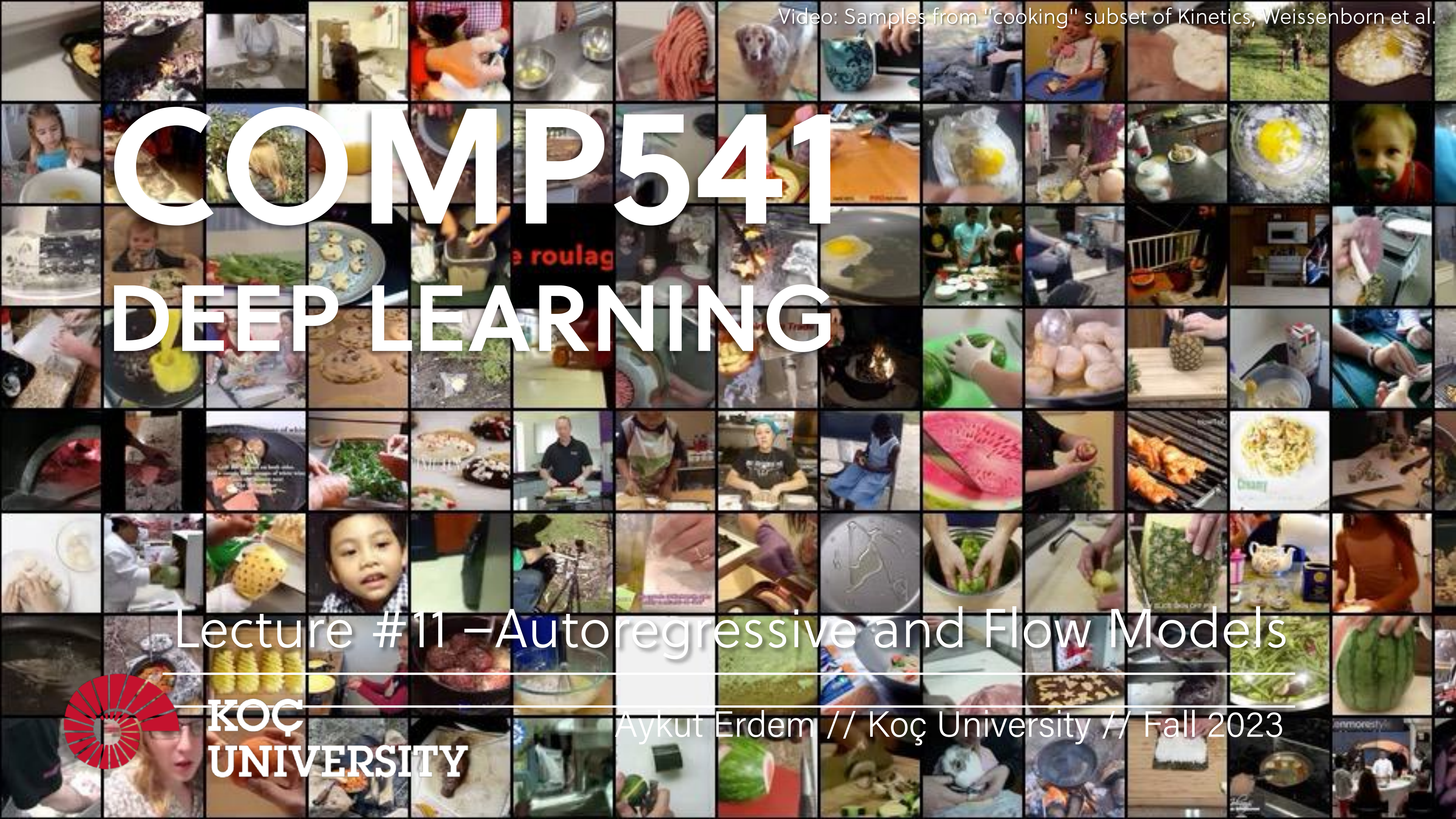


Video: Samples from "cooking" subset of Kinetics, Weissenborn et al.

COMP541
DEEP LEARNING

Lecture #11 –Autoregressive and Flow Models

KOÇ UNIVERSITY

Aykut Erdem // Koç University // Fall 2023

Previously on COMP541

- supervised vs unsupervised learning
- generative modeling
- basic foundations
  - sparse coding
  - autoencoders
- generative adversarial networks (GANs)


Artificial faces synthesized by StyleGAN (Nvidia)

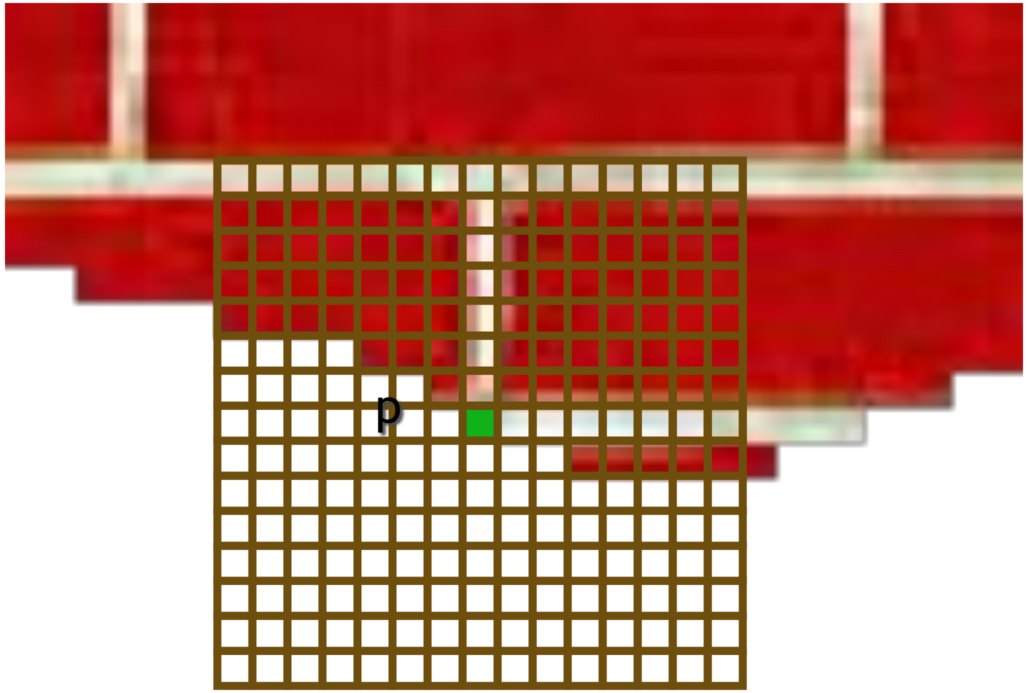
Lecture overview

- autoregressive generative models
- normalizing flow models

Disclaimer: Much of the material and slides for this lecture were borrowed from
—Bill Freeman, Antonio Torralba and Phillip Isola's MIT 6.869 class
—Nal Kalchbrenner's talks on "Generative Modelling as Sequence Learning" and "Generative Models of Language and Images"
—Chin-Wei Huang slides on Normalizing Flows

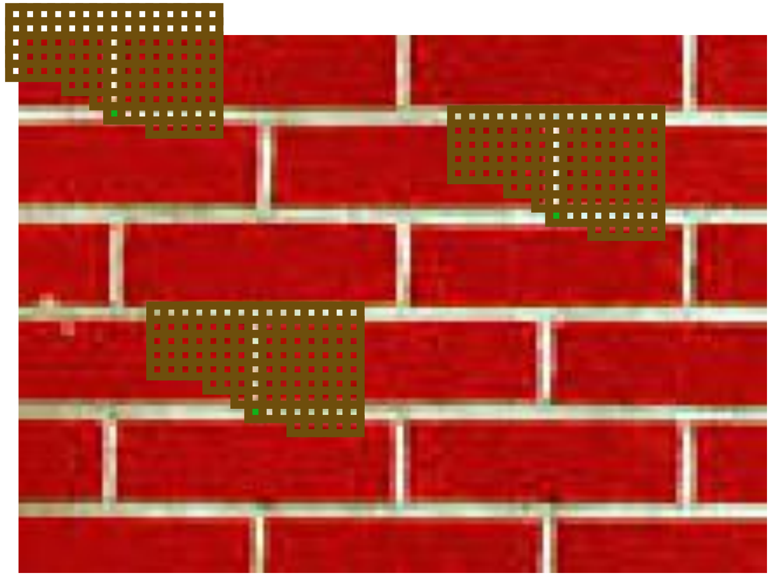
Autoregressive Generative Models

Texture synthesis by non-parametric sampling



Synthesizing a pixel

non-parametric
sampling

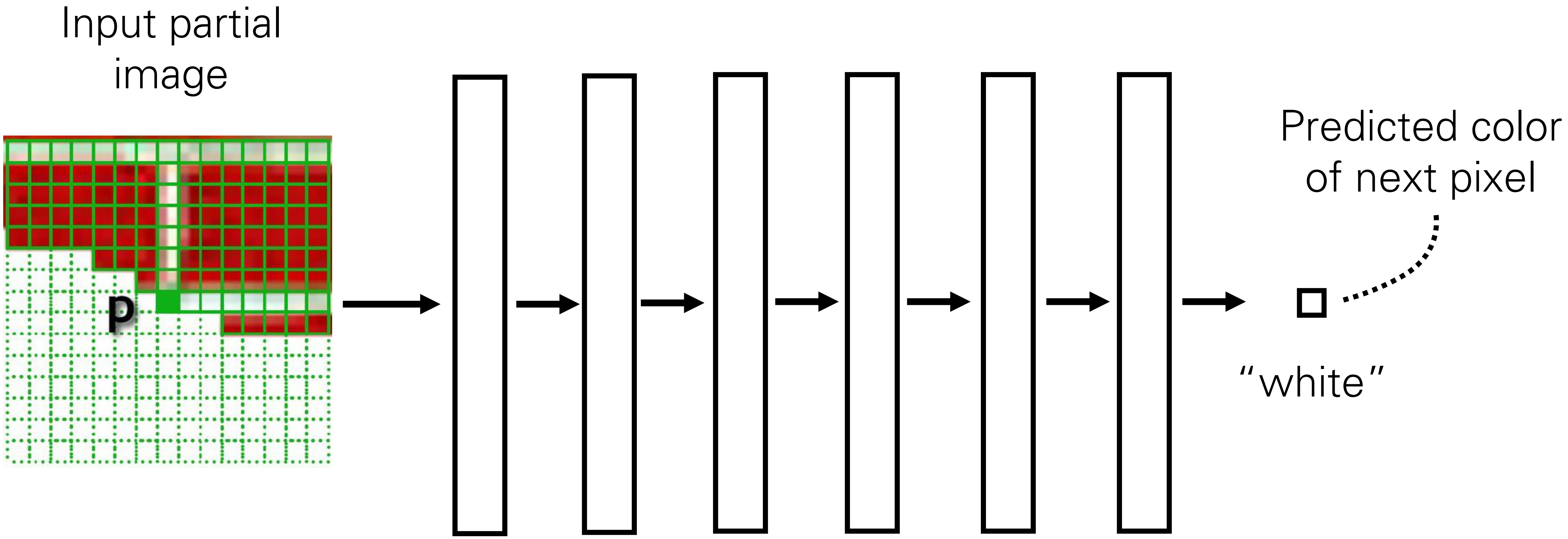


Input image

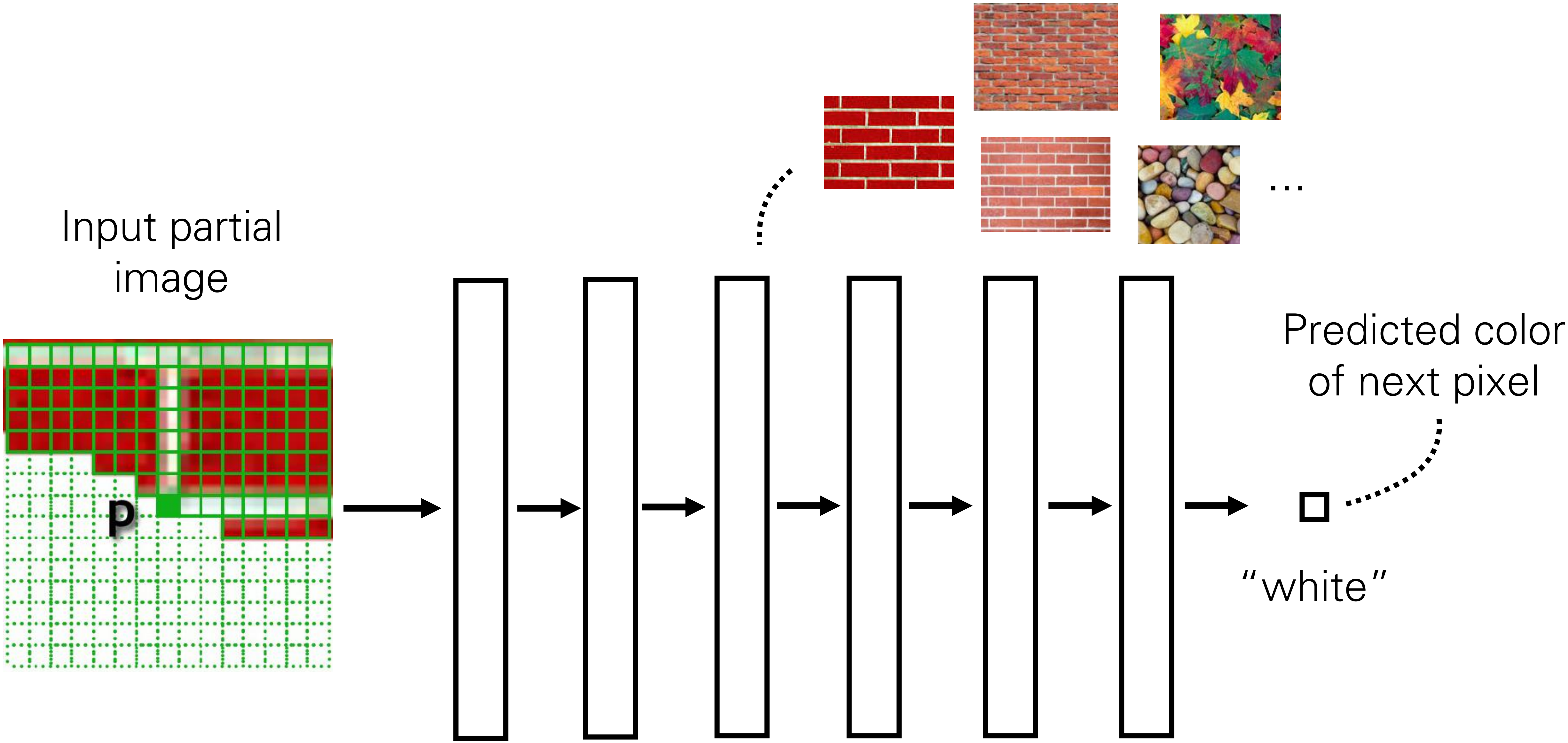
Models $P(p|N(p))$

[Efros & Leung 1999]

Texture synthesis with a deep net



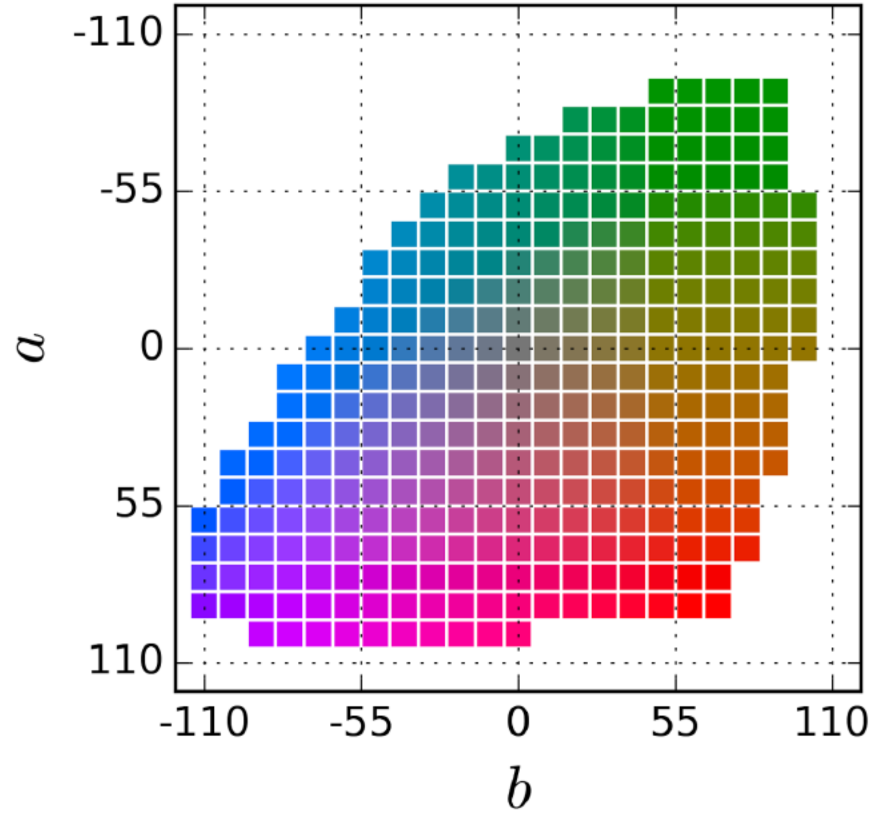
[PixelRNN, PixelCNN, van der Oord et al. 2016]



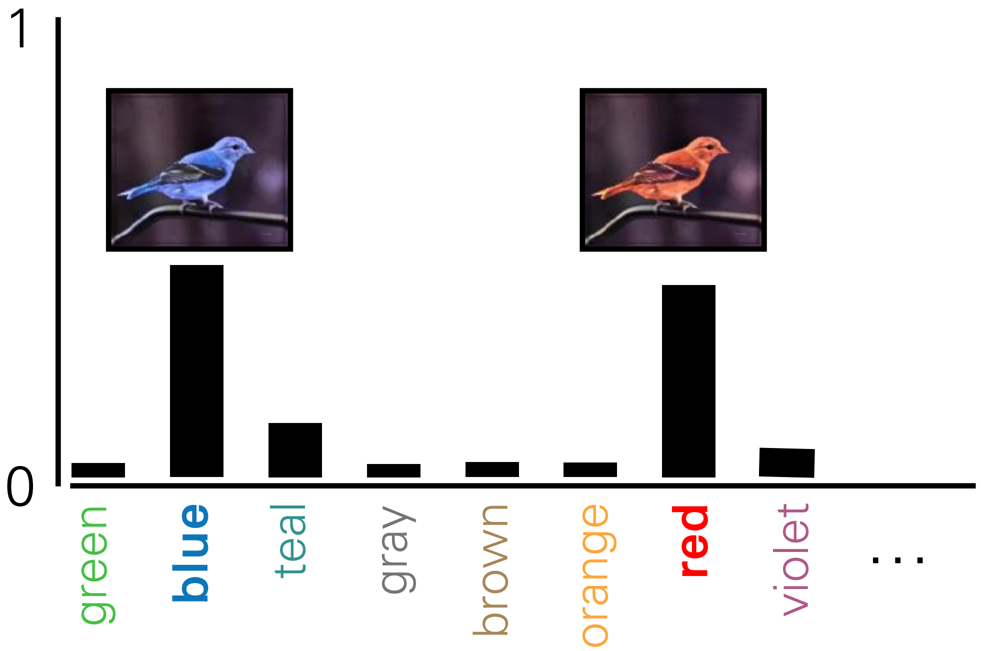
[PixelRNN, PixelCNN, van der Oord et al. 2016]

Idea: We can represent colors as discrete classes

$$\mathbf{y} \in \mathbb{R}^{H \times W \times K}$$



Prediction for a single pixel i,j



$$\mathcal{L}(\mathbf{y}, f_\theta(\mathbf{x})) = H(\mathbf{y}, \texttt{softmax}(f_\theta(\mathbf{x})))$$

And we can interpret the learner as modeling P(next pixel | previous pixels):

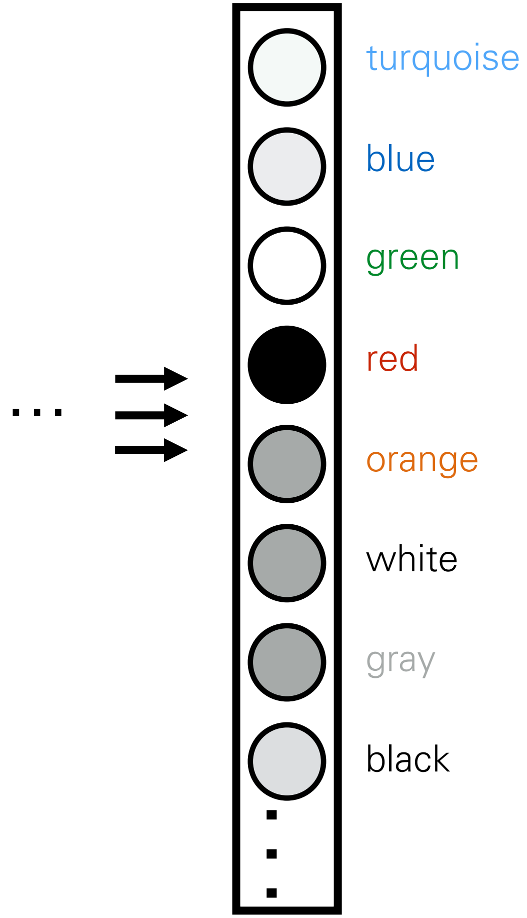
Softmax regression (a.k.a. multinomial logistic regression)

$$\hat{\mathbf{y}} \equiv [P_\theta(Y = 1|X = \mathbf{x}), \dots, P_\theta(Y = K|X = \mathbf{x})]$$ ← predicted probability of each class given input $\mathbf{x}$

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$ ← picks out the -log likelihood of the ground truth class $\mathbf{y}$ under the model prediction $\hat{\mathbf{y}}$

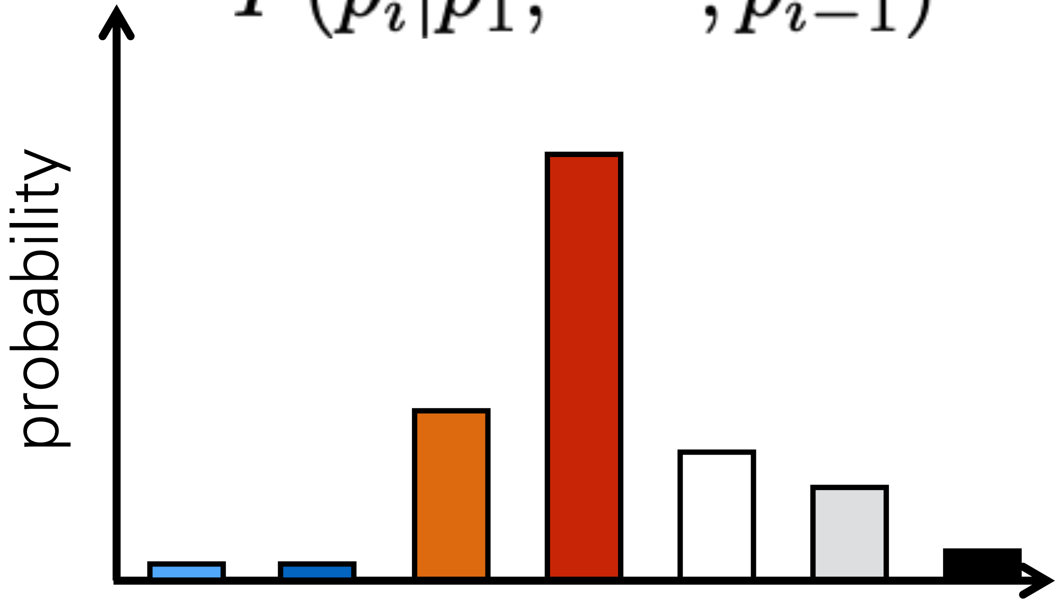
$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} H(\mathbf{y}_i, \hat{\mathbf{y}}_i)$$ ← max likelihood learner!

