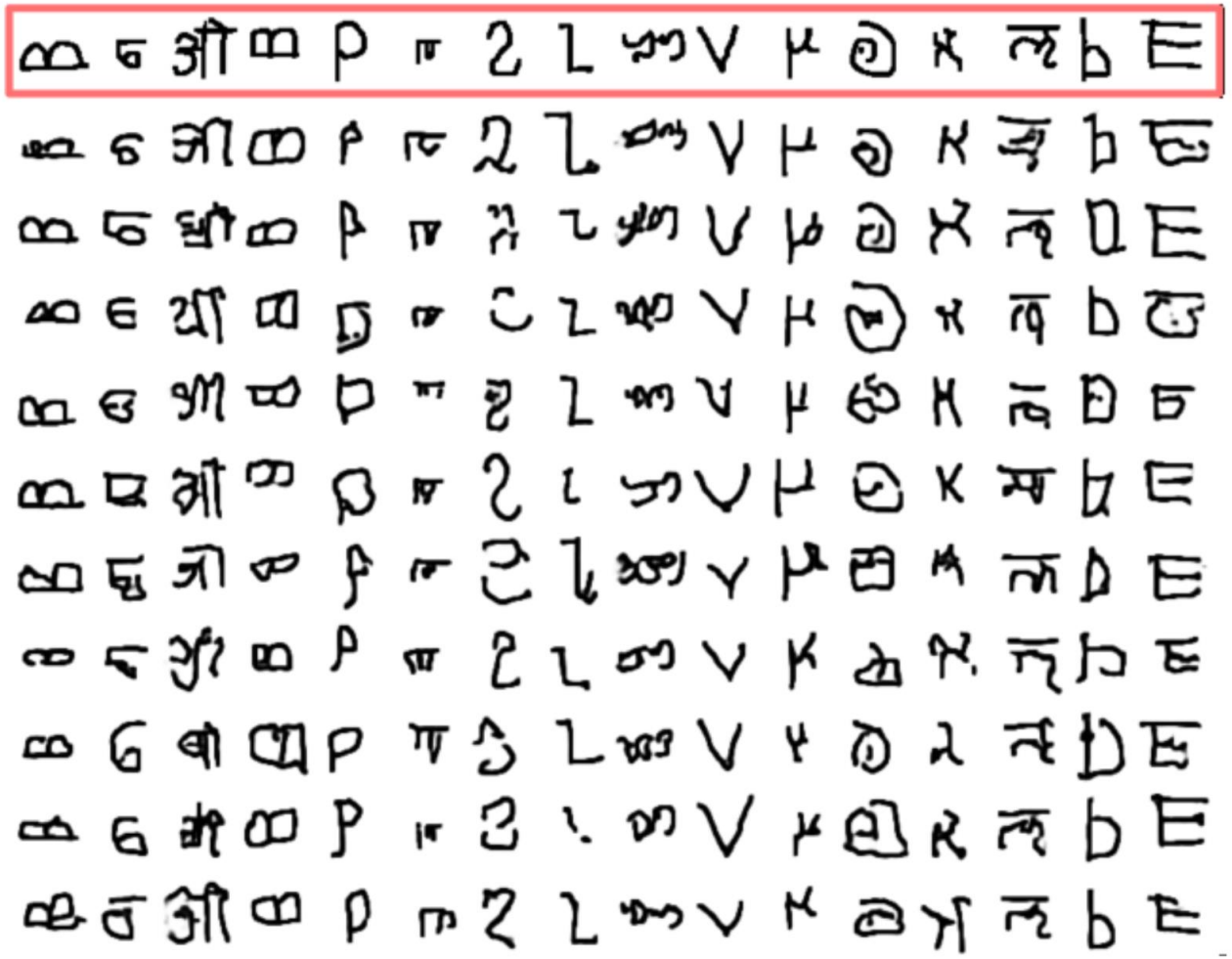
Generative Model for One-shot Generalization



One-Shot Generalization: Results

Given new example:

Examples generated by the network:



In this work, they used spatial transformer networks for attentive read and write.

More details at: https://arxiv.org/abs/1506.02025

Summary:

- Deep learning tools can be put together in many different innovative ways to obtain interesting results for different applications
  - PixelRNN, Deep generative model for conceptual compression

- We essentially studied extensions of “one-shot” image generation techniques to include some feedback in them using RNNs
  - Such extension is possible for GAN-based generation as well

Topics We Didn’t Cover

- Image Style Transfer: Take style of one image and draw the contents of other image in that style (Heard of Prisma App?)
- L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. CVPR 2016
- J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. ECCV 2016
- C. Castillo, S. De, X. Han, B. Singh, A. K. Yadav, and T. Goldstein. Son of Zorn's Lemma: Targeted Style Transfer Using Instance-aware Semantic Segmentation. ICASSP 2017

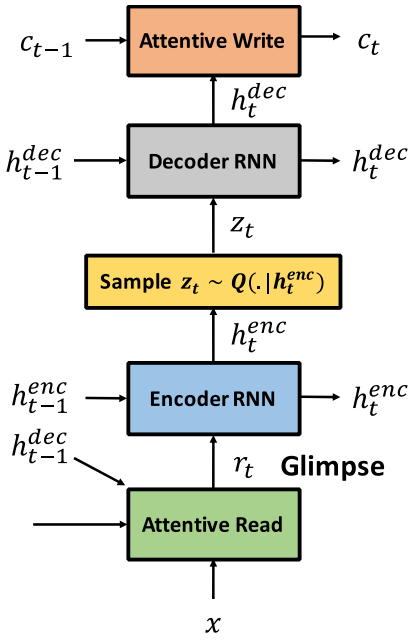
- Combining GANs and RNNs to form Generative: DRAW paper combined VAEs and RNNs. Similar combination of GANs and RNNs can be achieved (See backup slides)

Reading List

- A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. ICML 2016
- A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. NIPS 2016
- N. Kalchbrenner, A. van den Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. arXiv 2016
- K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra. DRAW: a recurrent neural network for image generation. ICML 2015
- K. Gregor, F. Besse, D. Rezende, I. Danihelka, and D. Wierstra. Towards conceptual compression. NIPS 2016
- B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. Science 2015
- D. J. Rezende, S. Mohamed, I. Danihelka, K. Gregor, and D. Wierstra. One-shot generalization in deep generative models. ICML 2016
- L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. CVPR 2016
- J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. ECCV 2016
- C. Castillo, S. De, X. Han, B. Singh, A. K. Yadav, and T. Goldstein. Son of Zorn's Lemma: Targeted Style Transfer Using Instance-aware Semantic Segmentation. ICASSP 2017

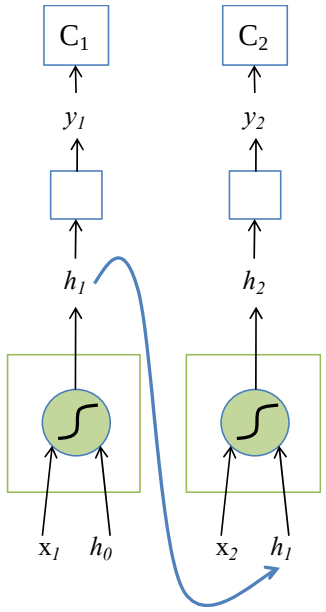
Backup Slides

What We Achieved



Deep Generative model that we studied

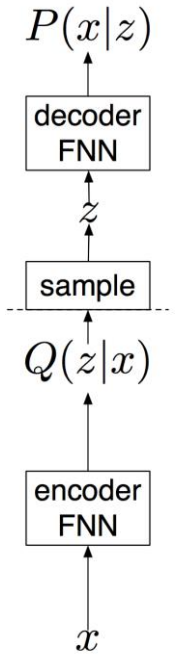
=



RNNs

(Figure courtesy: Arun Mallya)