Network output

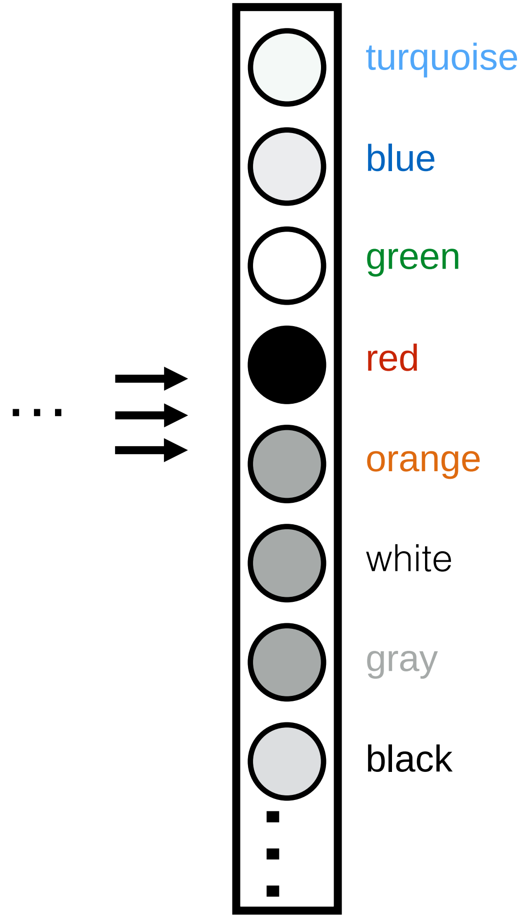


P(next pixel | previous pixels)

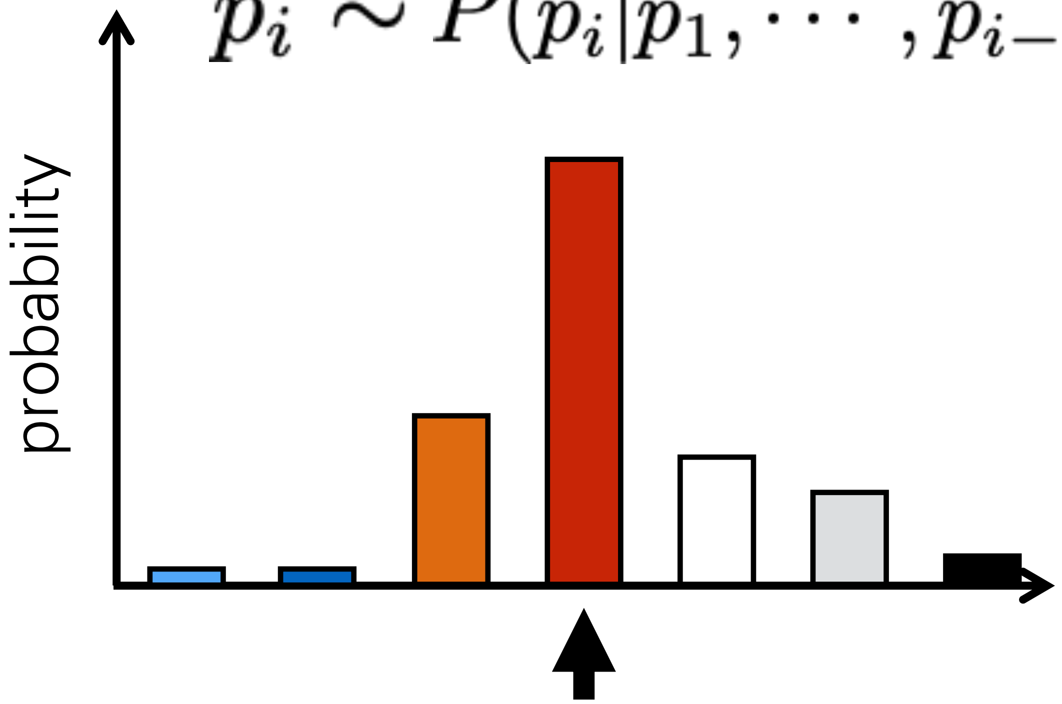
$$P(p_i|p_1, \cdots, p_{i-1})$$



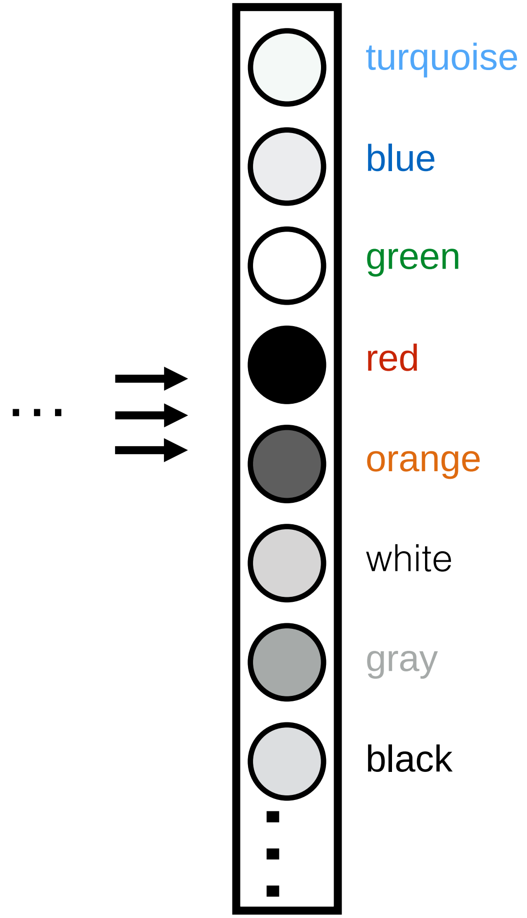
Network output

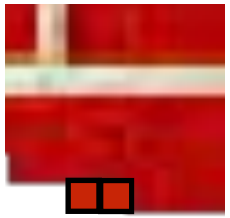


$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$

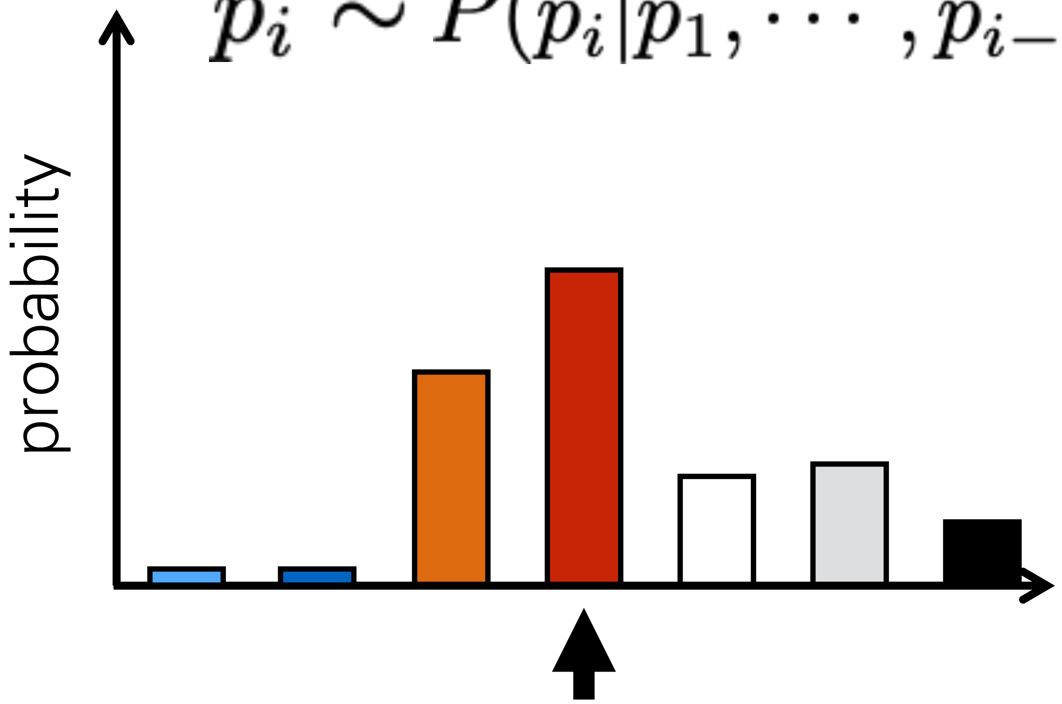


Network output

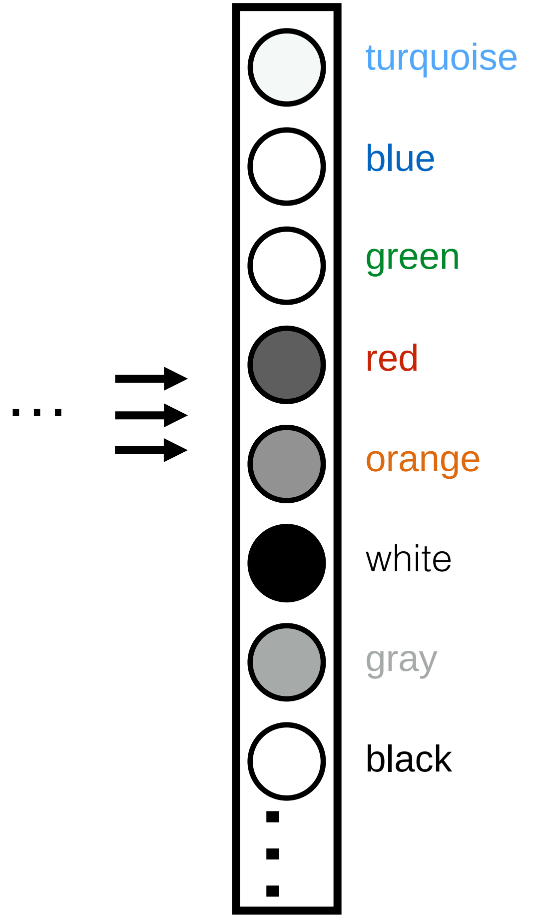


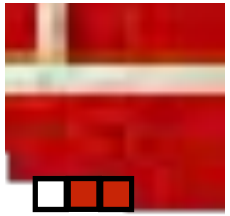


$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$

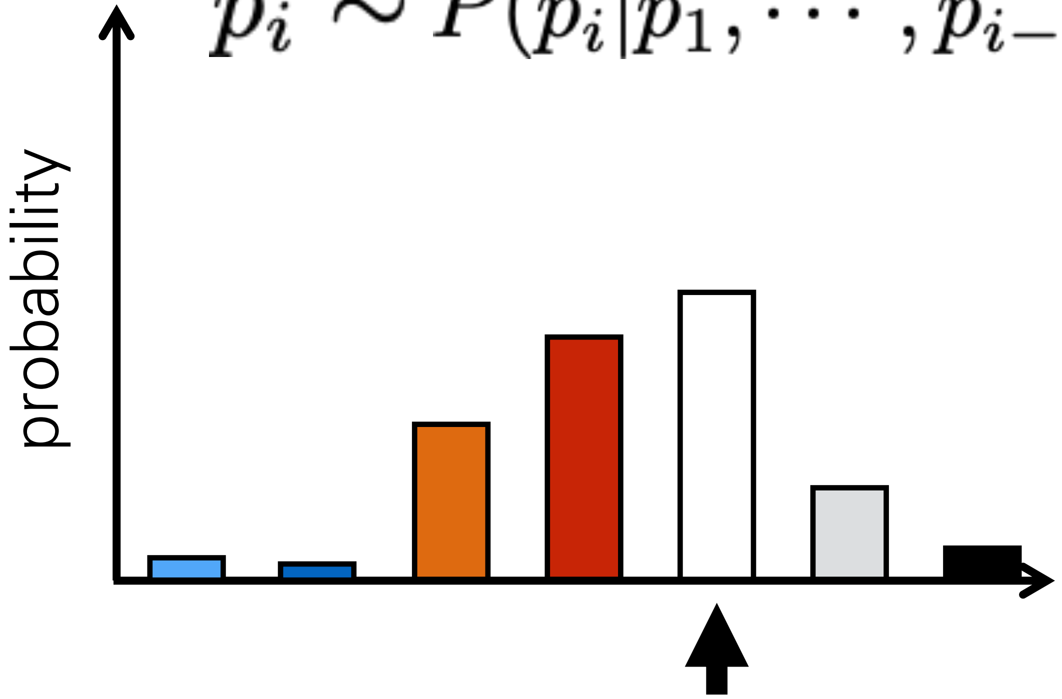


Network output

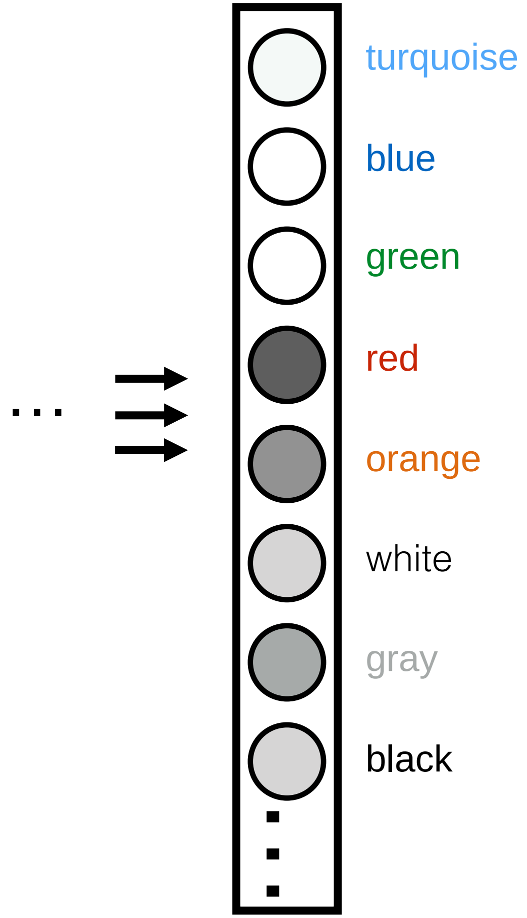


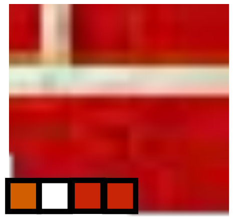


$$p_i \sim P(p_i|p_1,\cdots,p_{i-1})$$

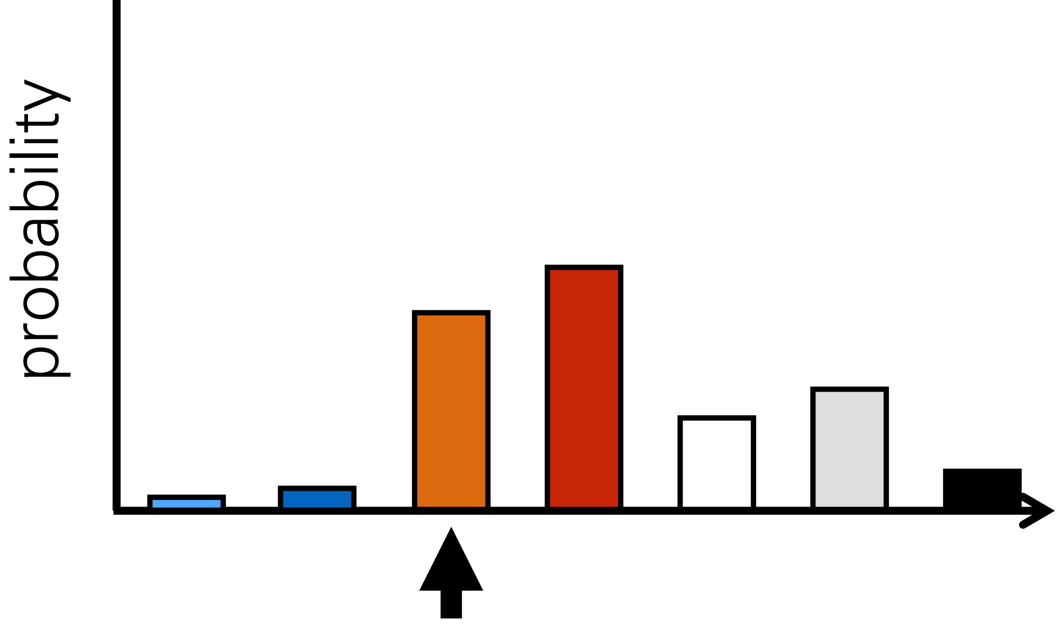


Network output





$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$

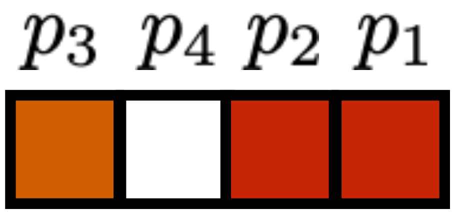


$$p_1 \sim P(p_1)$$

$$p_2 \sim P(p_2|p_1)$$

$$p_3 \sim P(p_3|p_1, p_2)$$

$$p_4 \sim P(p_4|p_1, p_2, p_3)$$



$$\{p_1, p_2, p_3, p_4\} \sim P(p_4|p_1, p_2, p_3)P(p_3|p_1, p_2)P(p_2|p_1)P(p_1)$$

$$p_i \sim P(p_i|p_1, \ldots, p_{i-1})$$

$$\mathbf{p} \sim \prod_{i=1}^{N} P(p_i|p_1, \ldots, p_{i-1})$$

Autoregressive probability model

$$\mathbf{p} \sim \prod_{i=1}^{N} P(p_i | p_1, \ldots, p_{i-1})$$

$$P(\mathbf{p}) = \prod_{i=1}^{N} P(p_i | p_1, \ldots, p_{i-1})$$ ← General product rule

The sampling procedure we defined above takes exact samples from the learned probability distribution (pmf).

Multiplying all conditionals evaluates the probability of a full joint configuration of pixels.

Learning the Distribution of Natural Data

$$p(\mathbf{x}) = \prod_i p(x_i|\mathbf{x}_<)$$

1D sequences such as text or sound

$$p(\mathbf{x}) = \prod_j \prod_i p(x_{i,j}|\mathbf{x}_<)$$

2D tensors such as images

$$p(\mathbf{x}) = \prod_k \prod_j \prod_i p(x_{i,j,k}|\mathbf{x}_<)$$

3D tensors such as videos

- Fully visible belief networks [Frey et al.,1996] [Frey, 1998]
- NADE/MADE [Larochelle and Murray, 2011] [Germain et al., 2015]
- PixelRNN/PixelCNN (Images) [van den Oord, Kalchbrenner, Kavukcuoglu, 2016] [van den Oord, Kalchbrenner, Vinyals, et al., 2016]
- Video Pixel Nets (Videos) [Kalchbrenner, van den Oord, Simonyan, et al., 2016]
- ByteNet (Language/seq2seq) [Kalchbrenner, Espeholt, Simonyan, et al., 2016]
- WaveNet (Audio) [van den Oord, Dieleman, Zen, et al., 2016]

Slide adapted from Nal Kalchbrenner

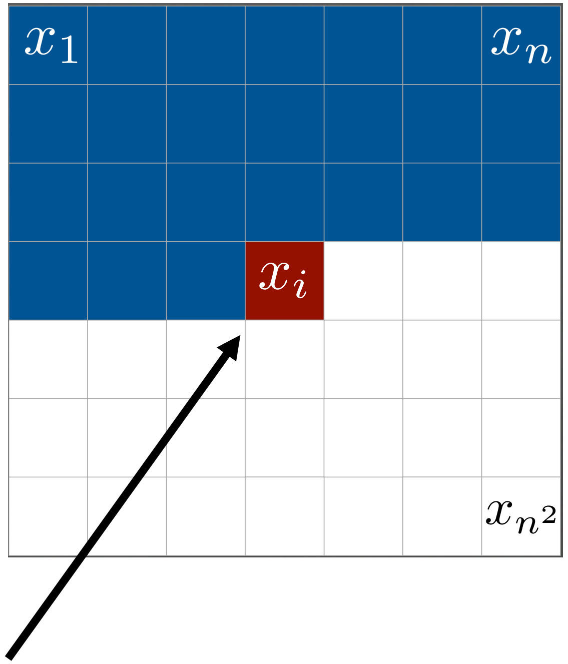
PixelCNN

$P($  $)$

- approach the generation process as sequence modeling problem
- an explicit density model

PixelCNN

$P($  $)$

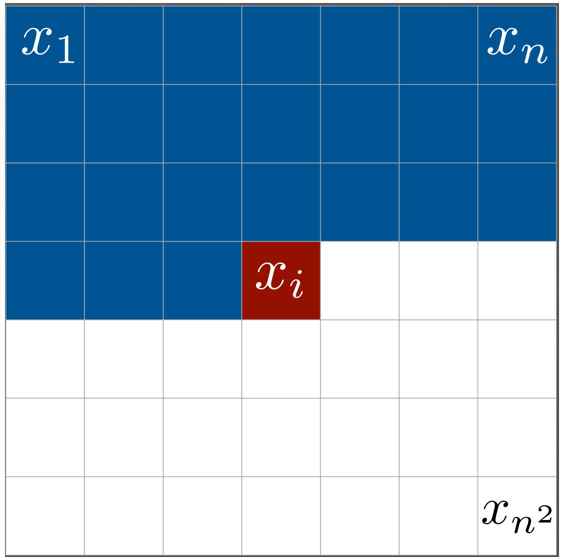


$$p(x_i|\mathbf{x}_<)$$

Slide adapted from Nal Kalchbrenner

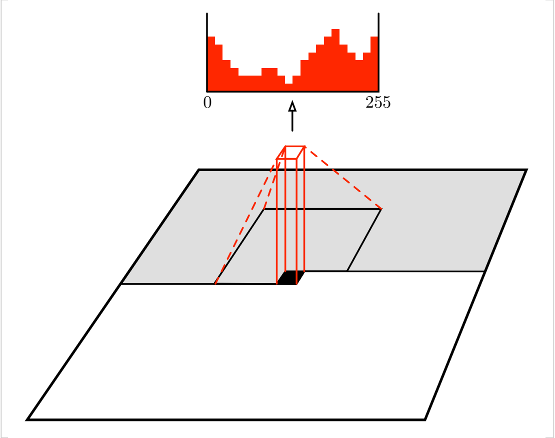
PixelCNN

$P($  $)$

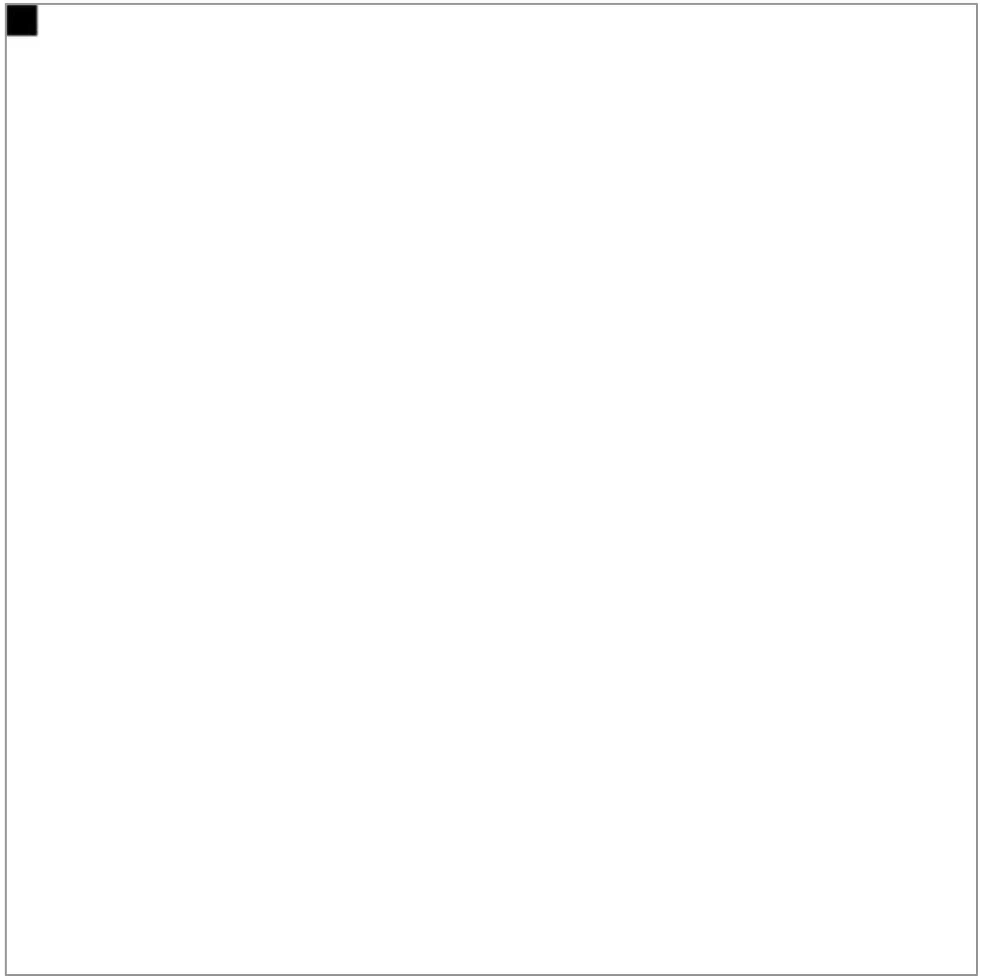


By chain rule and using pixels as variables,

$$P(X) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)$$

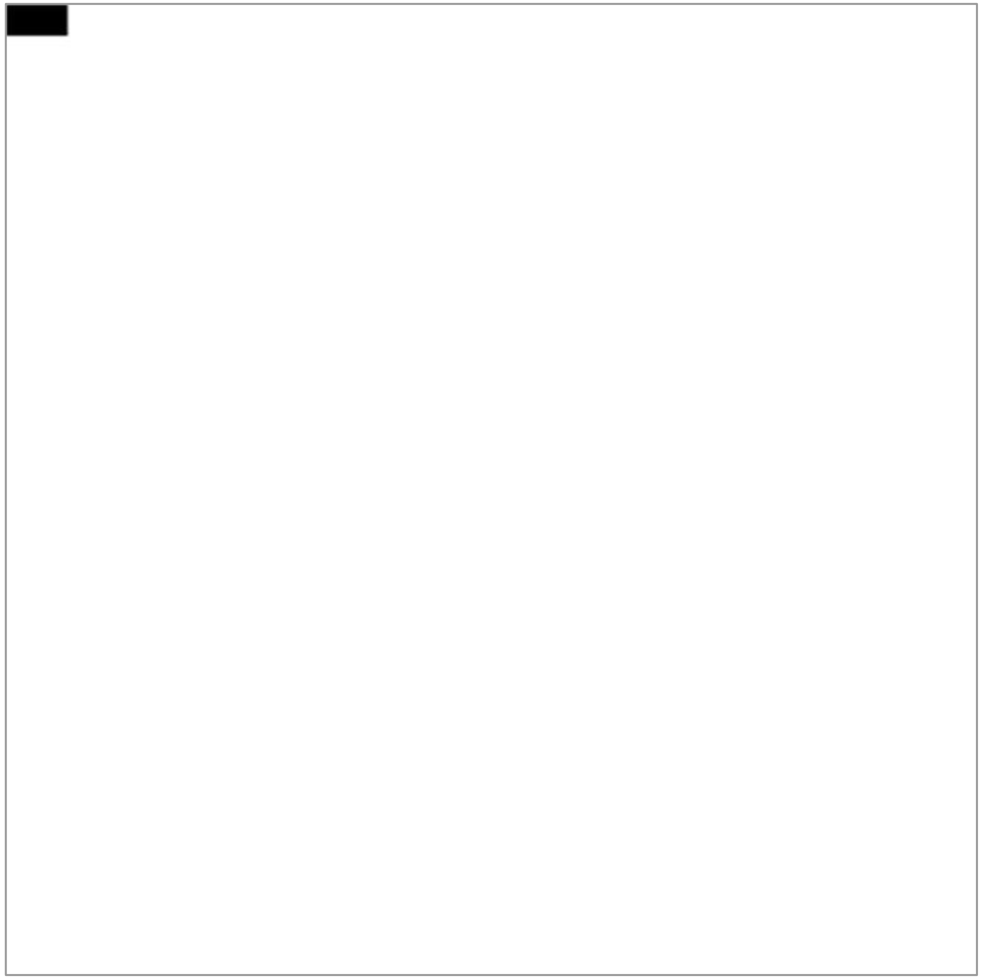


PixelCNN



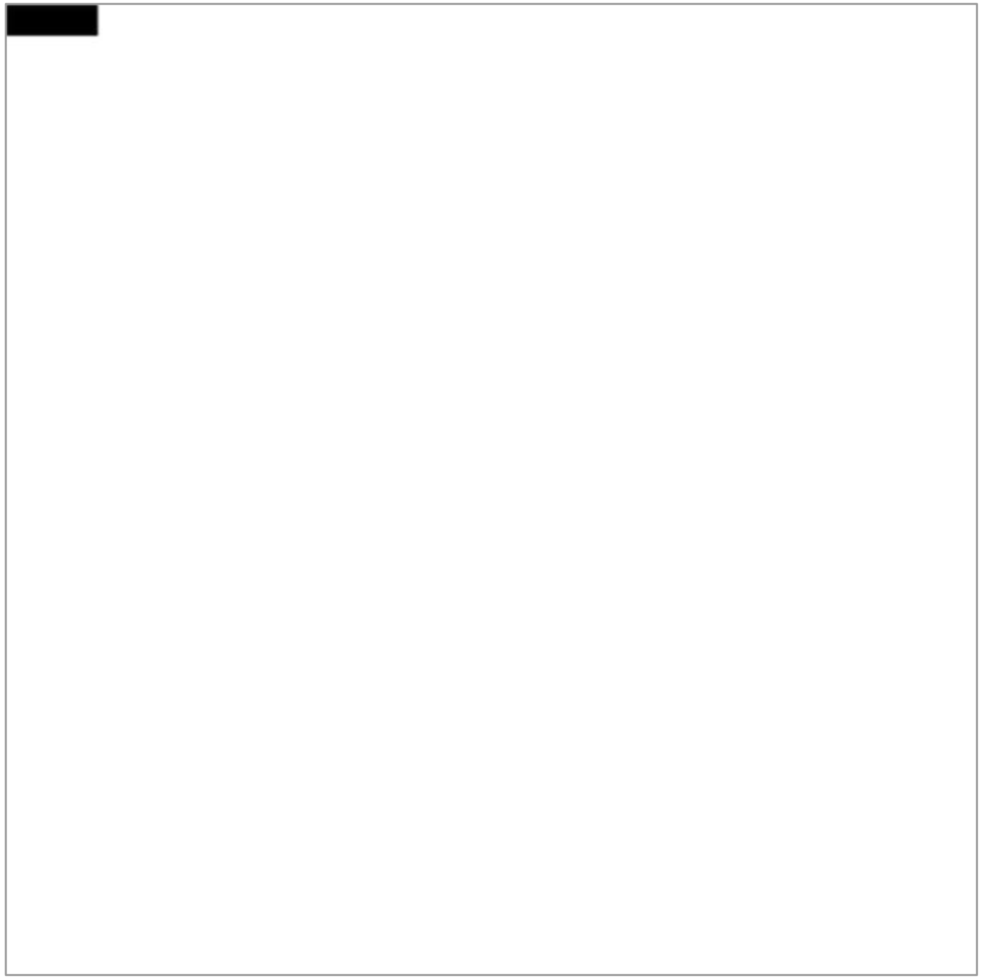
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



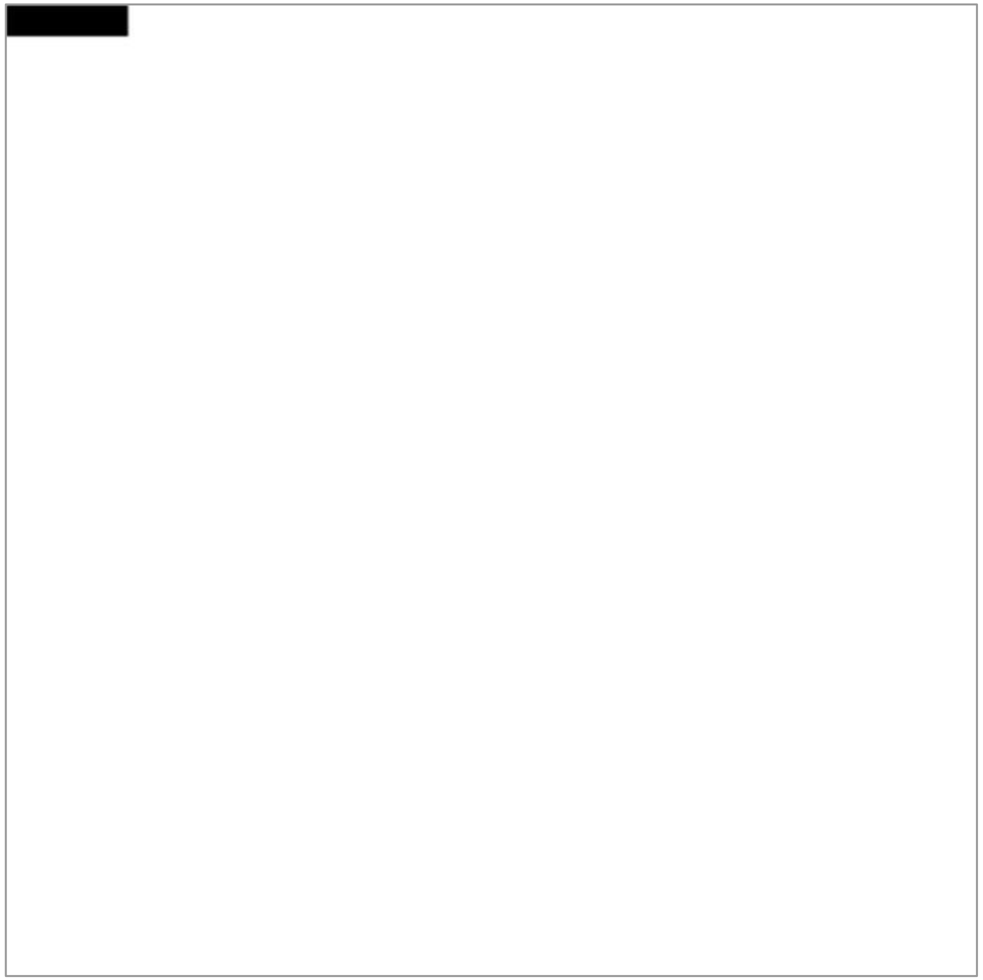
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



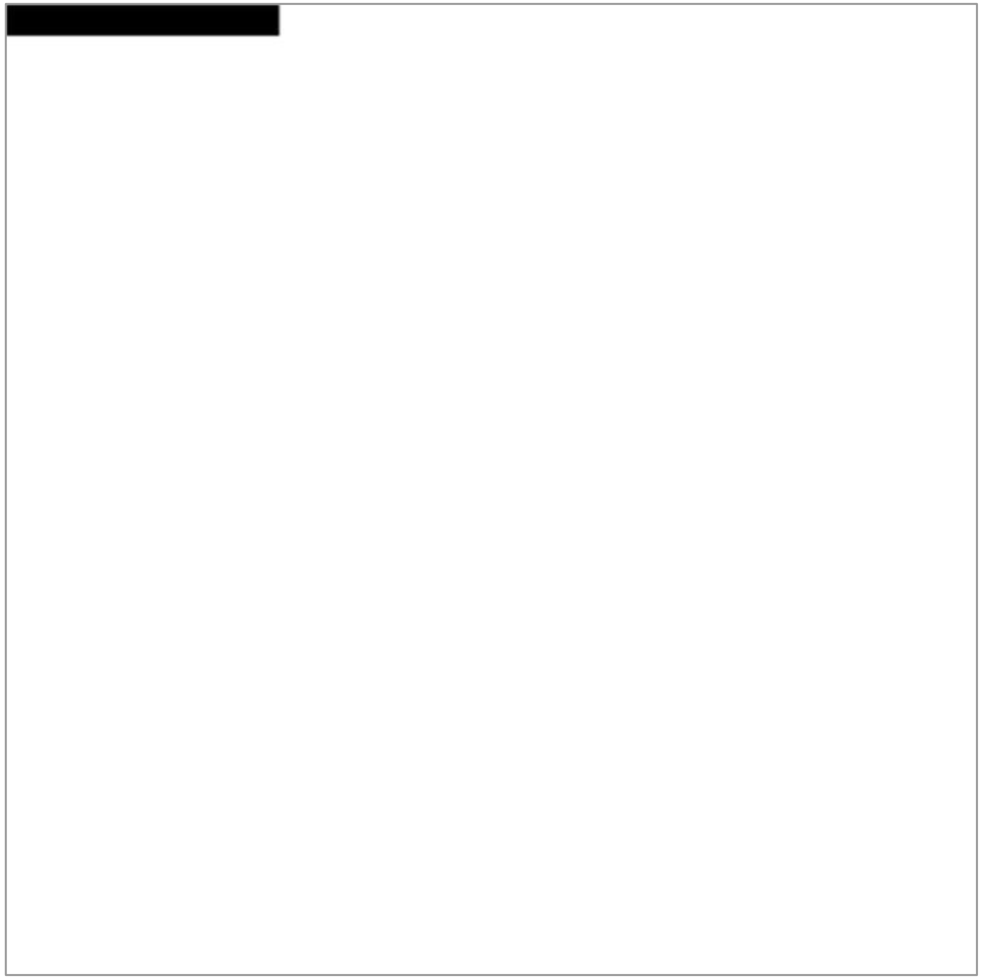
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



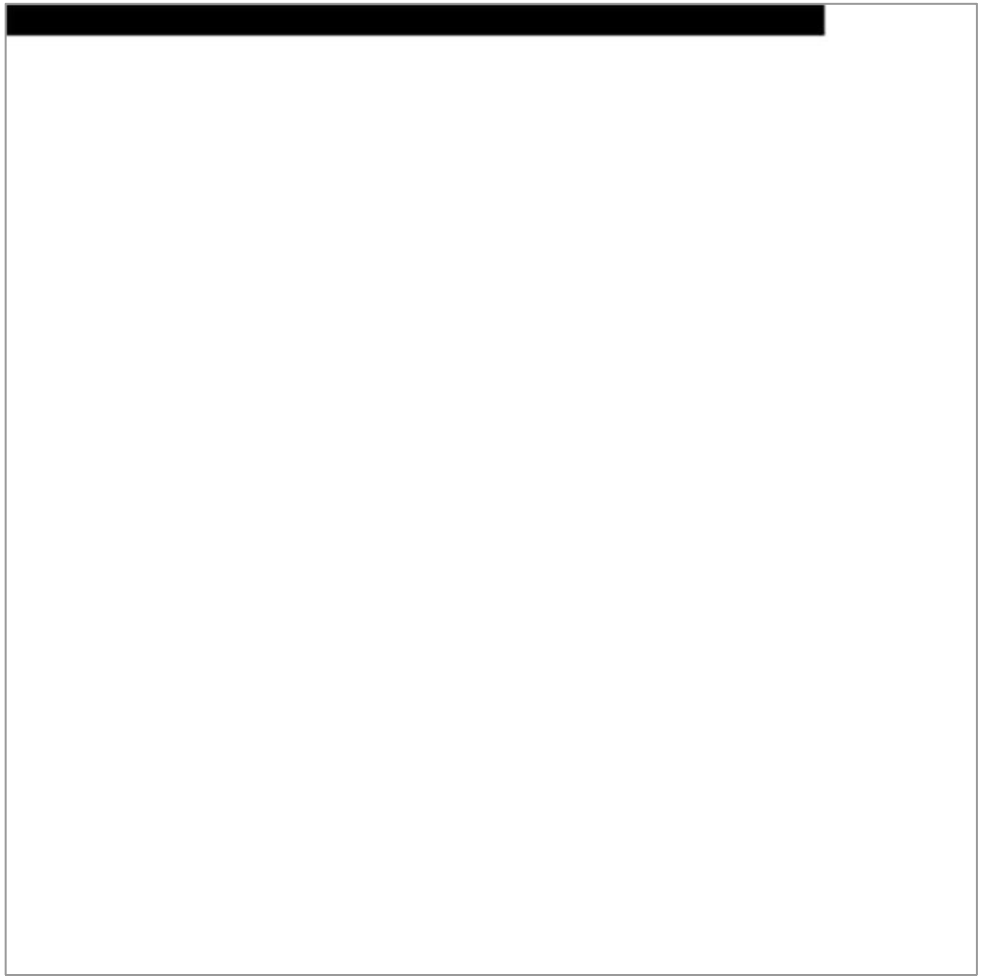
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



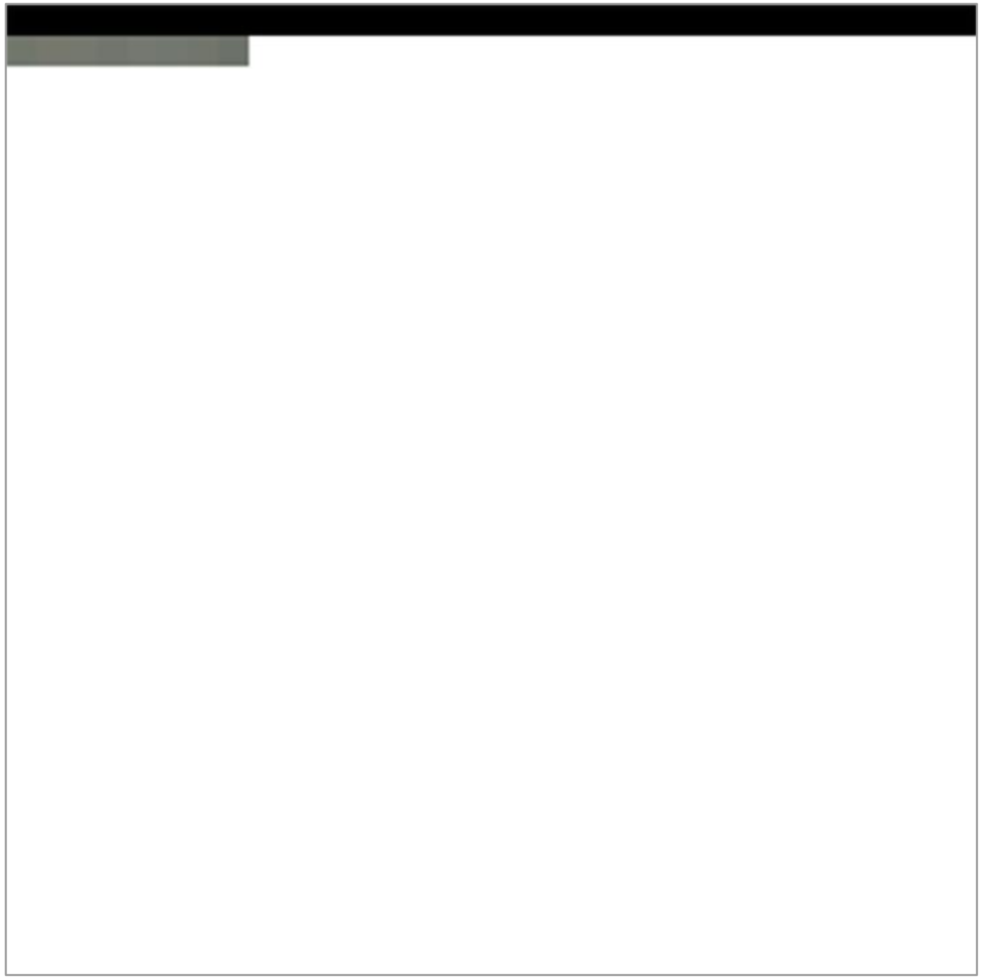
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



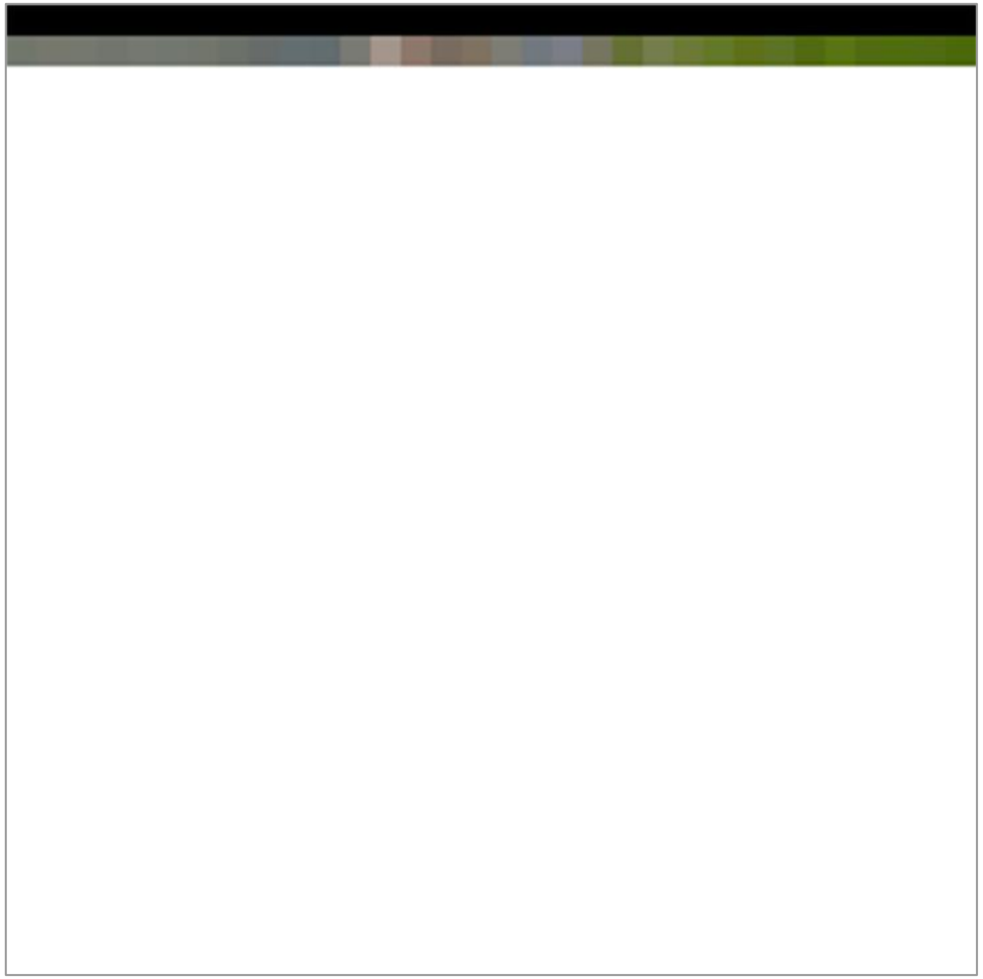
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



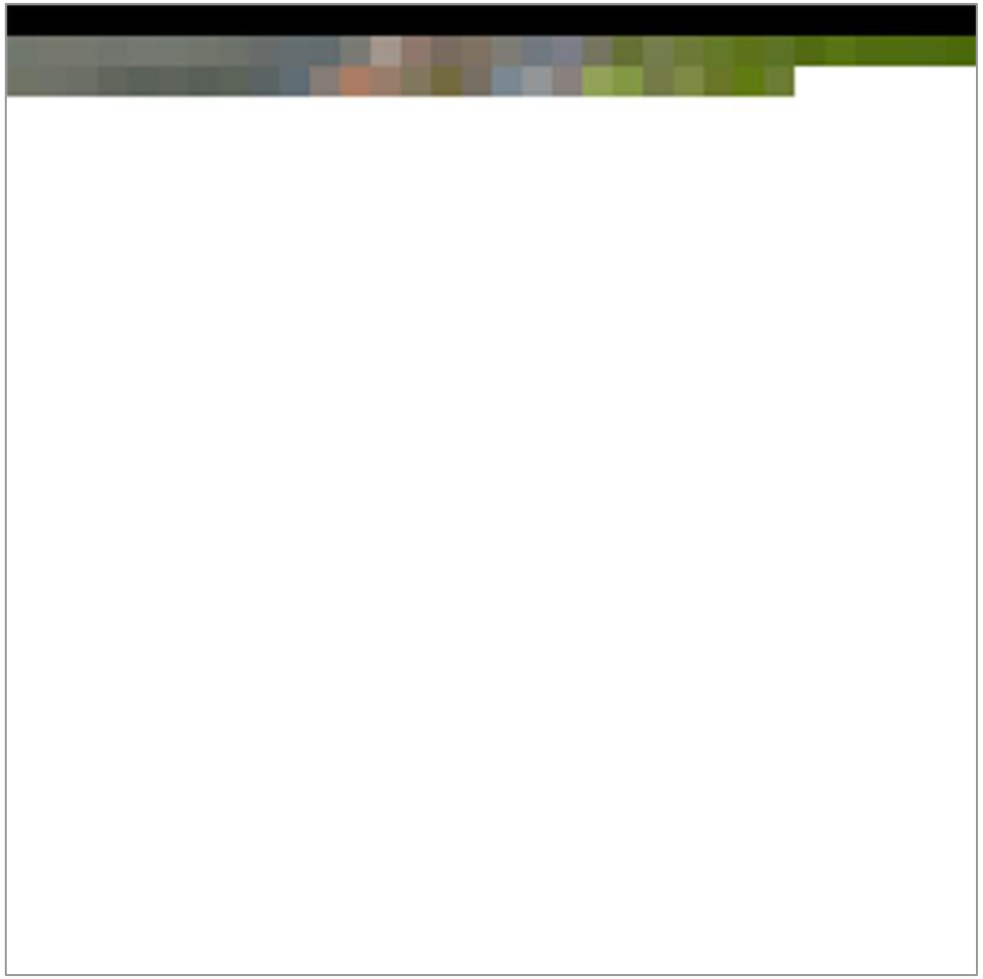
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



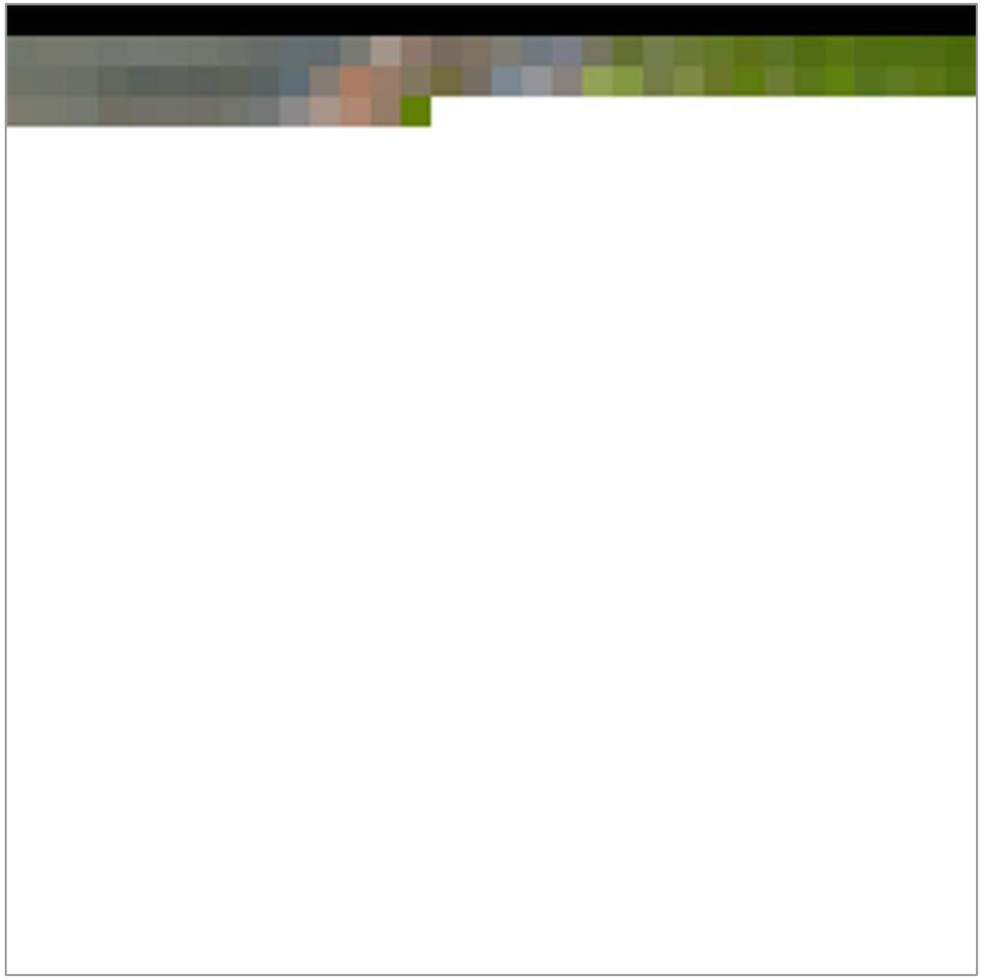
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



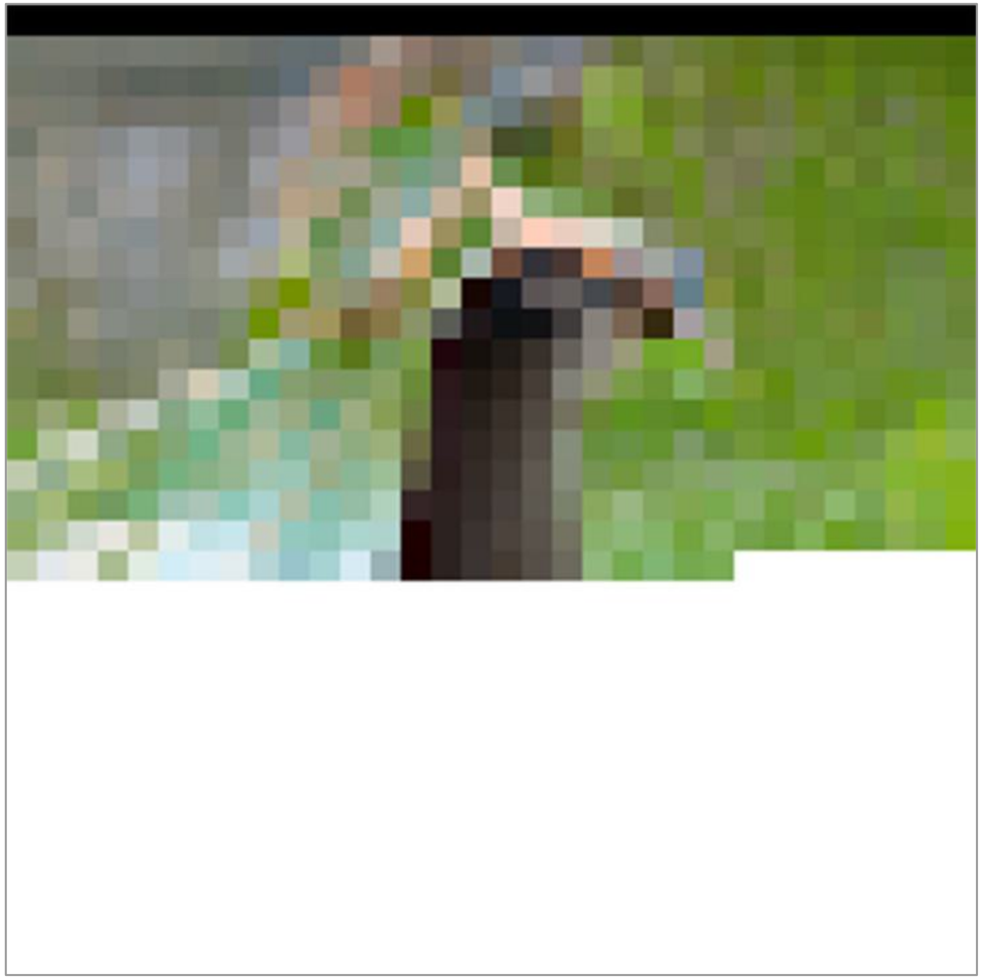
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



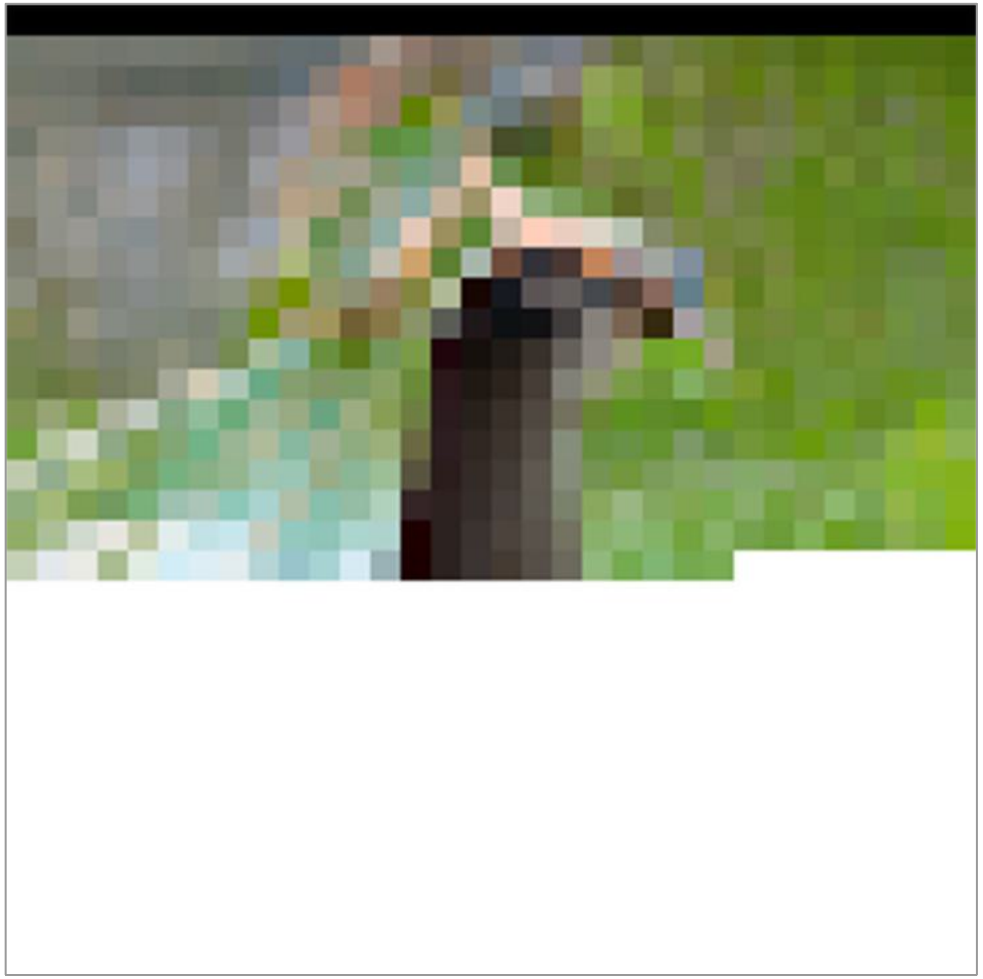
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