+



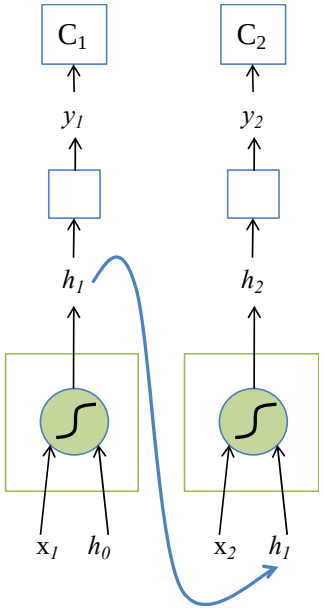
Variational Autoencoders

But, GANs generate sharper images

Are You Wondering ??

?

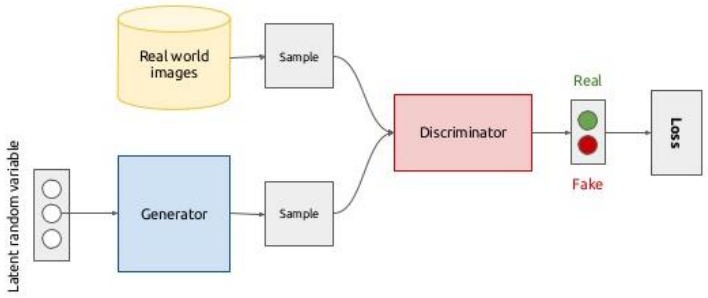
=



RNNs

(Figure courtesy: Arun Mallya)

+



GANs

(Figure courtesy: TUM wiki system)

Generating images with recurrent adversarial networks

Daniel Jiwoong Im[1]
Montreal Institute for Learning Algorithms
University of Montreal
imdaniel@iro.umontreal.ca

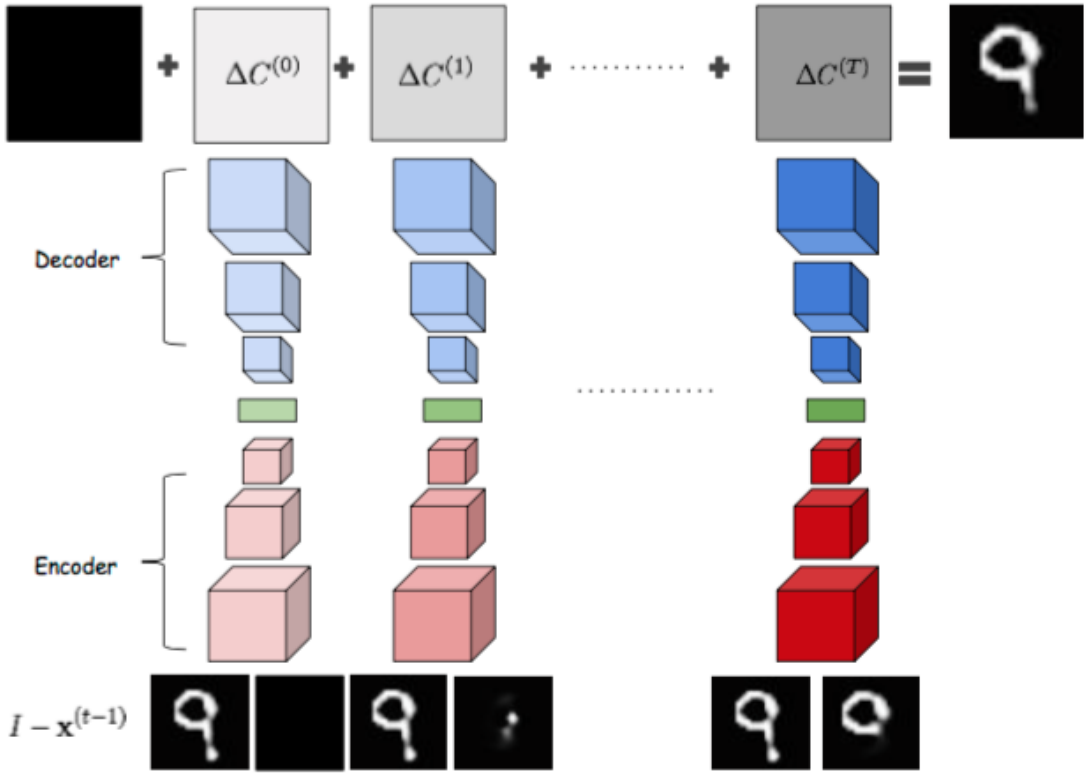
Chris Dongjoo Kim
Department of Engineering and Computer Science
York University
kimdon20@gmail.com