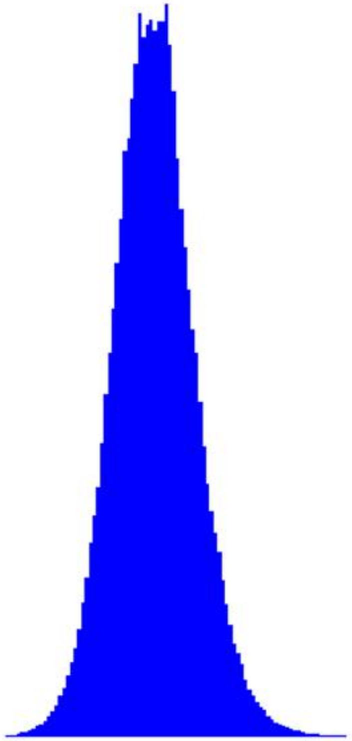
PixelCNN



Slide adapted from
Oriol Vinyals and Navdeep Jaitly

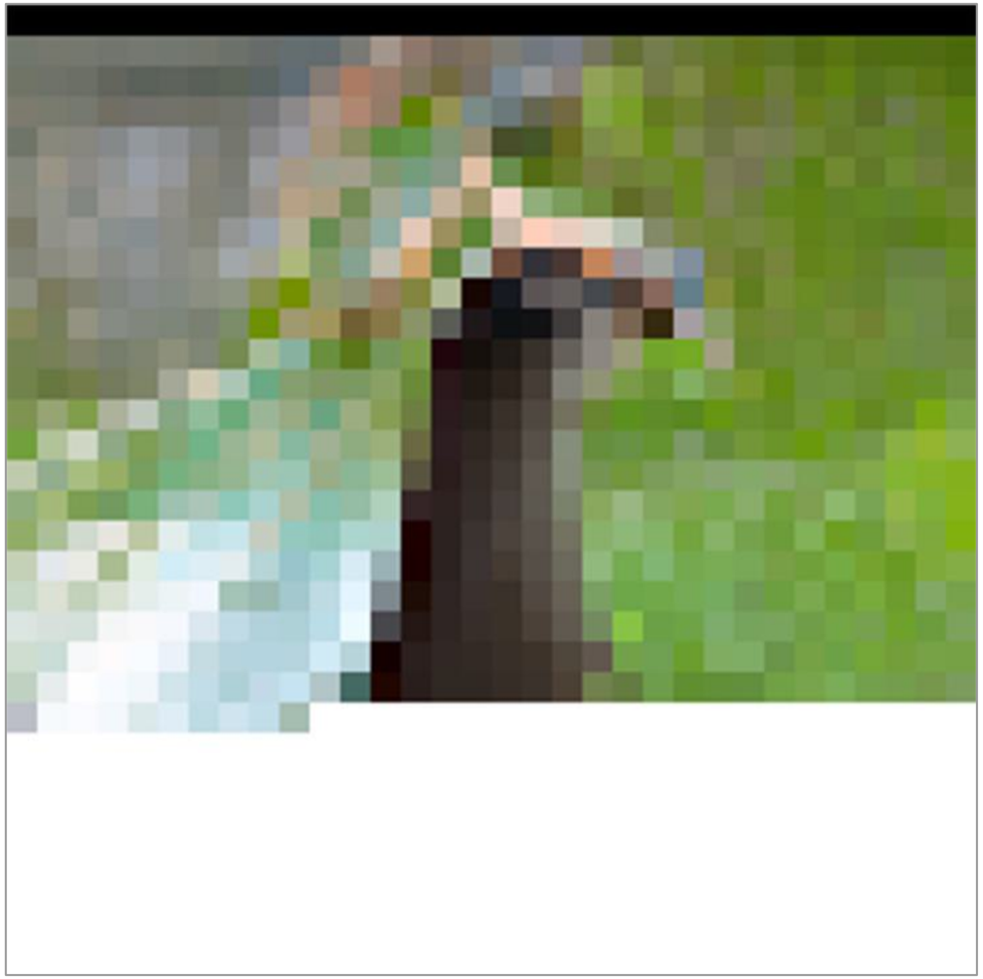
PixelCNN – Softmax Sampling

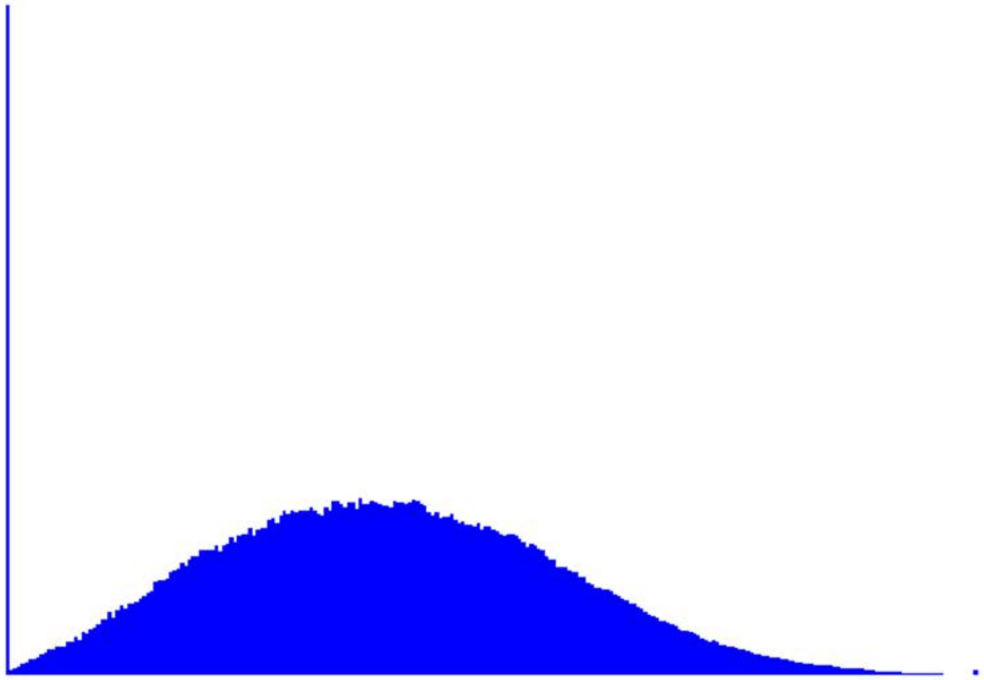




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

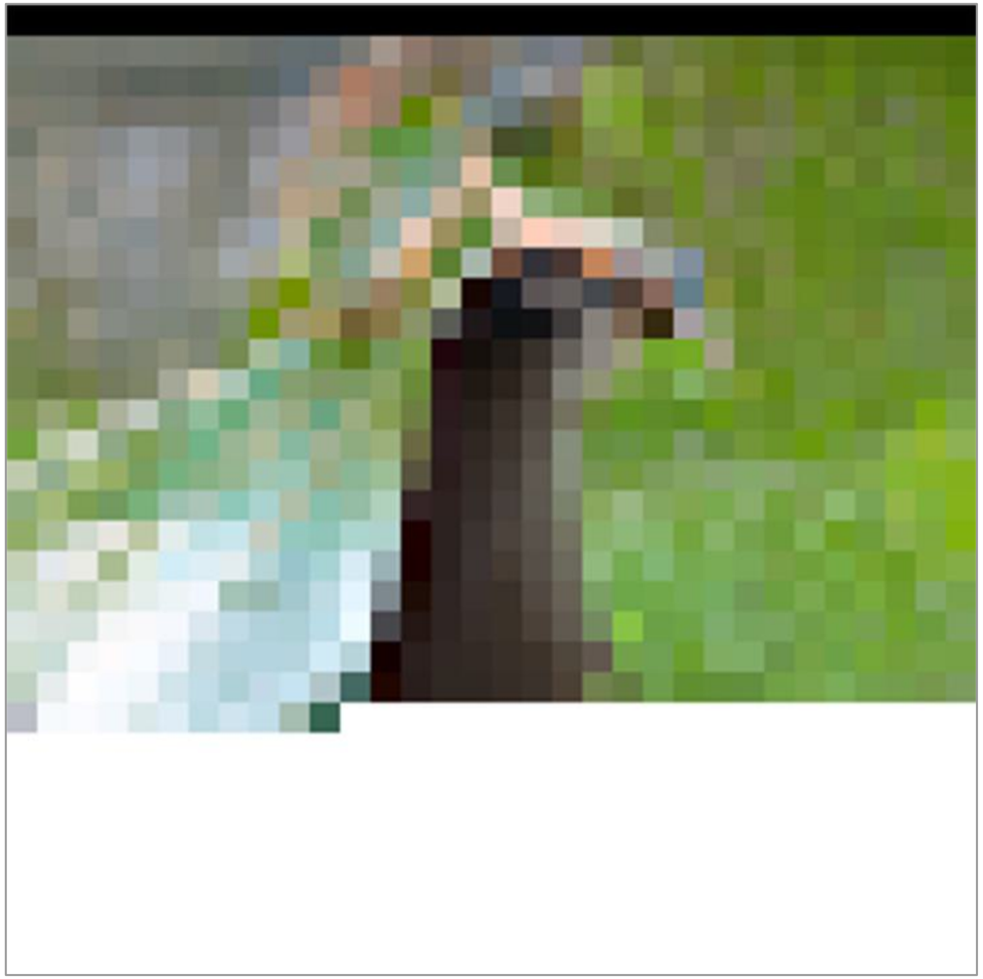
PixelCNN – Softmax Sampling

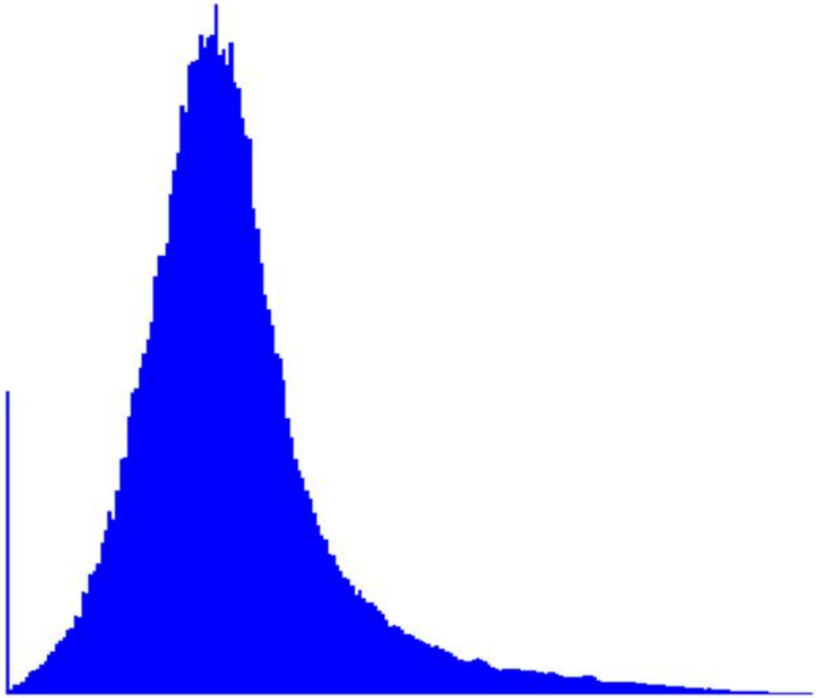




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

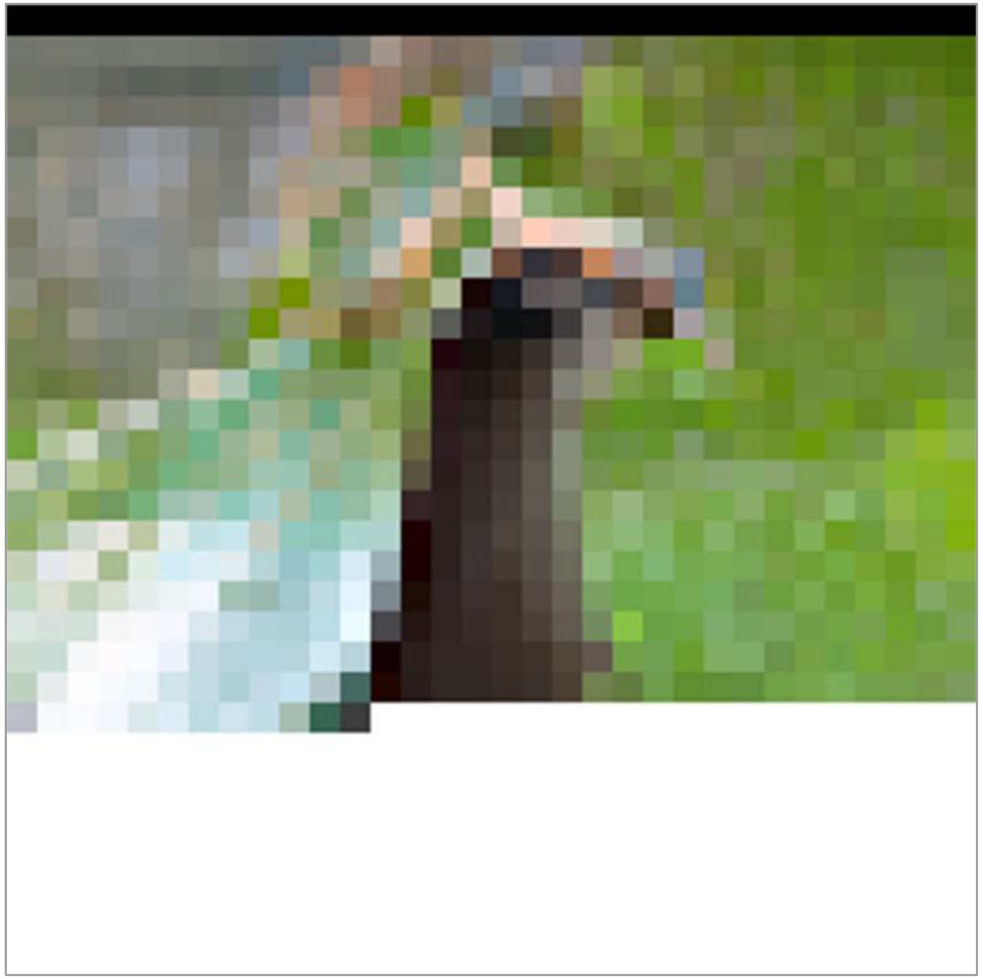
PixelCNN – Softmax Sampling

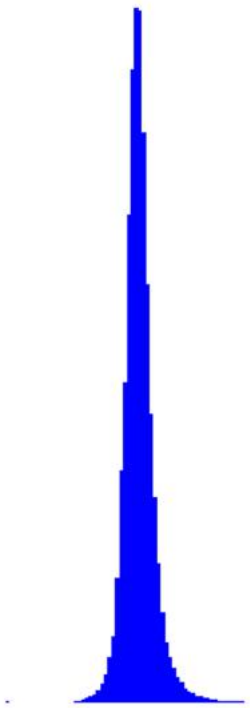




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

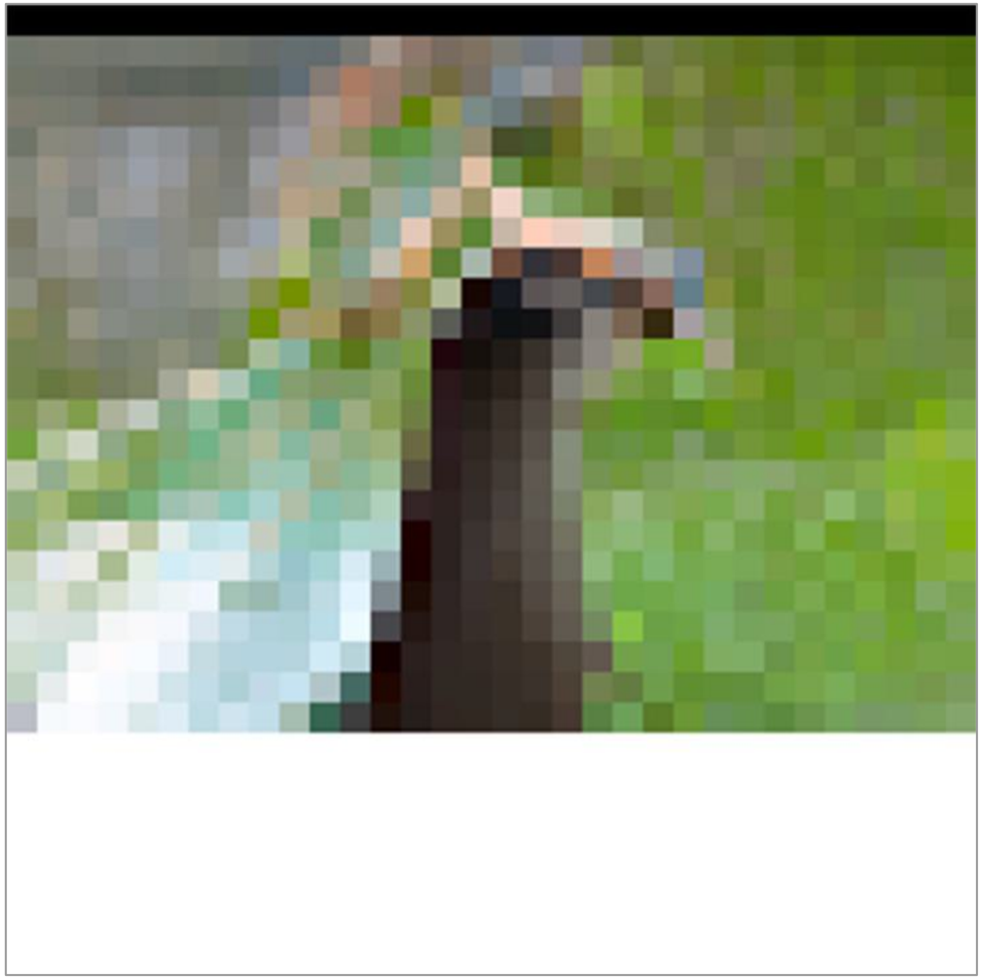
PixelCNN – Softmax Sampling

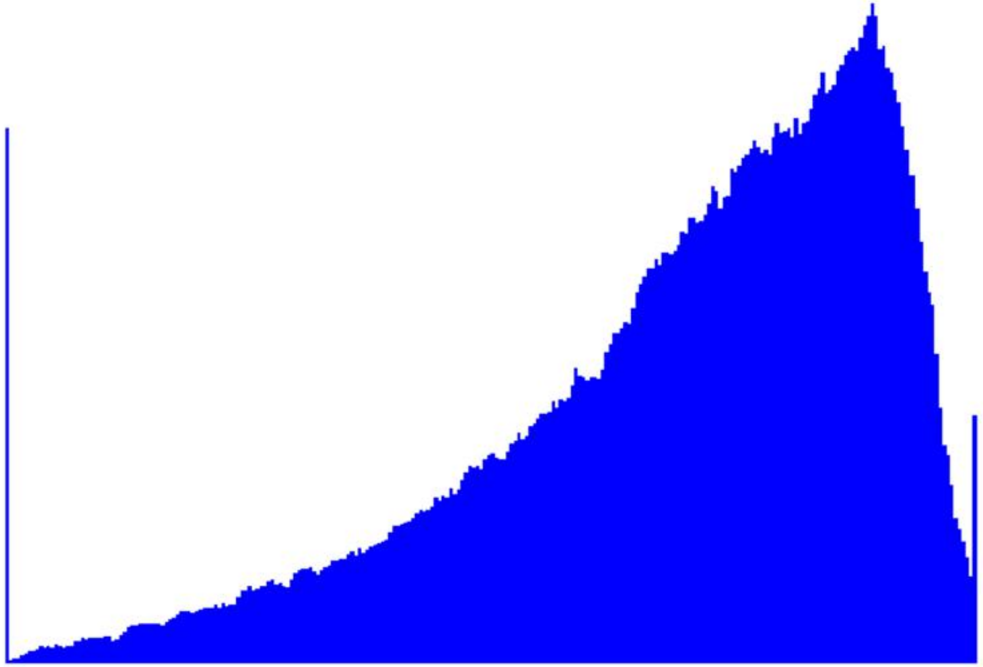




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

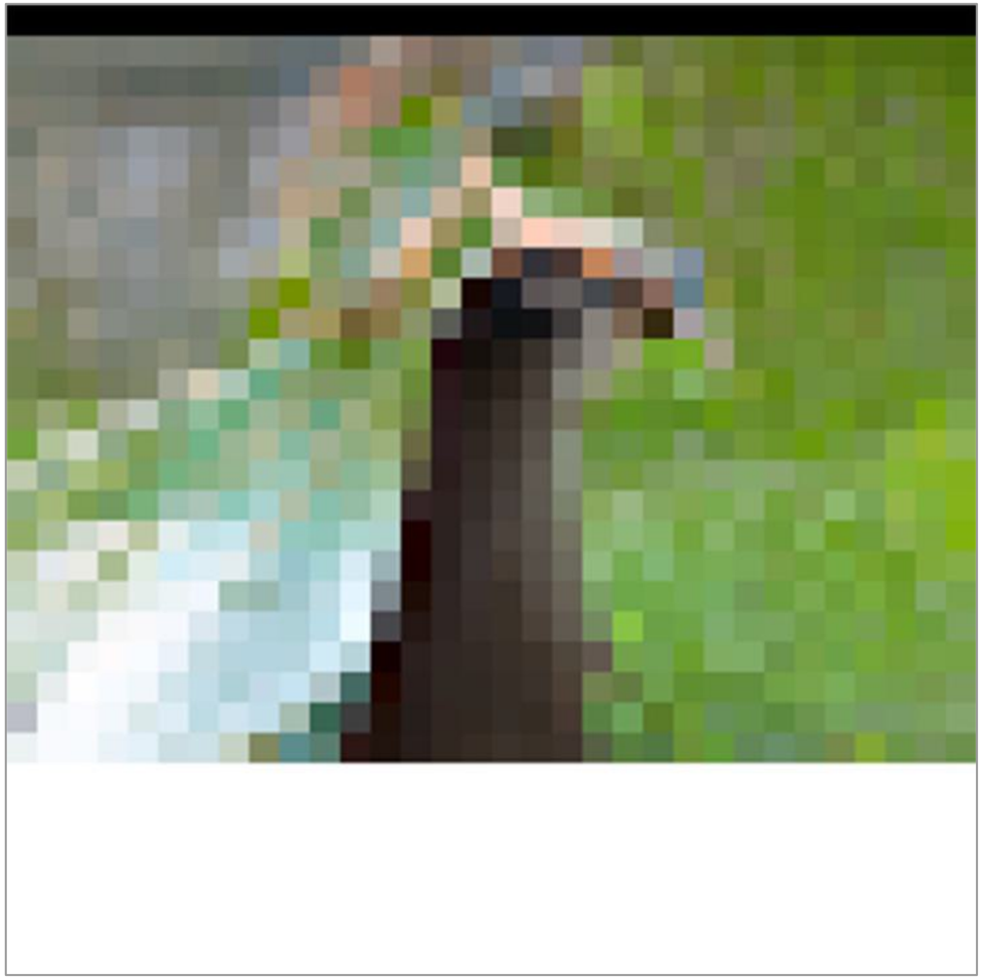
PixelCNN – Softmax Sampling

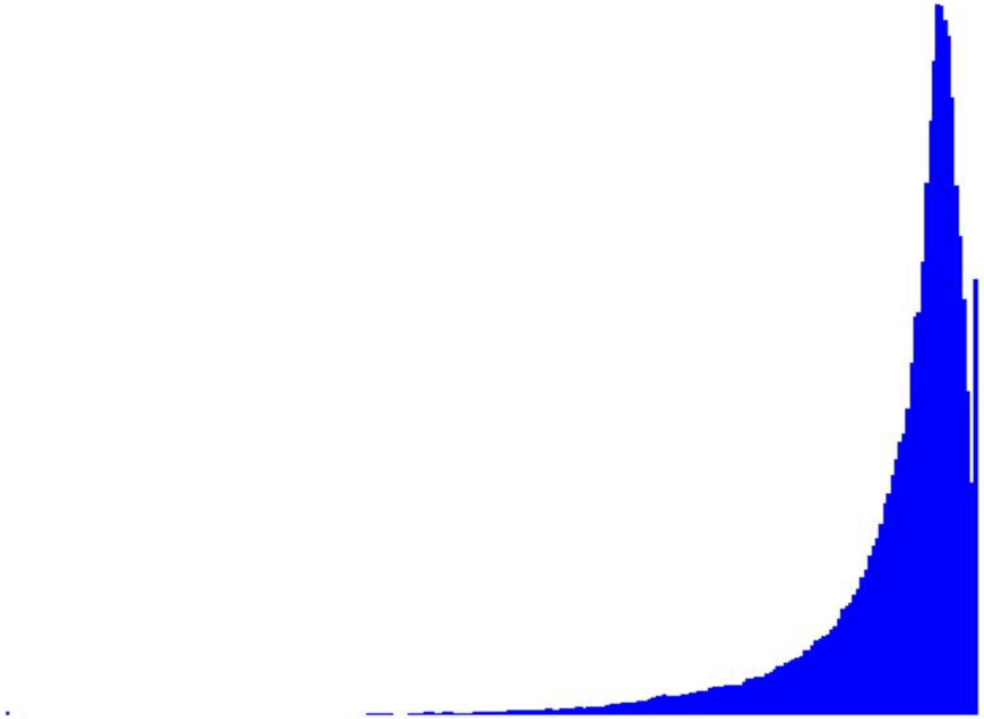




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

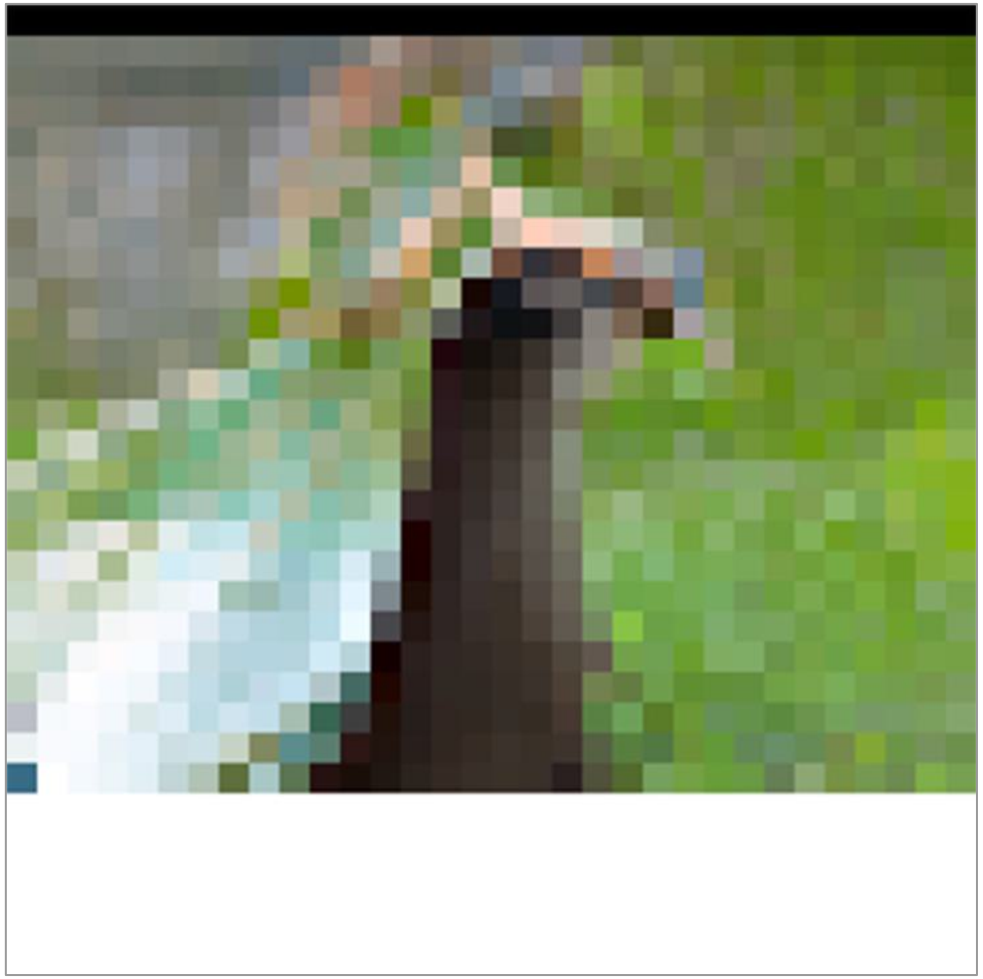
PixelCNN – Softmax Sampling

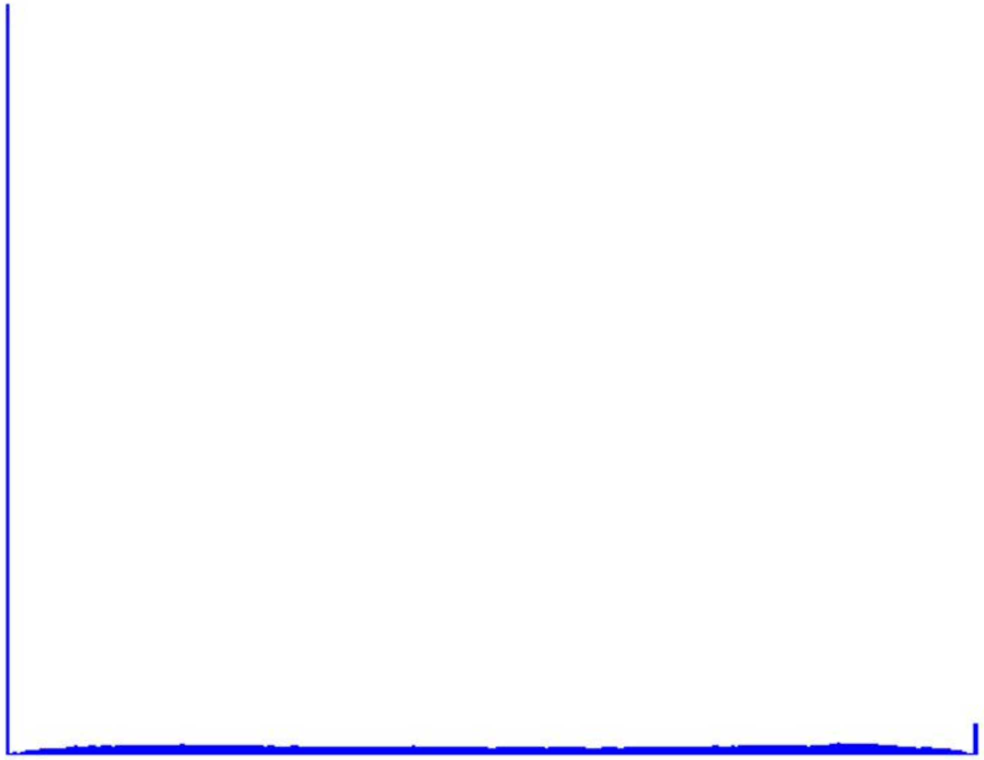




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

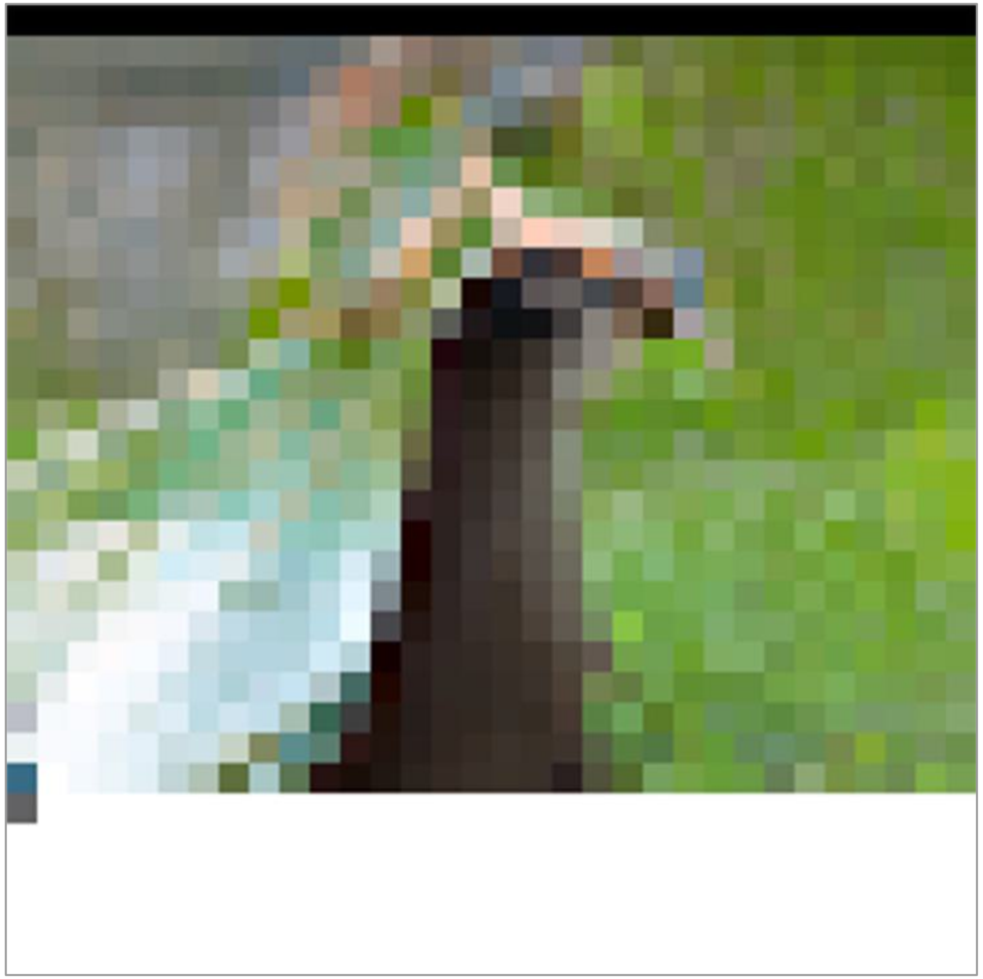
PixelCNN – Softmax Sampling

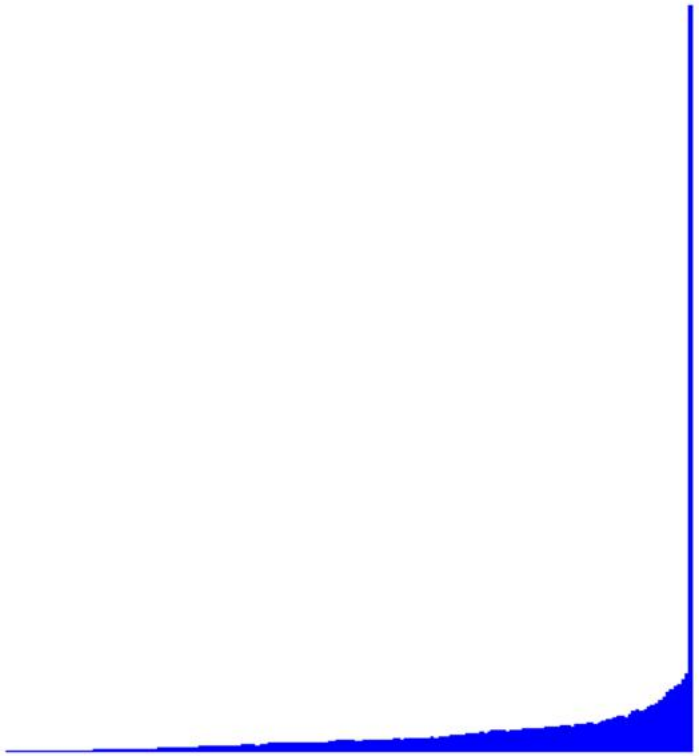




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

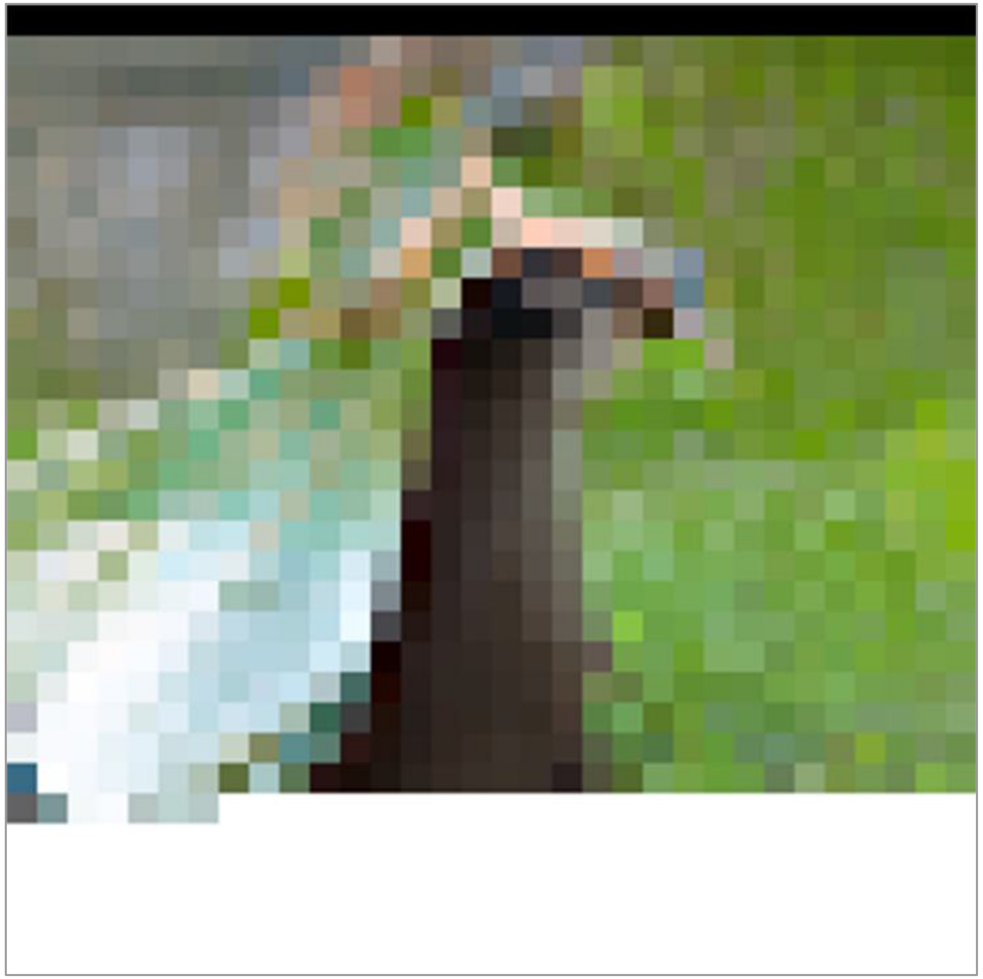
PixelCNN – Softmax Sampling

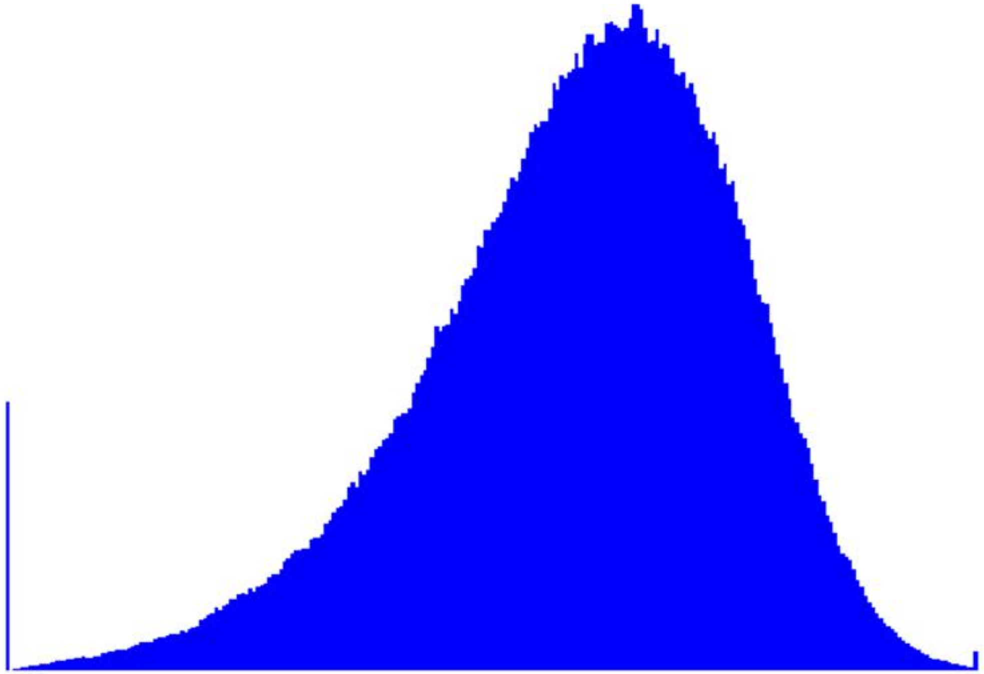




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

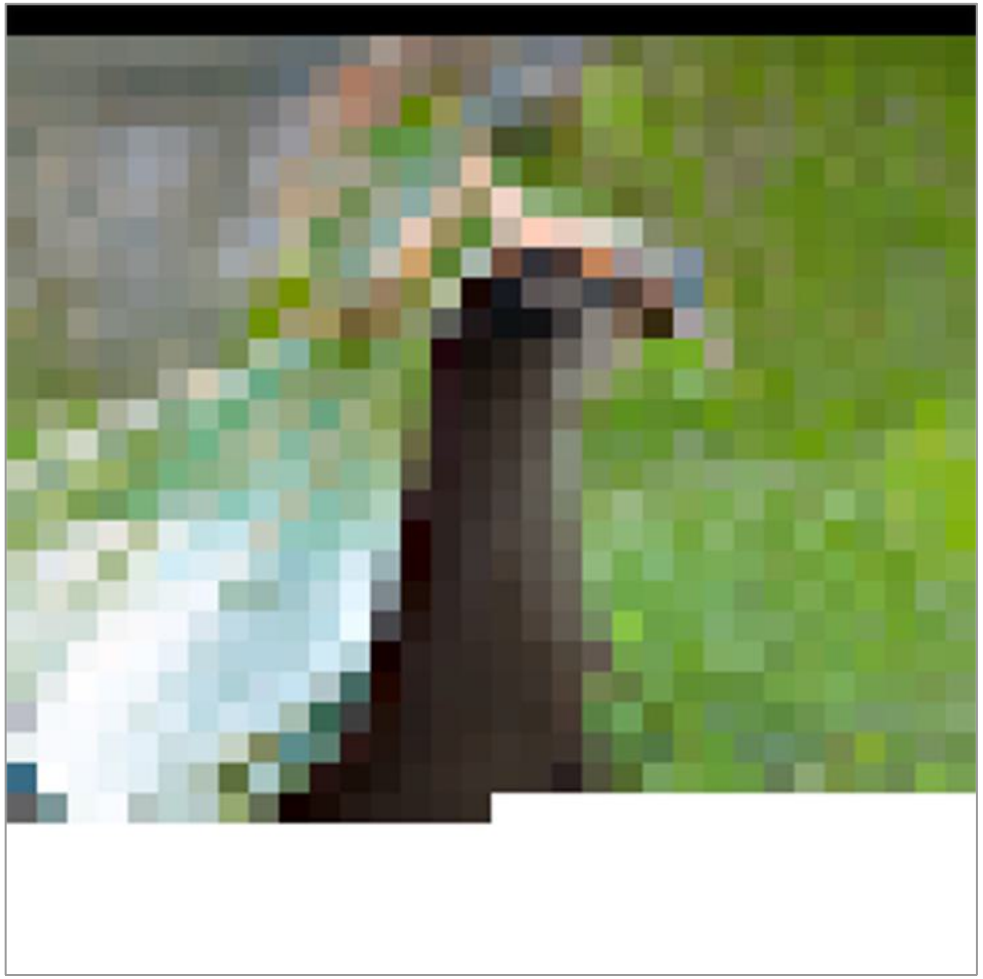
PixelCNN – Softmax Sampling

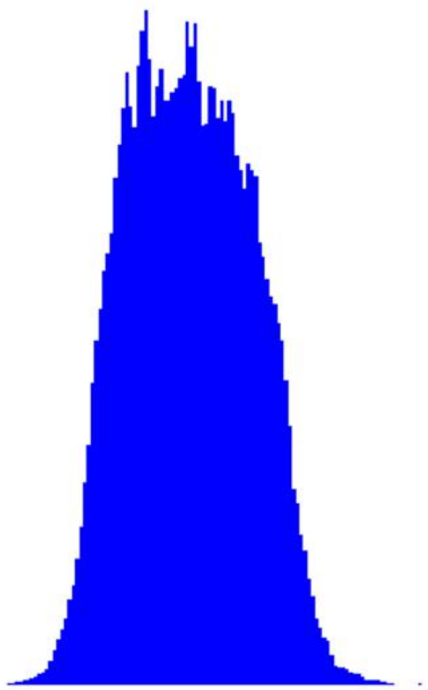




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

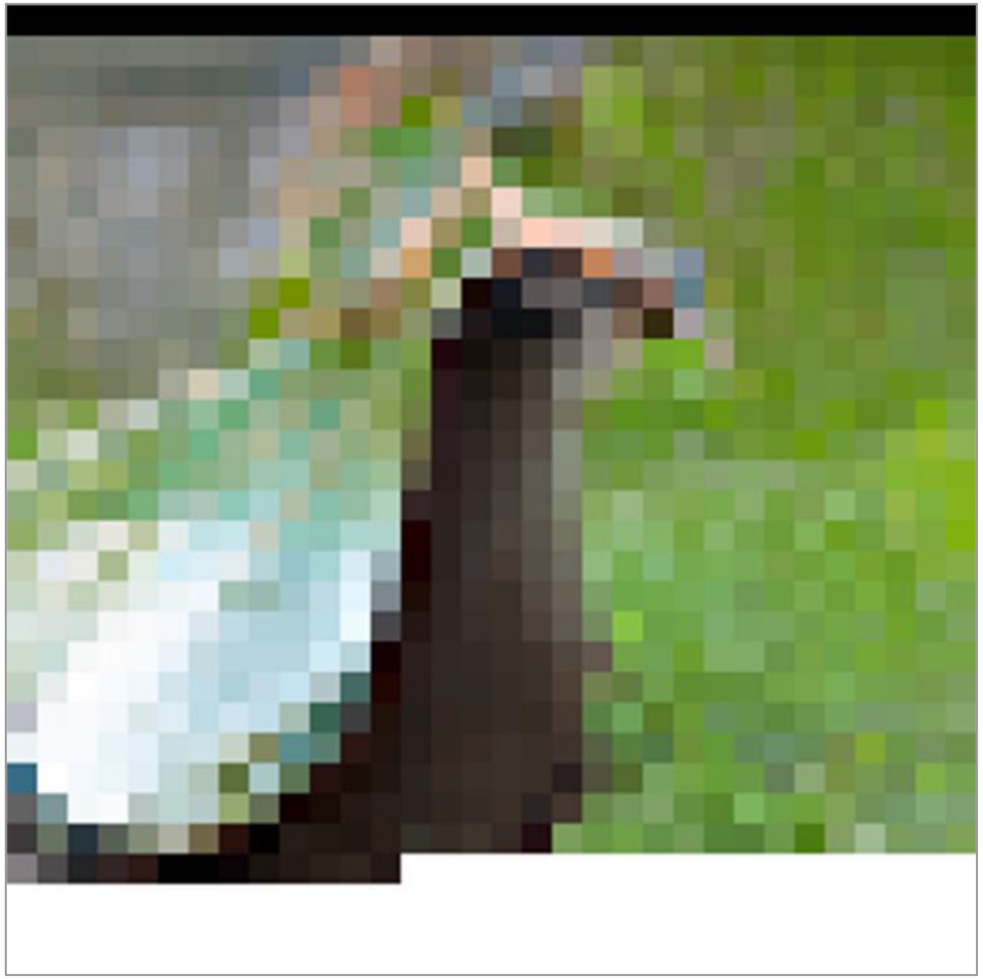
PixelCNN – Softmax Sampling

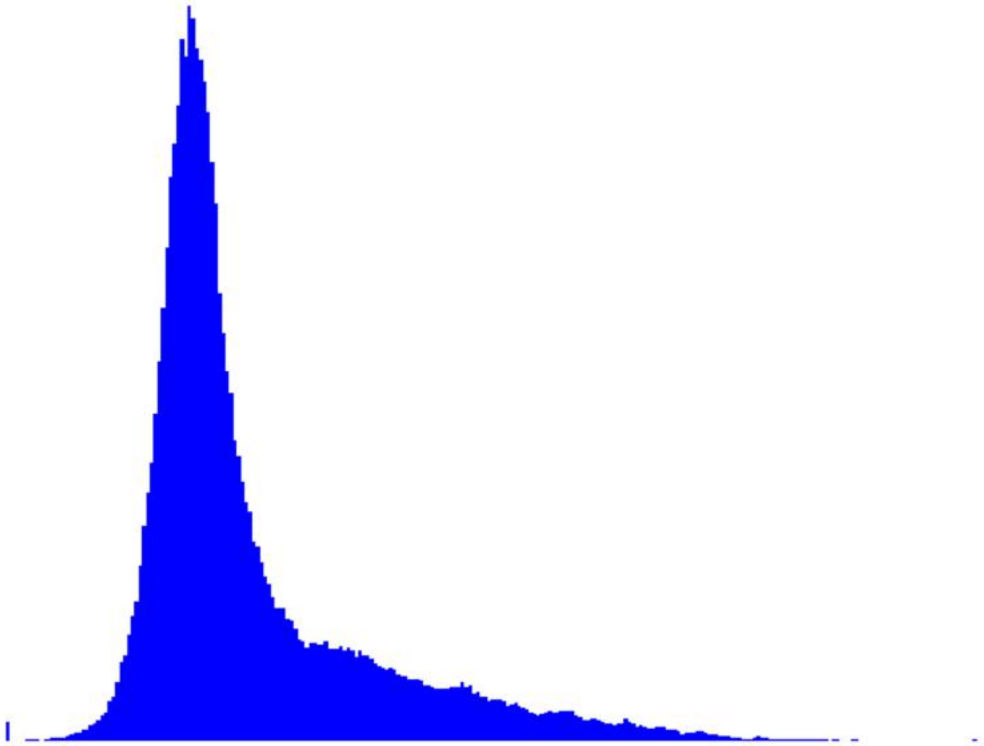




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

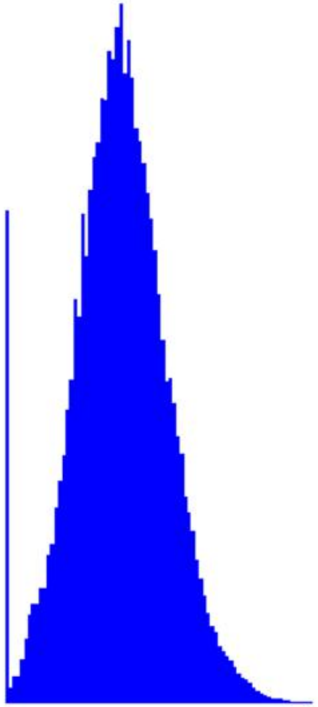
PixelCNN – Softmax Sampling





Slide adapted from
Oriol Vinyals and Navdeep Jaitly

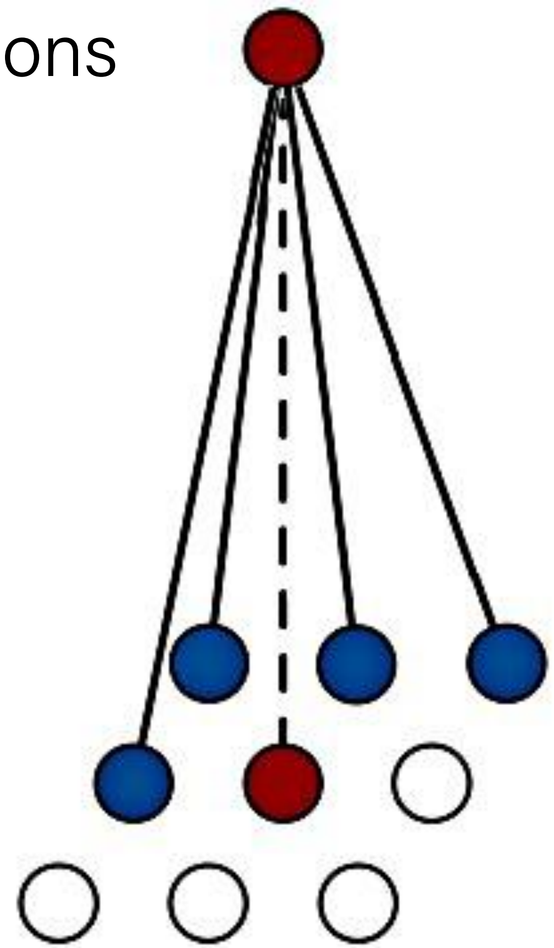
PixelCNN – Softmax Sampling

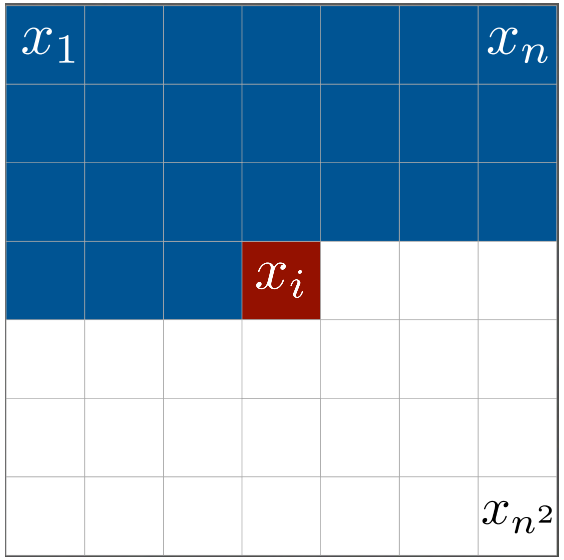


Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN

use masked convolutions to enforce the autoregressive relationship

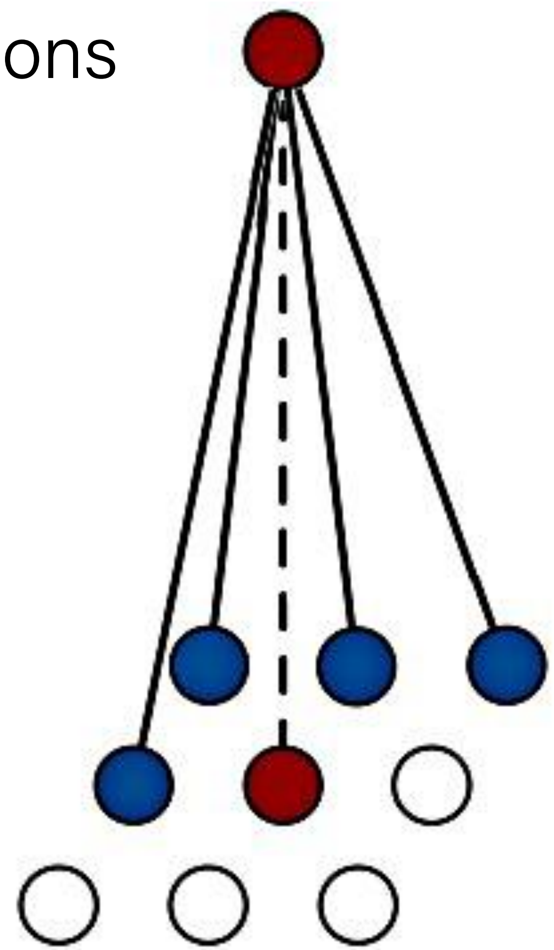


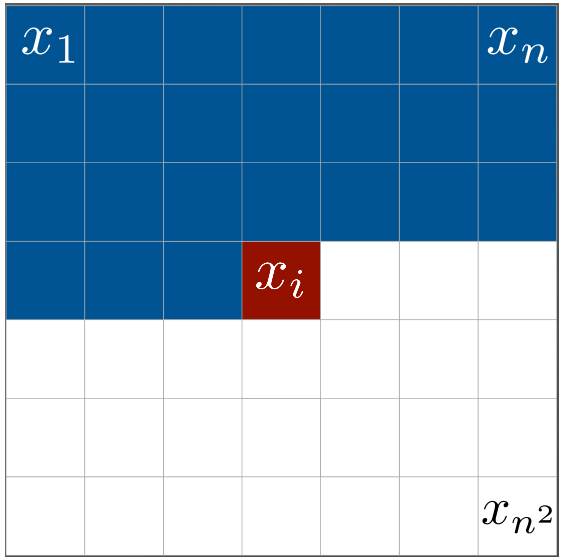


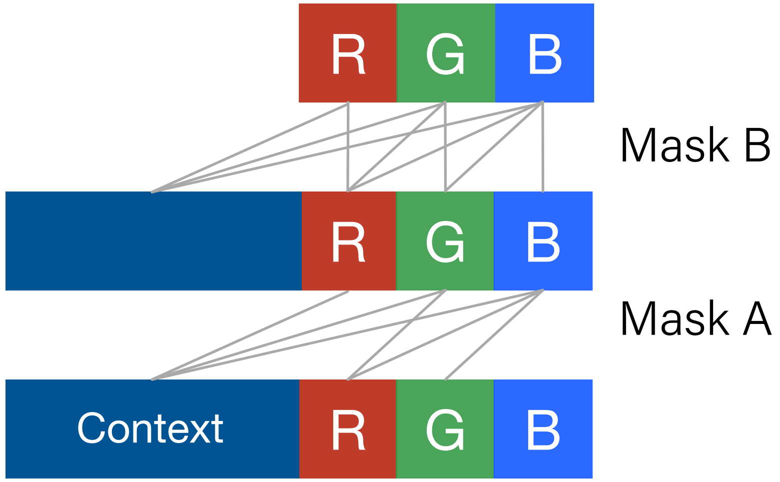
1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