Hui Jiang
Department of Engineering and Computer Science
York University
hj@cse.yorku.ca

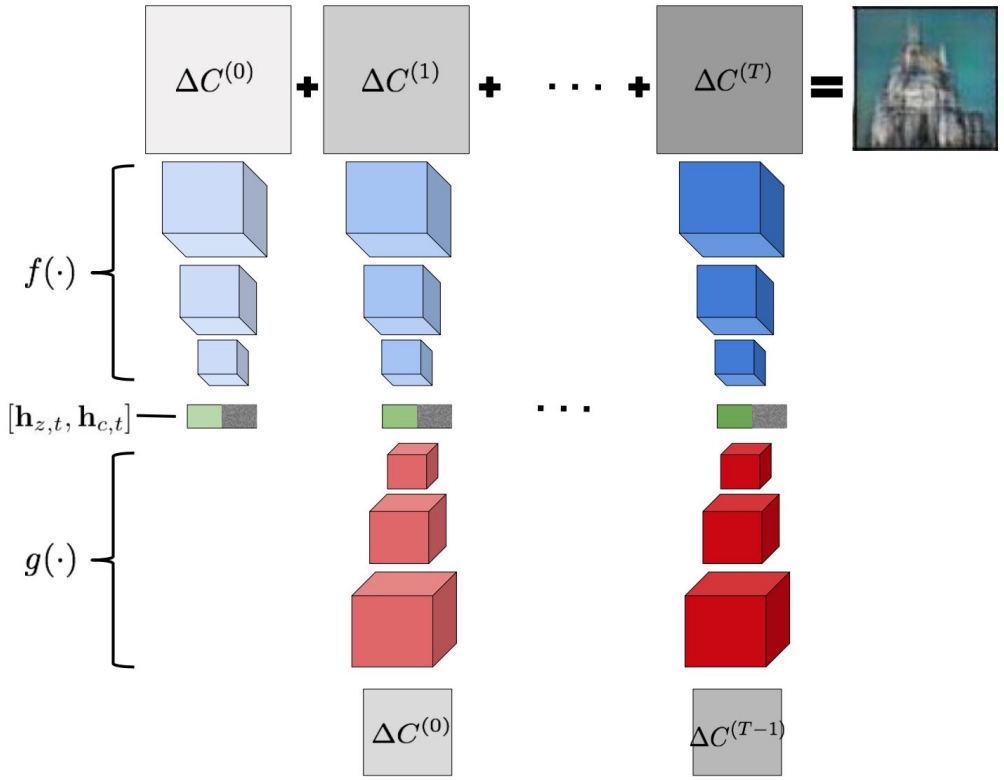
Roland Memisevic
Montreal Institute for Learning Algorithms
University of Montreal
memisevr@iro.umontreal.ca

December 2016

DRAW Network



Generative Recurrent Adverserial Network (GRAN)



They Generated Much Sharper Images

CFAR10

LSUN