PixelCNN

use masked convolutions to enforce the autoregressive relationship







autoregressive over color channels

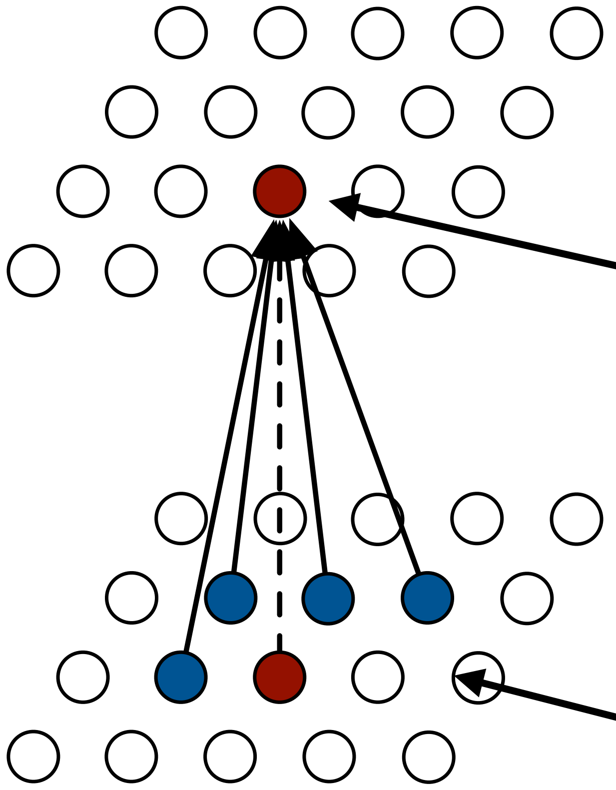
$$p(x_i \mid \mathbf{x}_{<i}) = p(x_{i,R} \mid \mathbf{x}_{<i})p(x_{i,G} \mid x_{i,R}, \mathbf{x}_{<i})p(x_{i,B} \mid x_{i,R}, x_{i,G}, \mathbf{x}_{<i})$$

Slide adapted from Nal Kalchbrenner

PixelCNN

Multiple layers of masked convolutions

composing multiple layers increases the context size



only depends on pixel above and to the left

masked convolution

Samples from PixelCNN

Topics: CIFAR-10

- Samples from a class-conditioned PixelCNN

Coral Reef

Samples from PixelCNN

Topics: CIFAR-10

- Samples from a class-conditioned PixelCNN

Sorrel horse

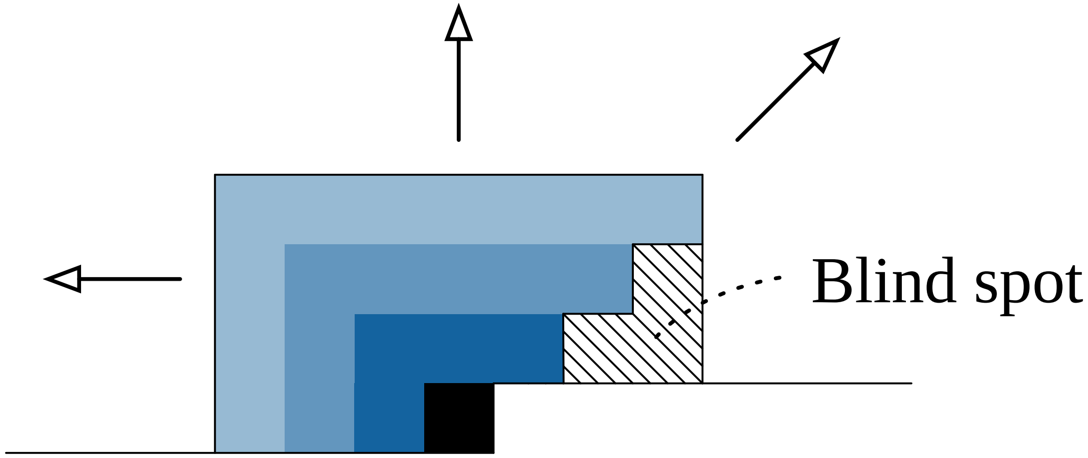
Samples from PixelCNN

Topics: CIFAR-10

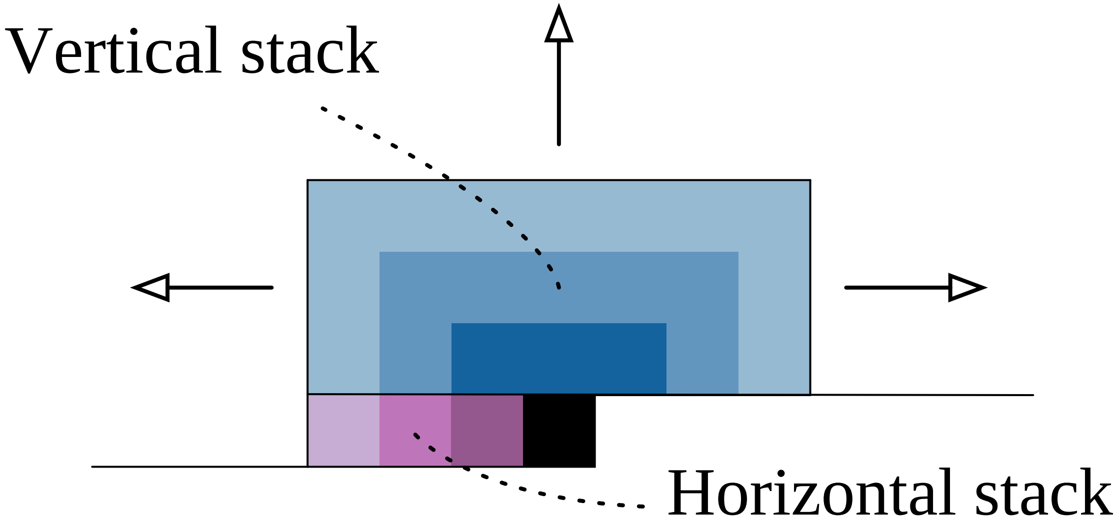
- Samples from a class-conditioned PixelCNN

Sandbar

Improving PixelCNN



Stacking layers of masked convolution creates a blindspot

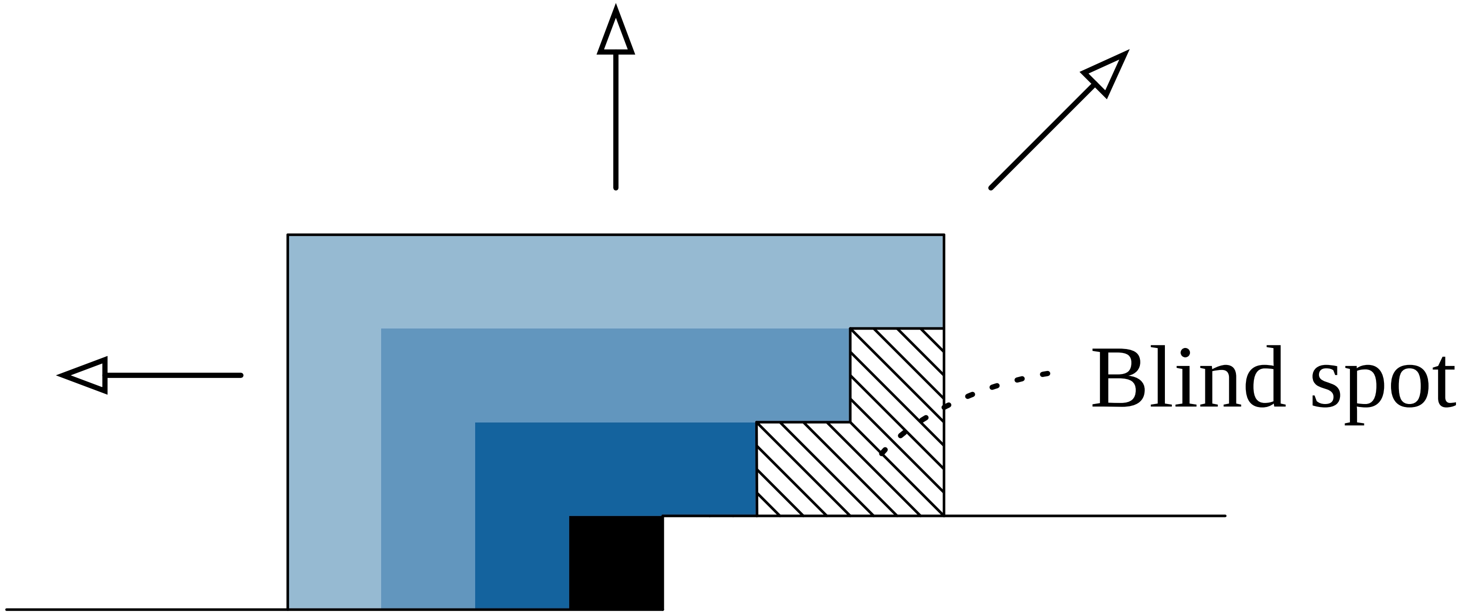


Solution: use two stacks of convolution, a vertical stack and a horizontal stack

Improving PixelCNN I

There is a problem with this form of masked convolution.

1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

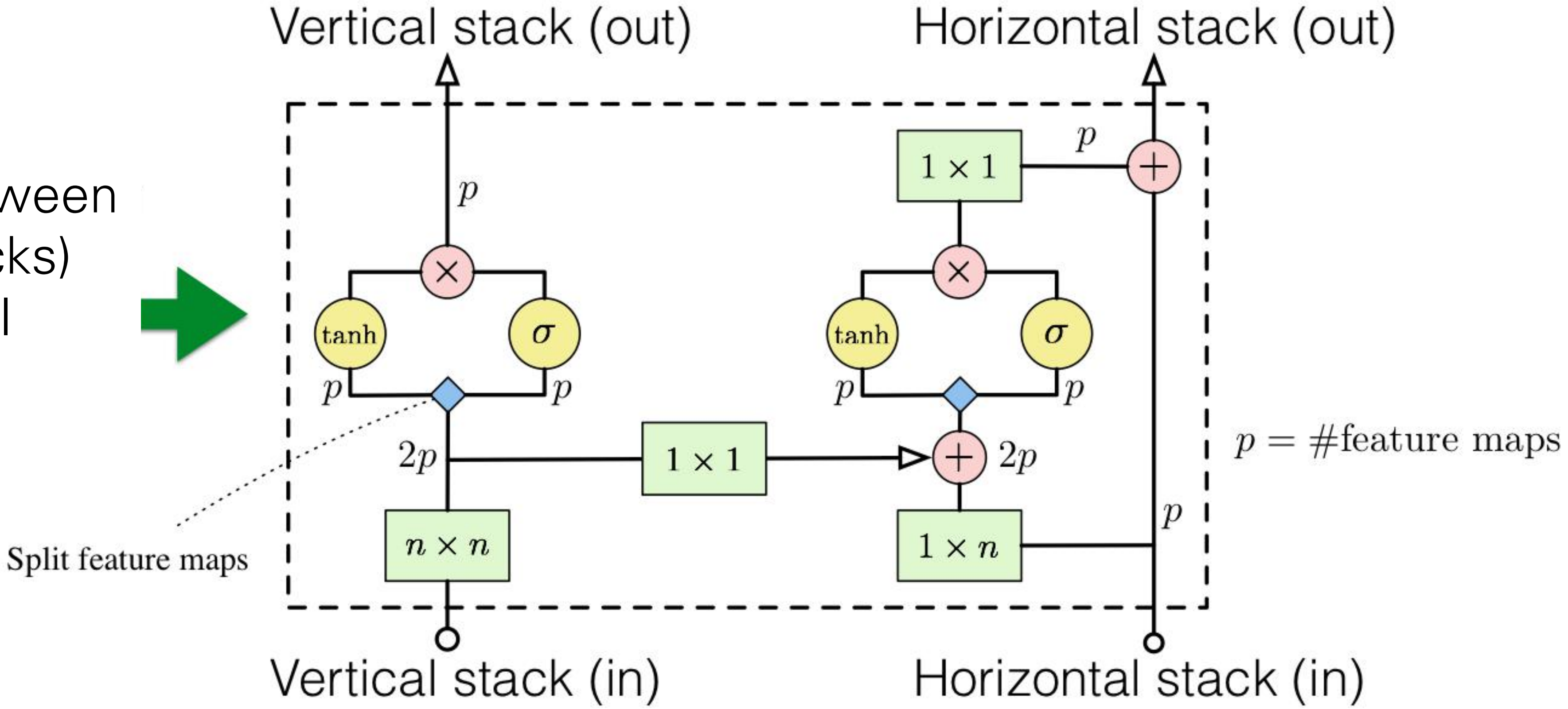


Stacking layers of masked convolution creates a blindspot

Improving PixelCNN II

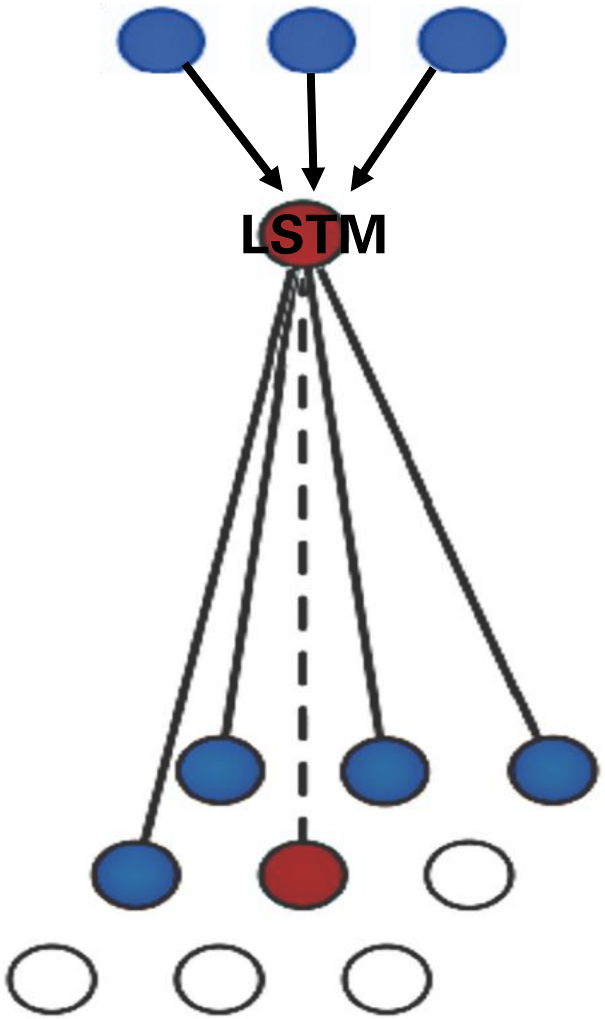
Use more expressive nonlinearity: $\mathbf{h}_{k+1} = \tanh(W_{k,f} * \mathbf{h}_k) \odot \sigma(W_{k,g} * \mathbf{h}_k)$

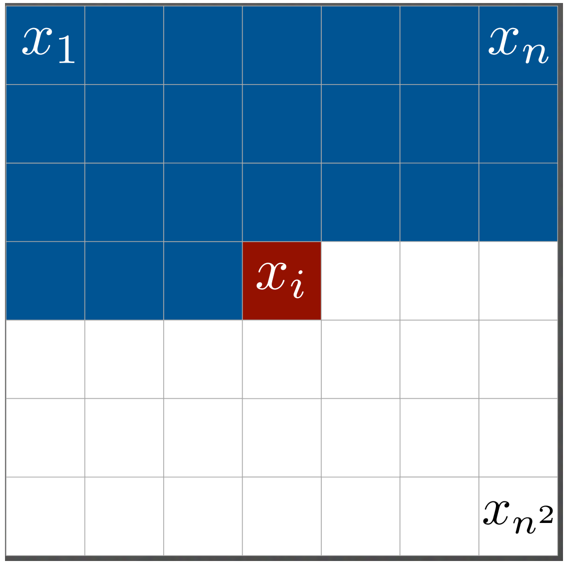
This information flow (between vertical and horizontal stacks) preserves the correct pixel dependencies



Convolutional Long Short-Term Memory

Row LSTM

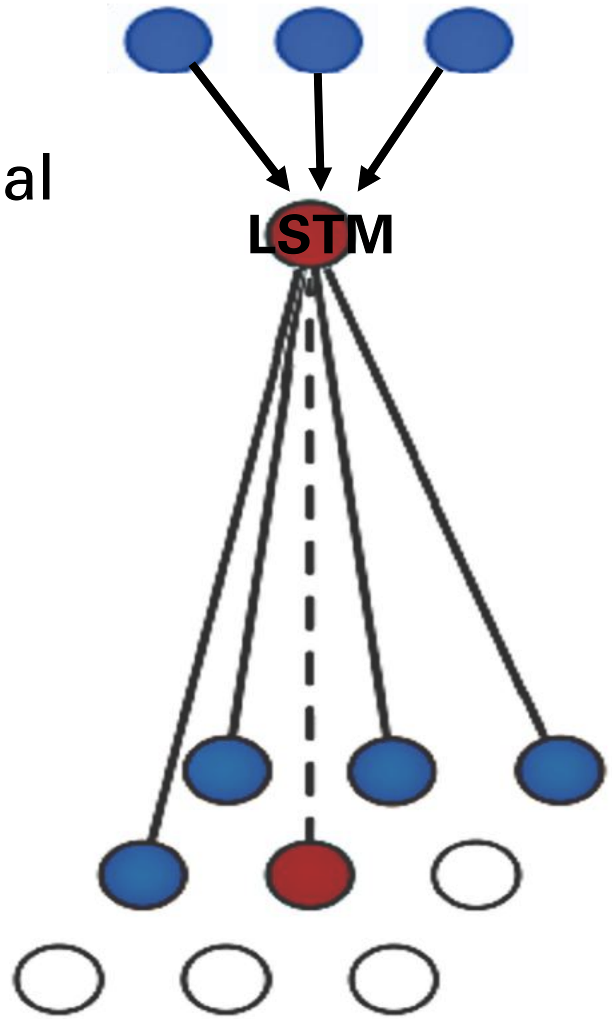


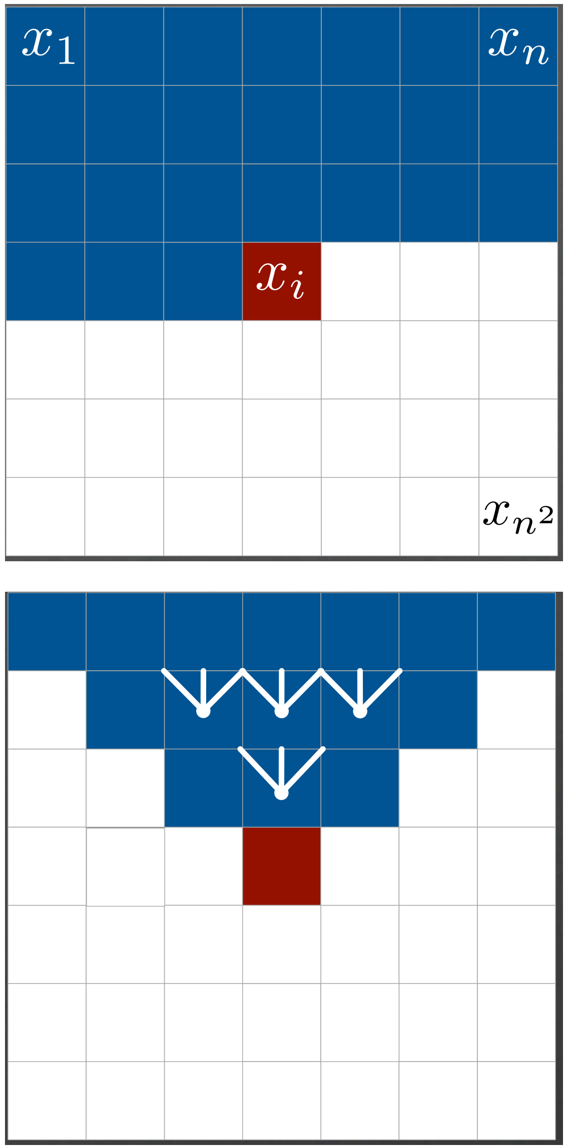


Stollenga et al, 2015
Oord, Kalchbrenner, Kavukcuoglu, 2016

Pixel RNN

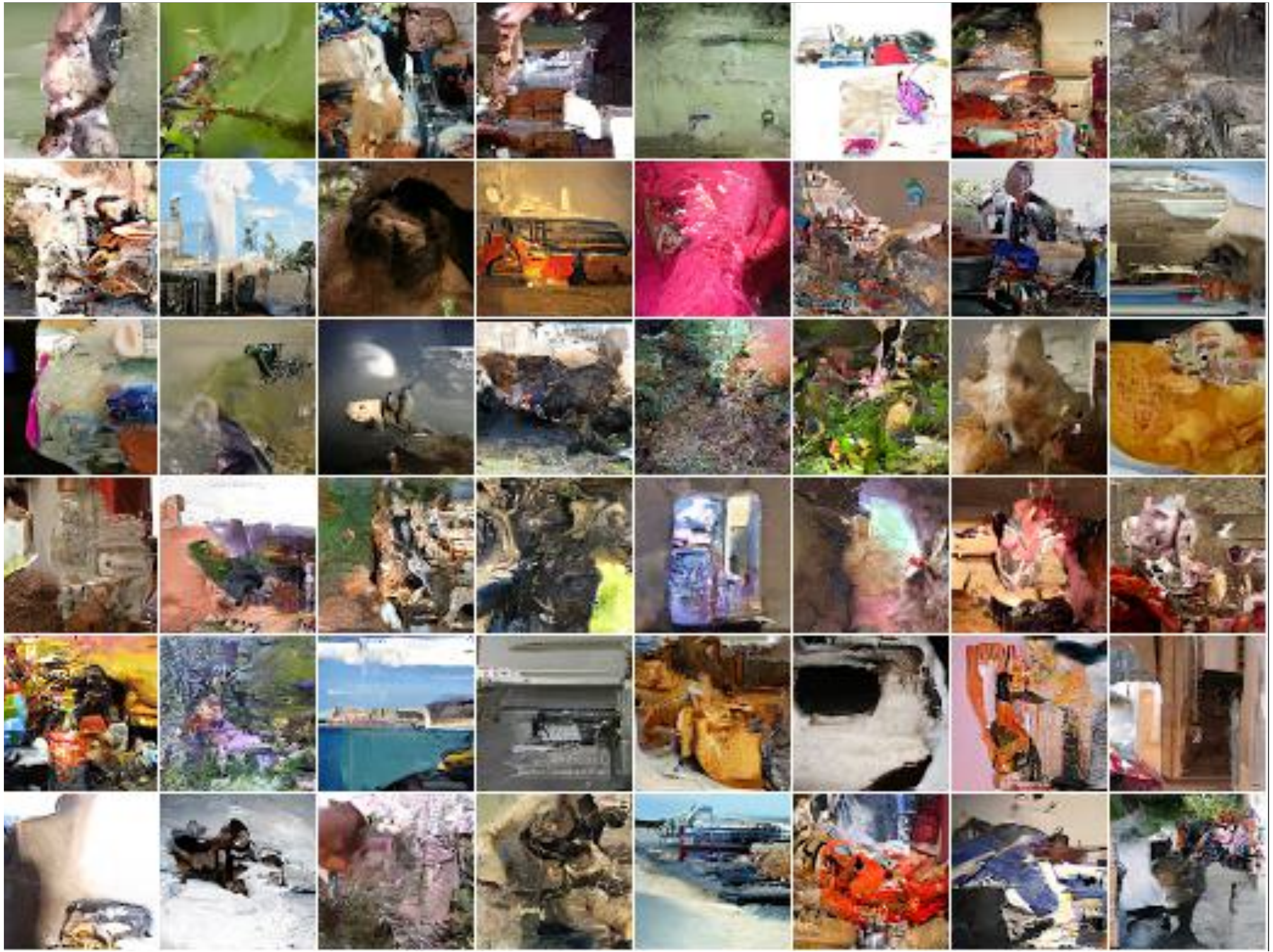
Multiple layers of convolutional LSTM





Oord, Kalchbrenner, Kavukcuoglu, 2016

Samples from PixelRNN



Slide credit:
Nal Kalchbrenner

Samples from PixelRNN

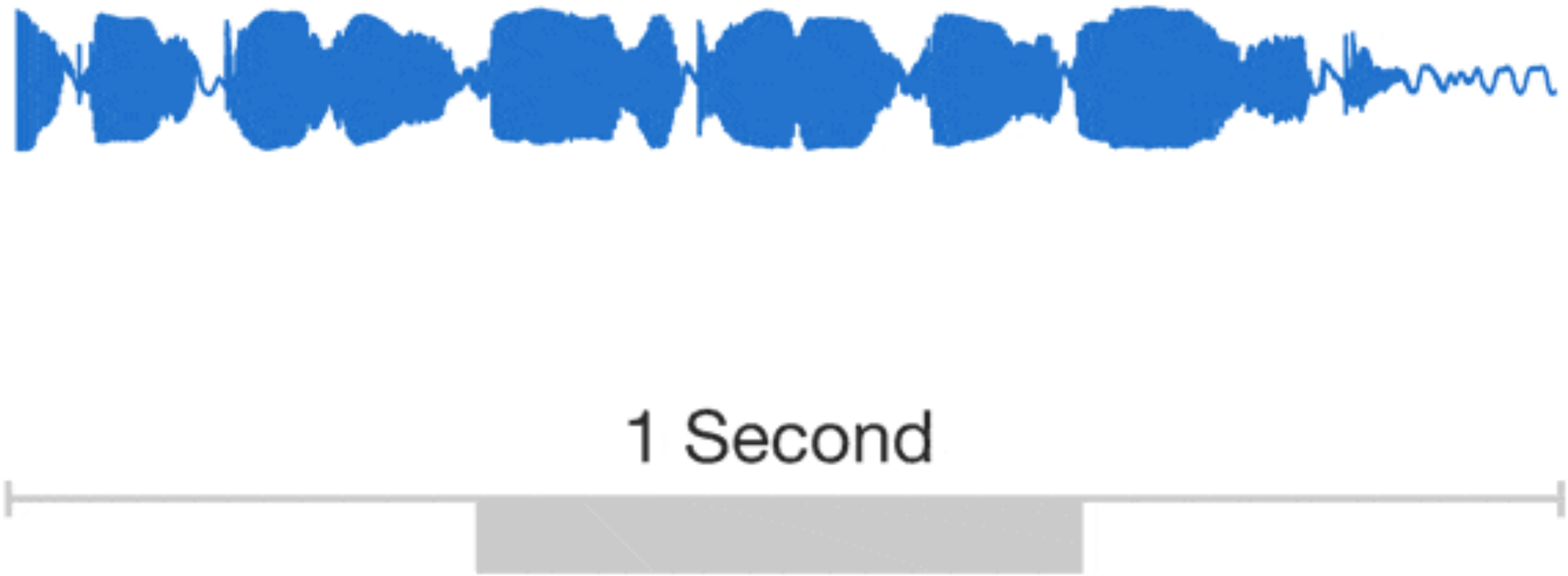
Slide credit: Nal Kalchbrenner

Image completions (conditional samples) from PixelRNN

occluded completions original

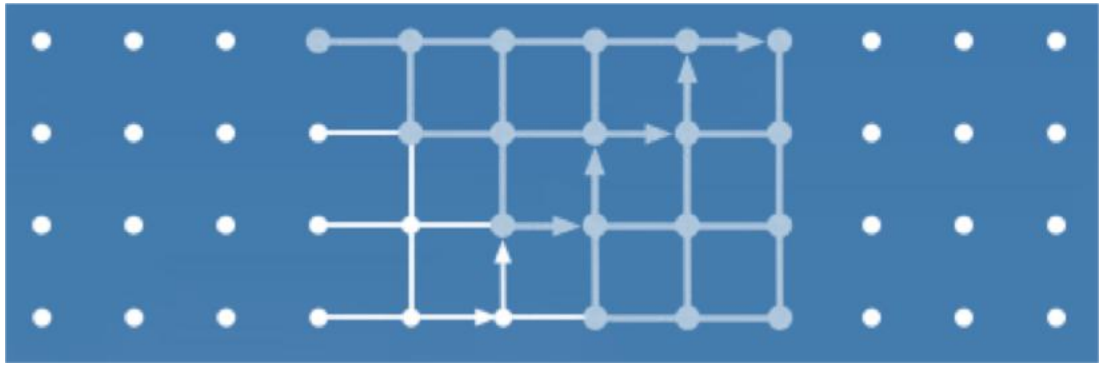
[PixelRNN, van der Oord et al. 2016]

Modeling Audio

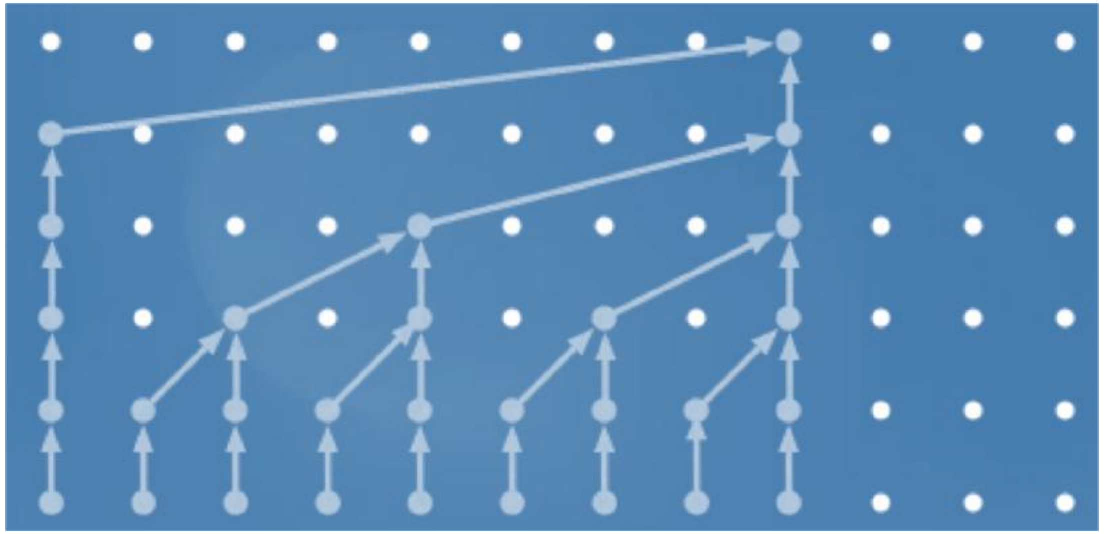


Architecture for 1D sequences (Bytenet / Wavenet)

Deep RNN

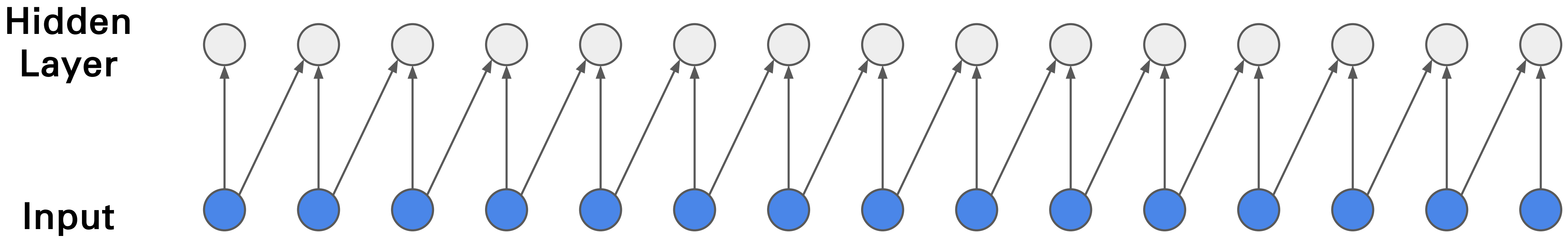


Bytenet decoder

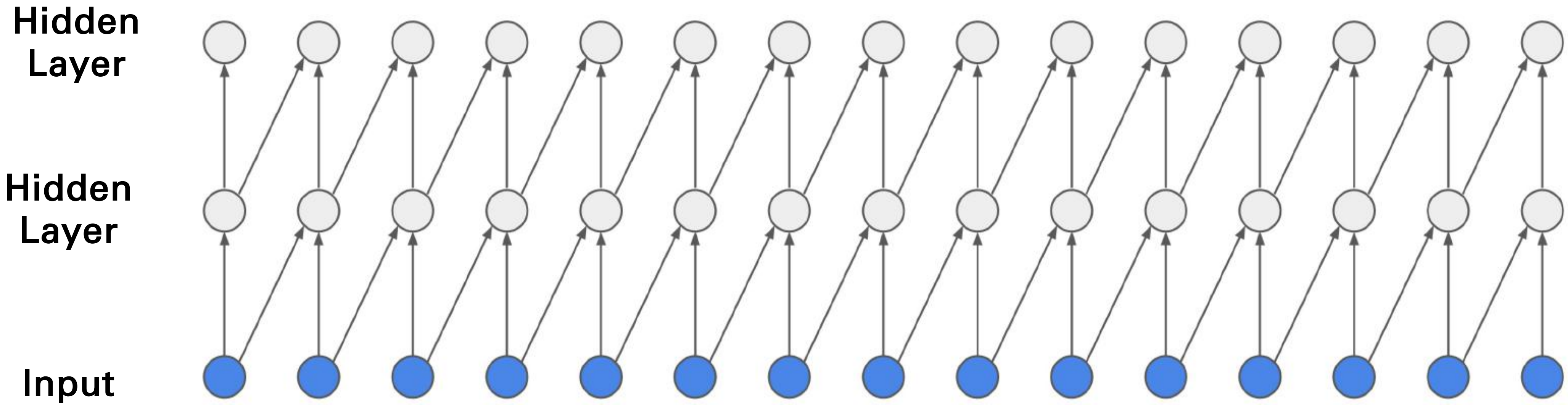


- Stack of **dilated, masked 1-D convolutions** in the decoder
- The architecture is **parallelizable** along the time dimension (during training or scoring)
- Easy access to **many states** from the past

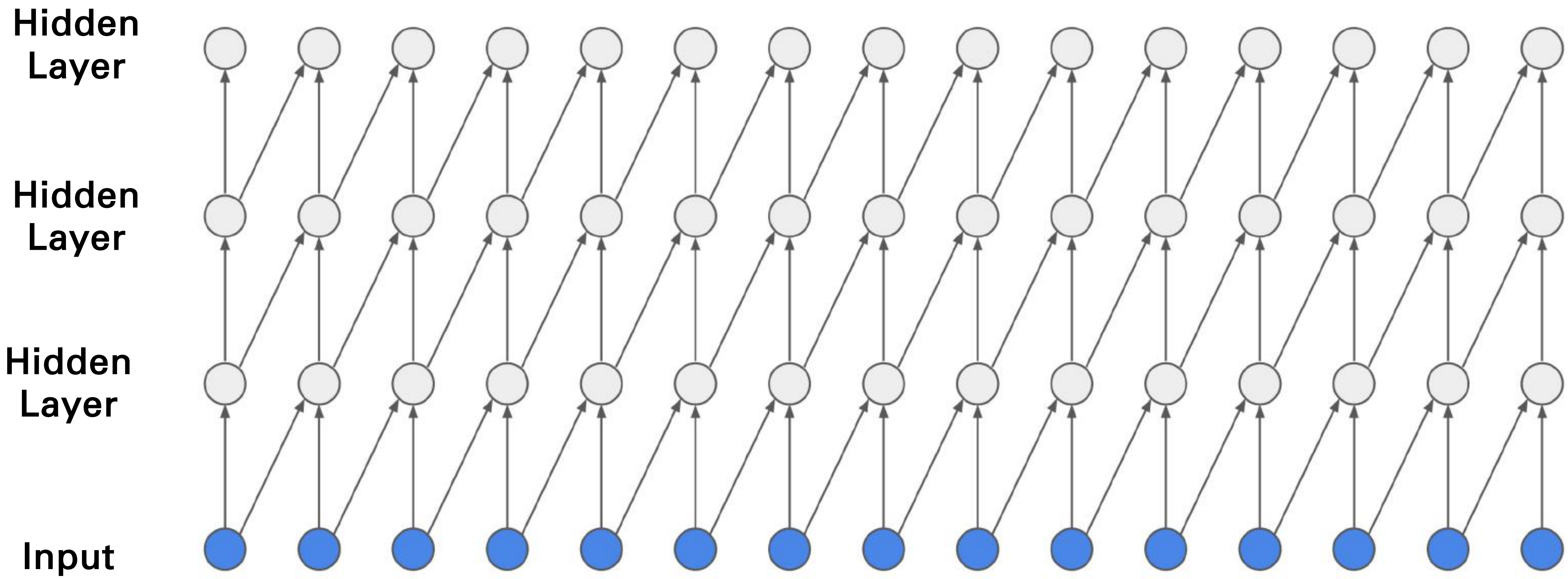
Causal Convolution



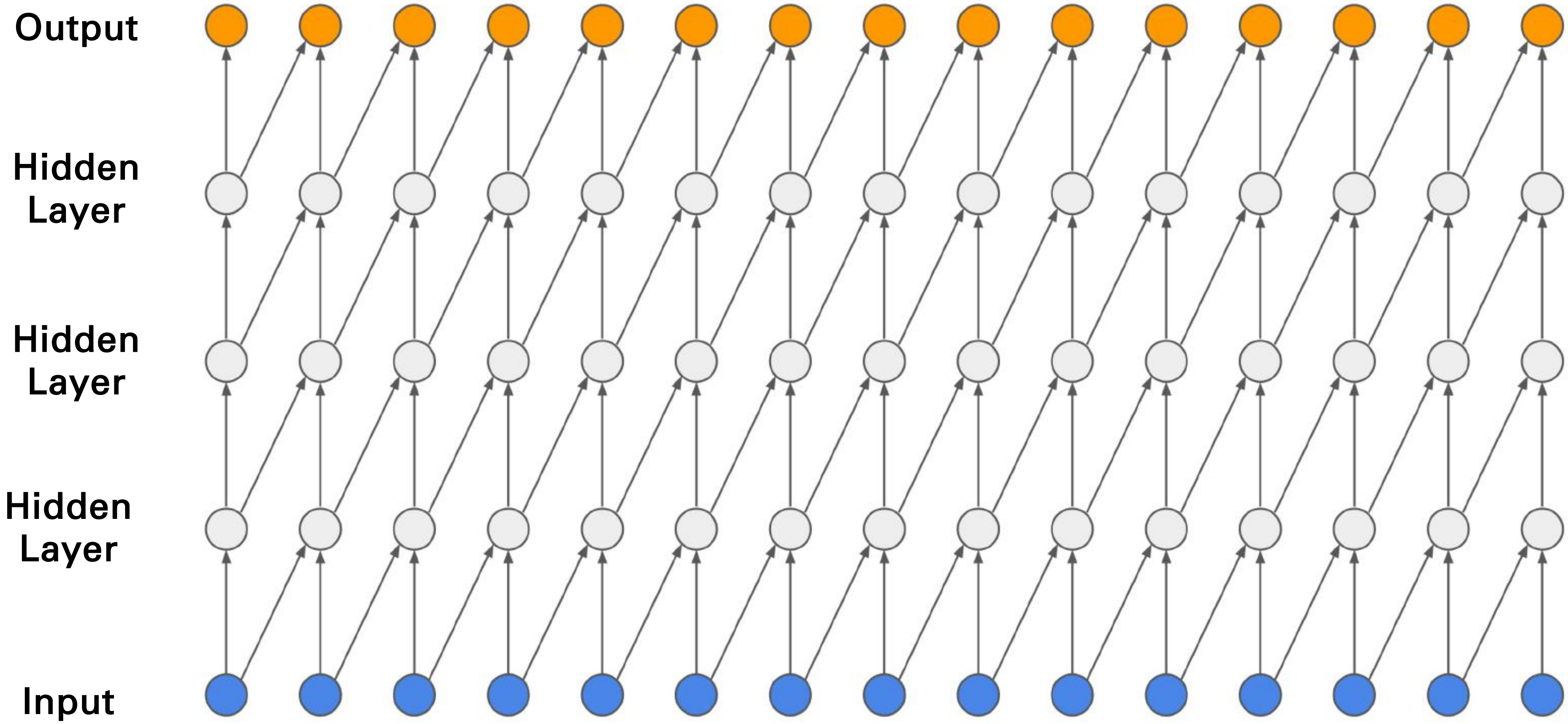
Causal Convolution



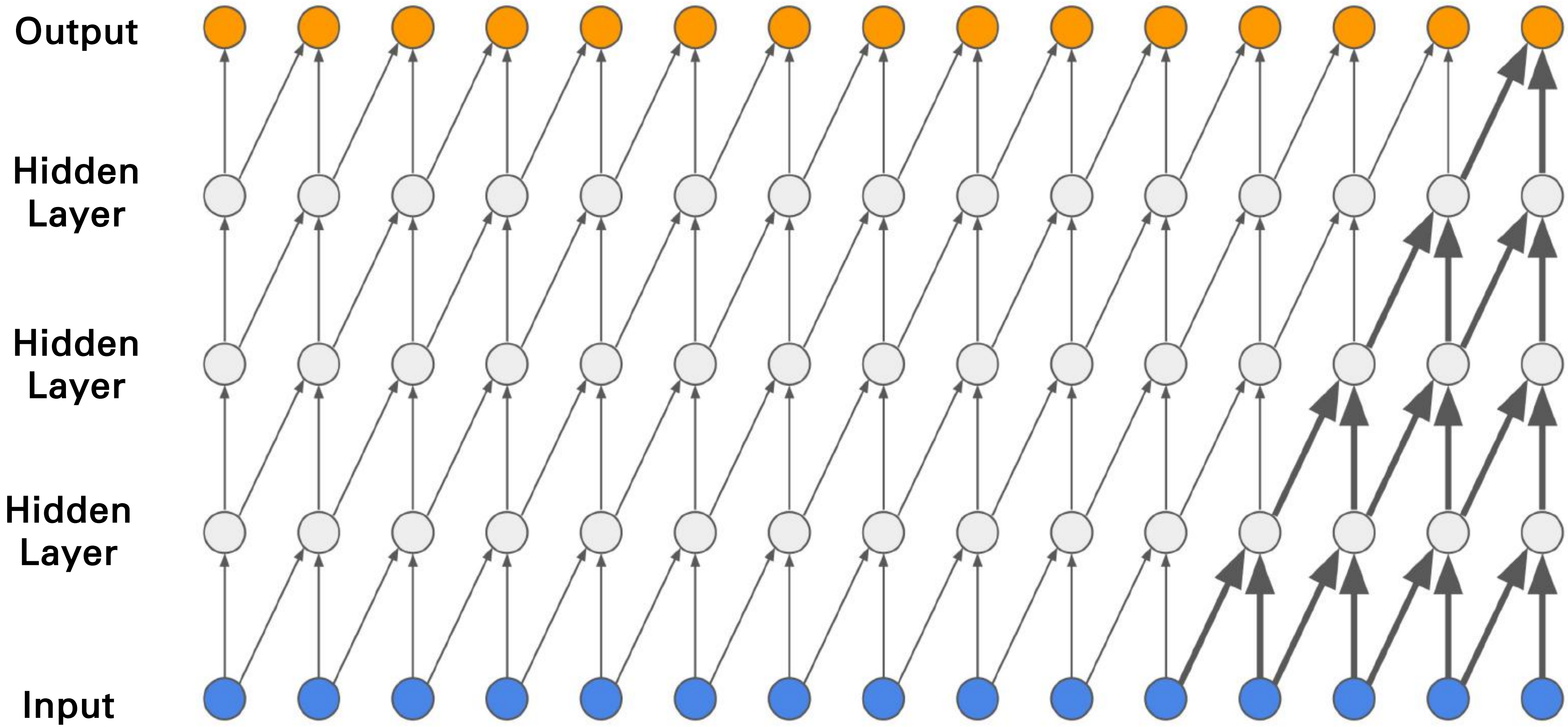
Causal Convolution



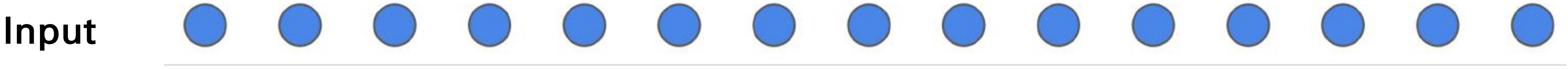
Causal Convolution



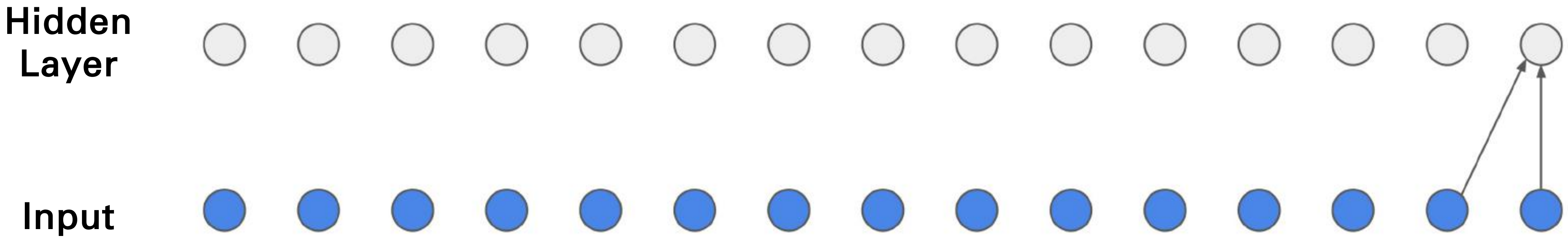
Causal Convolution



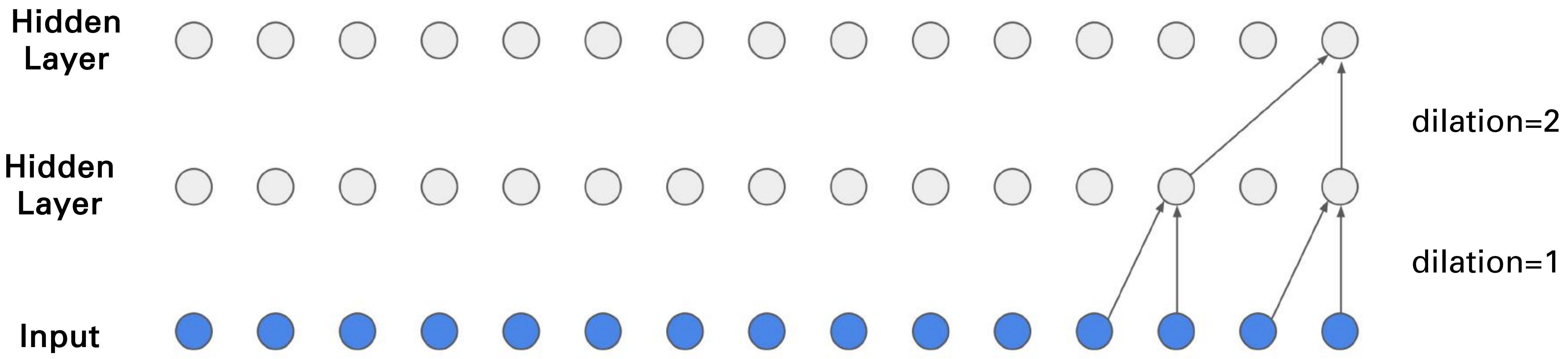
Causal Dilated Convolution



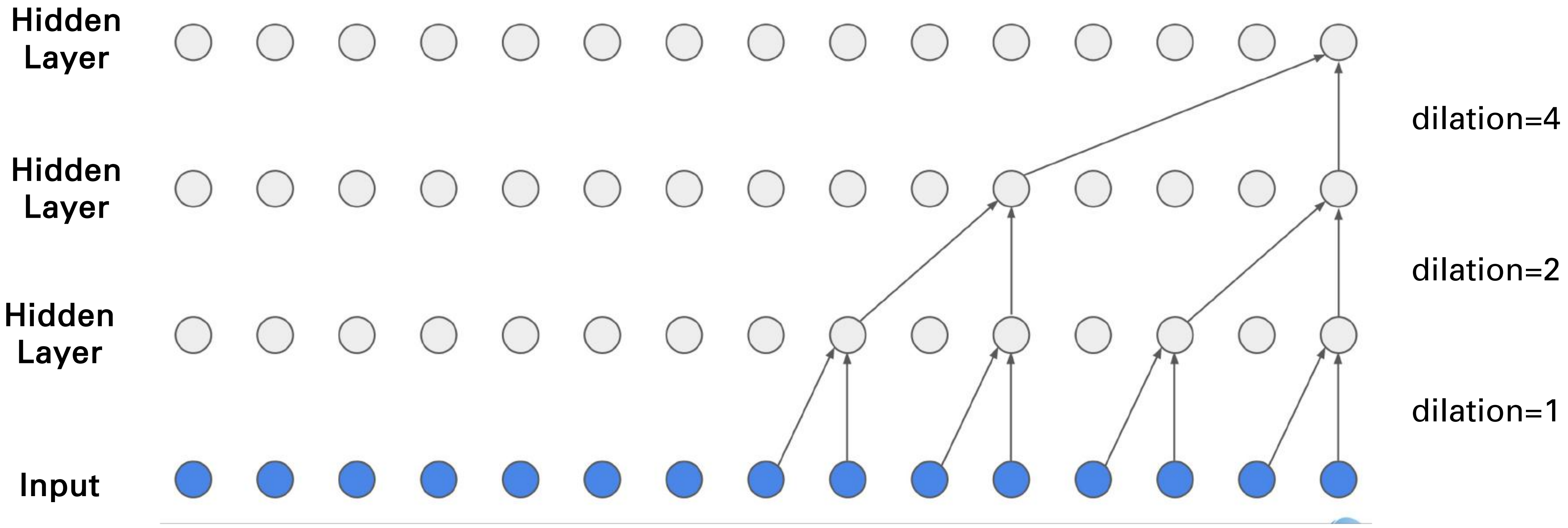
Causal Dilated Convolution



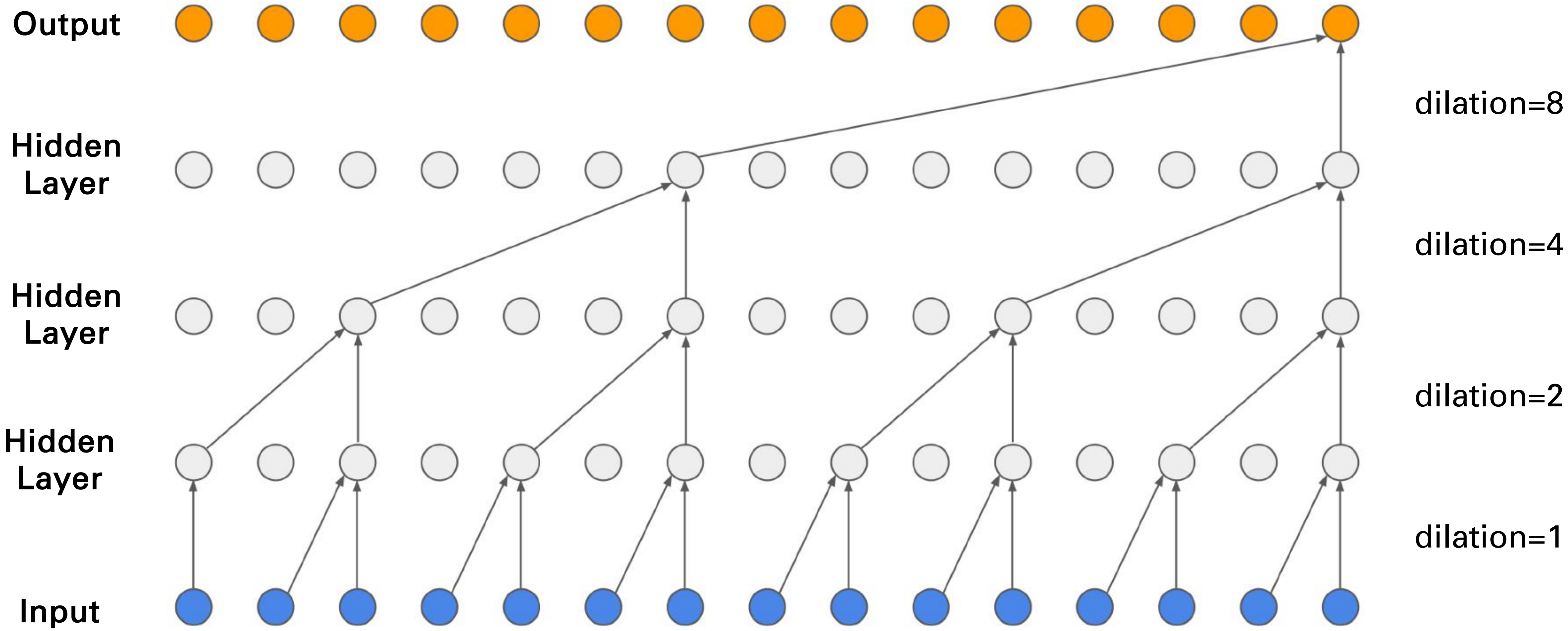
Causal Dilated Convolution



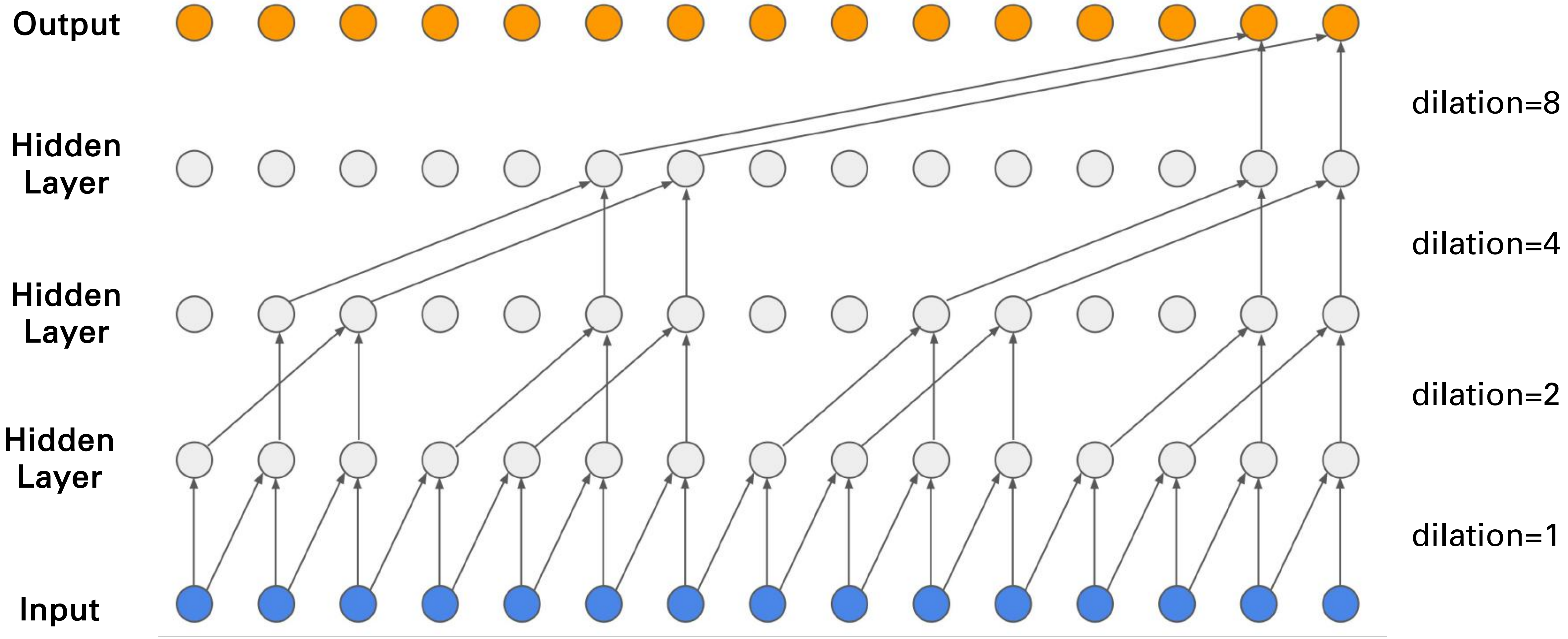
Causal Dilated Convolution



Causal Dilated Convolution

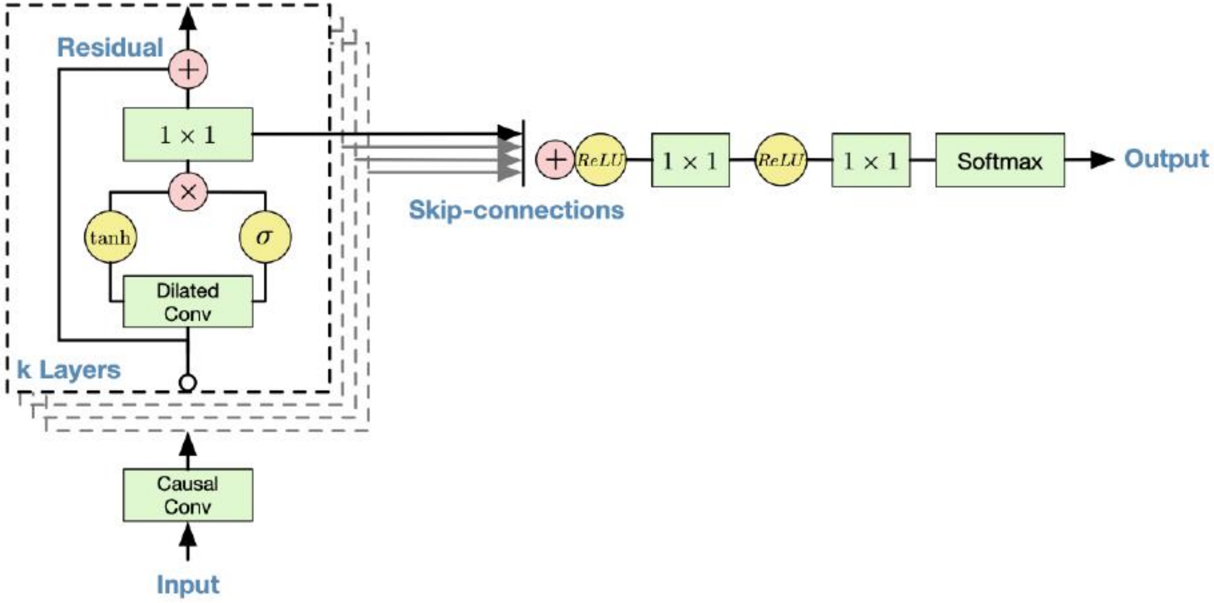


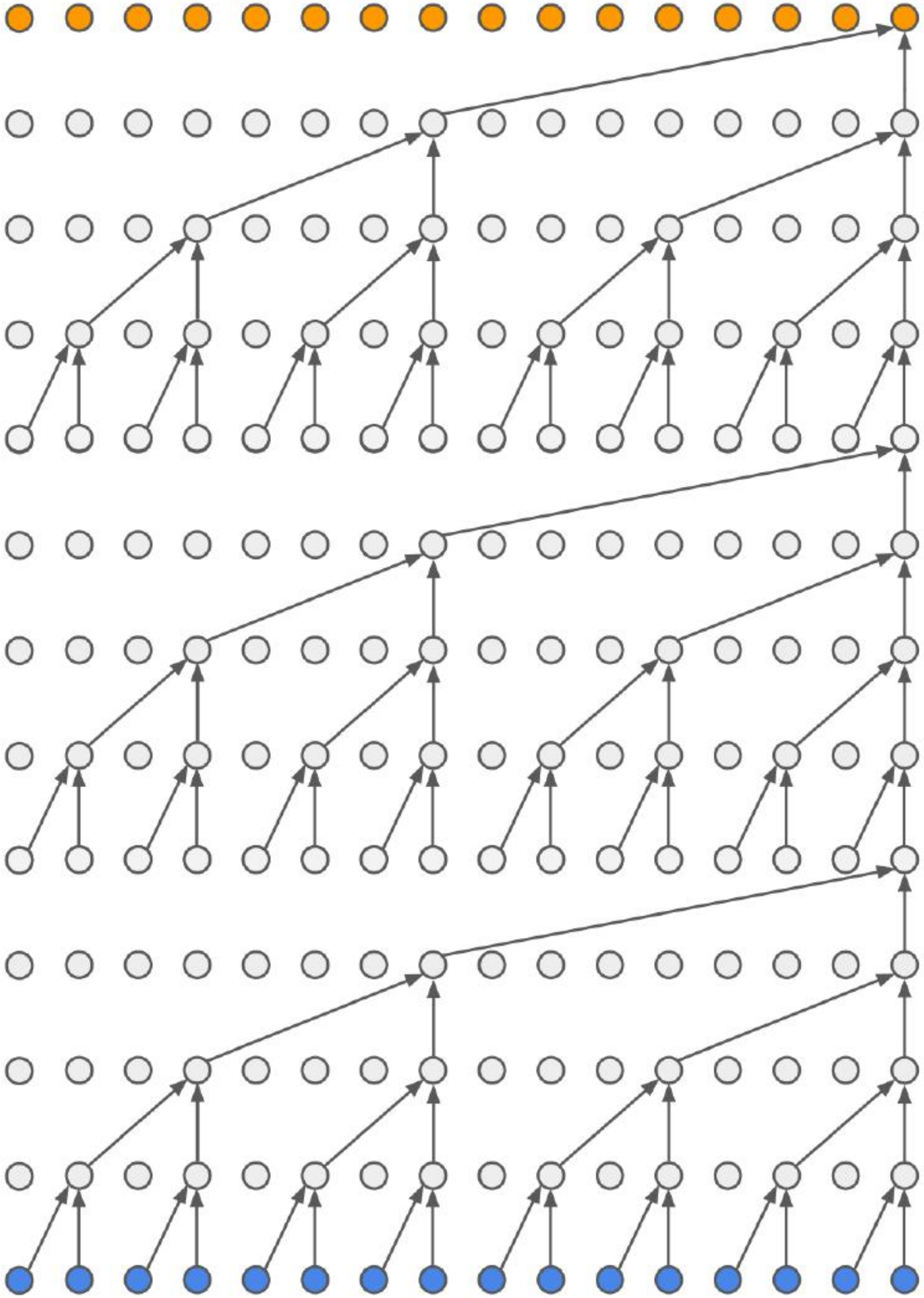
Causal Dilated Convolution



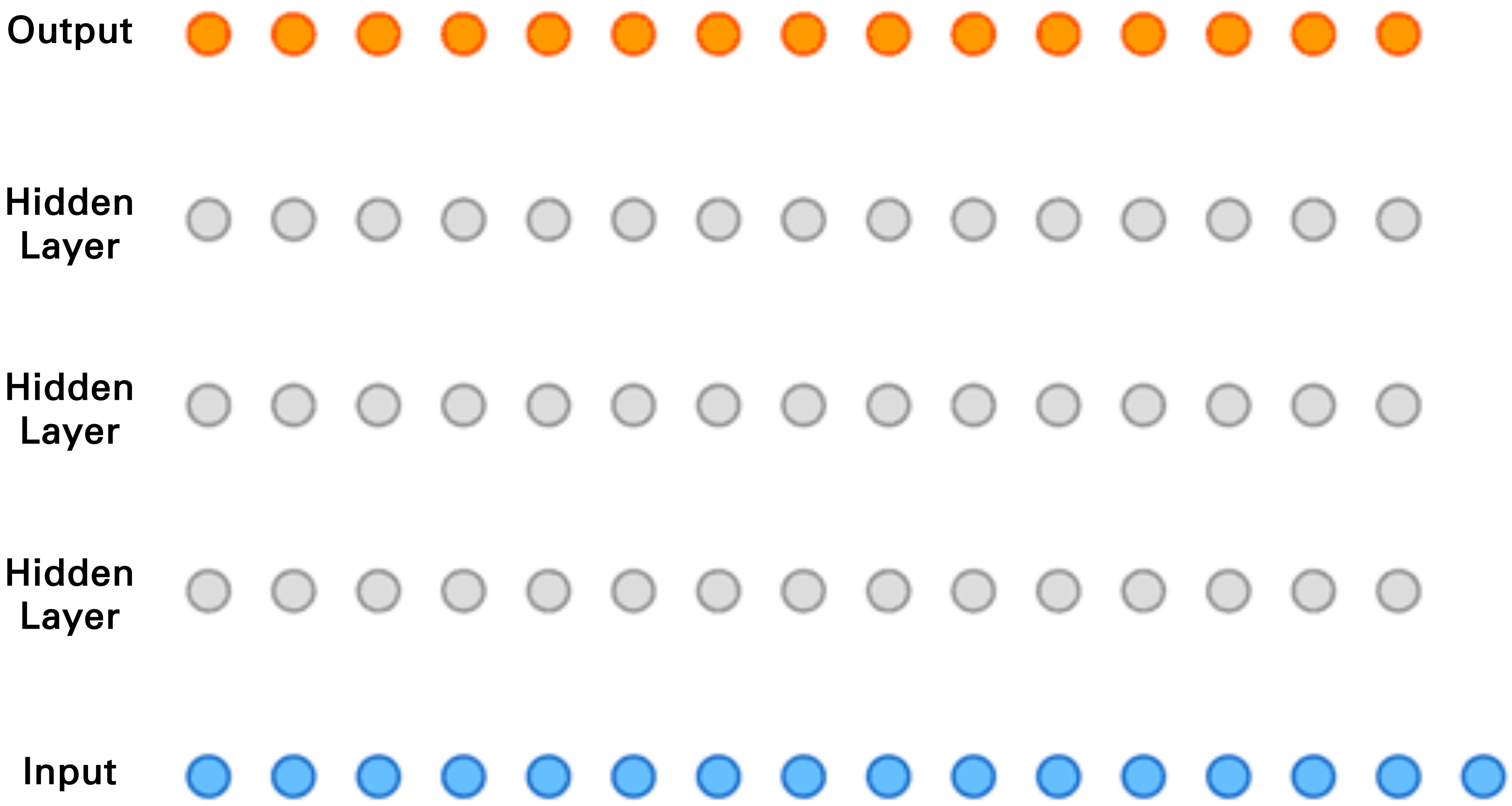
Multiple Stacks

- Improved receptive field with dilated convolutions
- Gated Residual block with skip connections





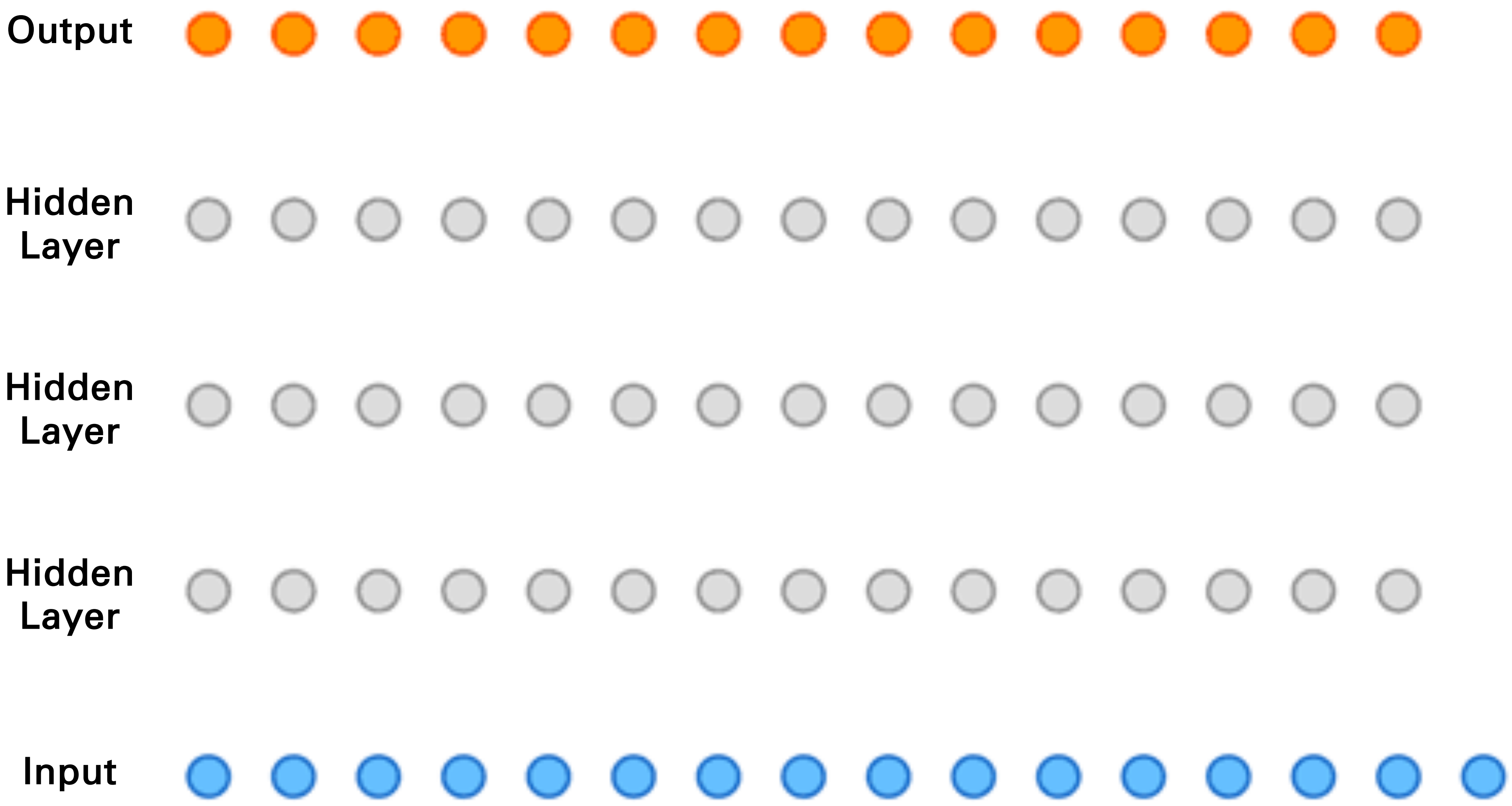
Sampling



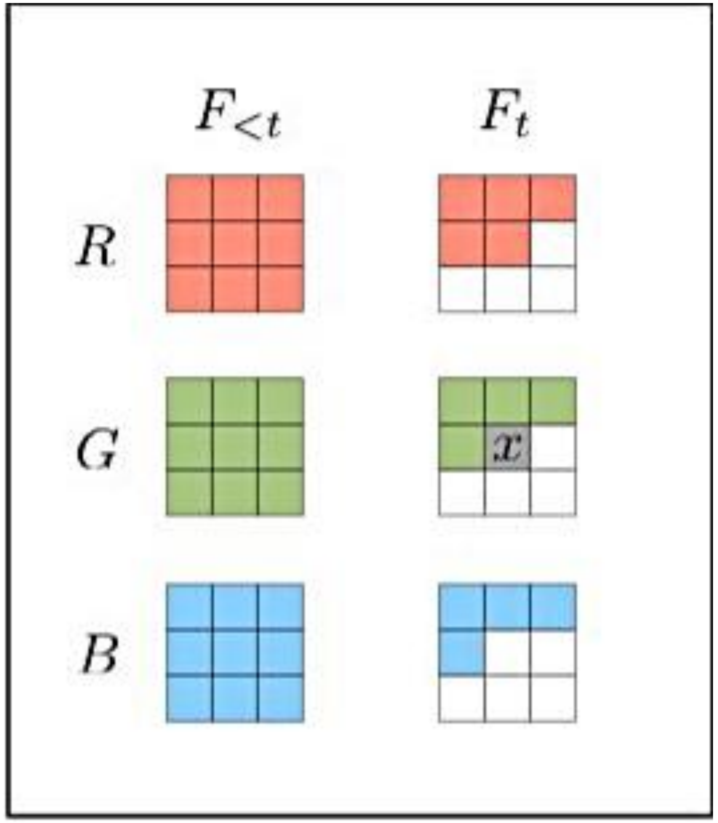
Sampling

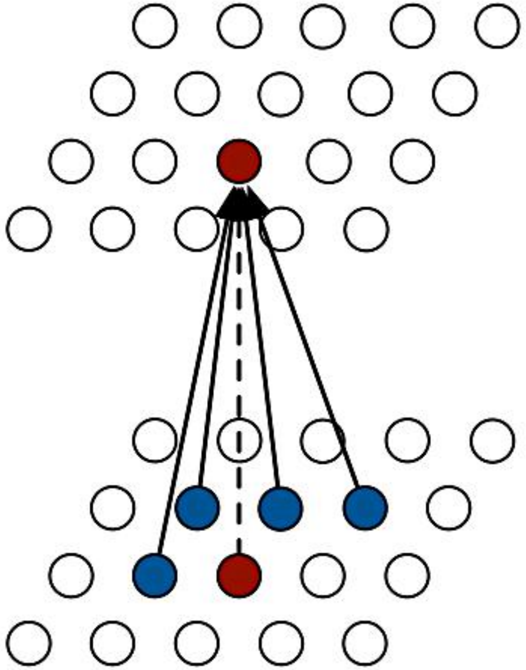
sample speech

sample music

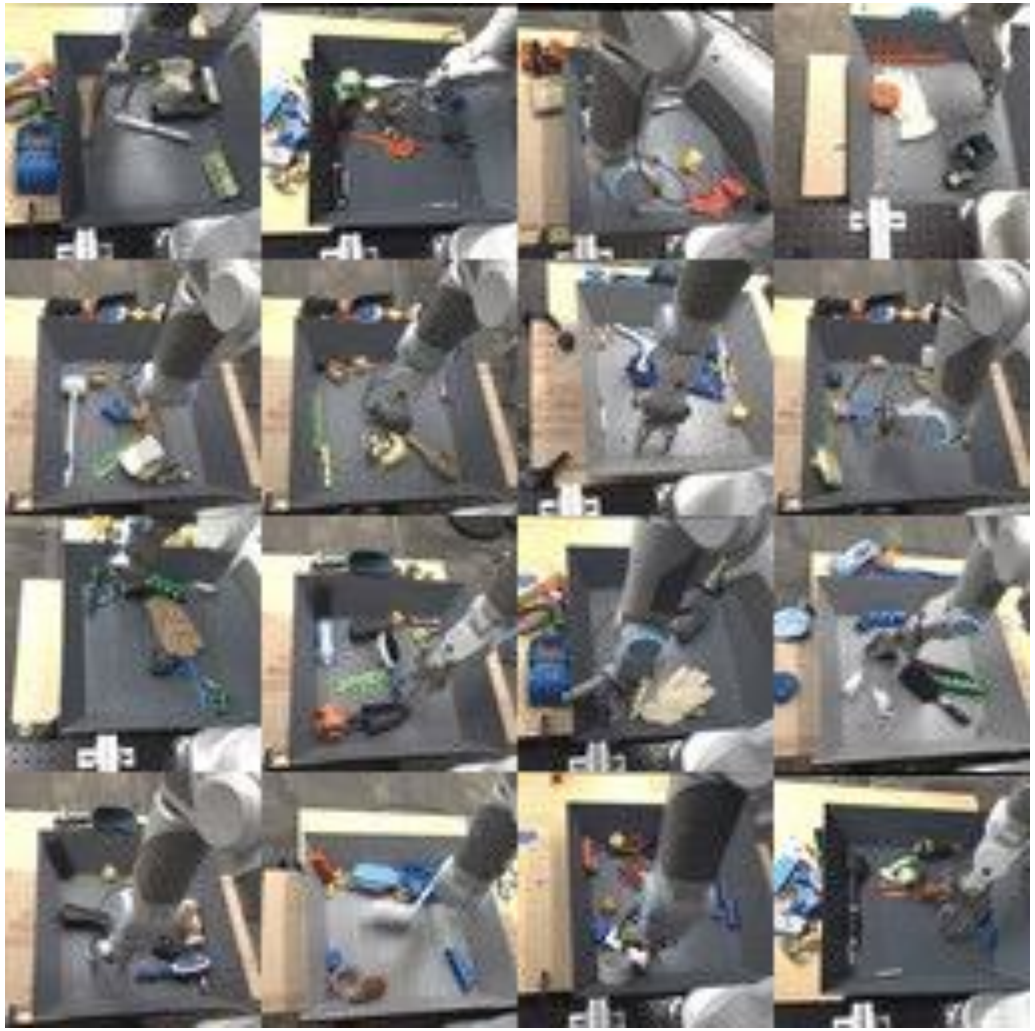


Video Pixel Net (VPN)



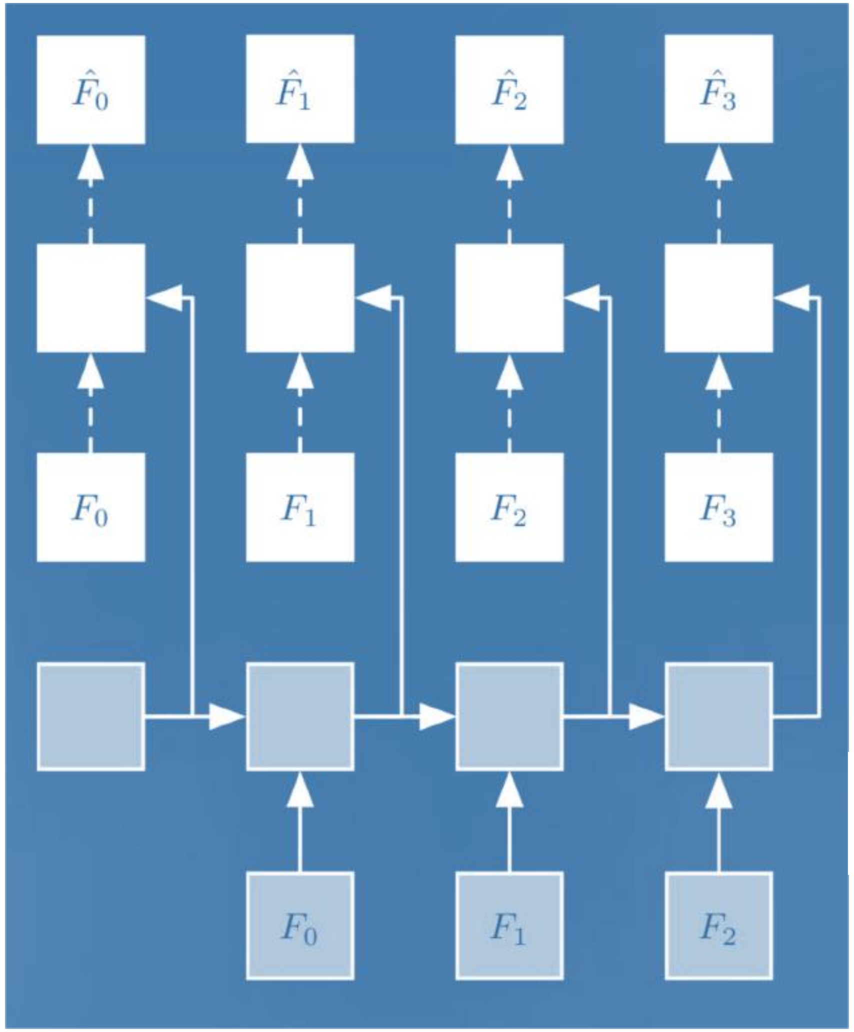


masked convolution



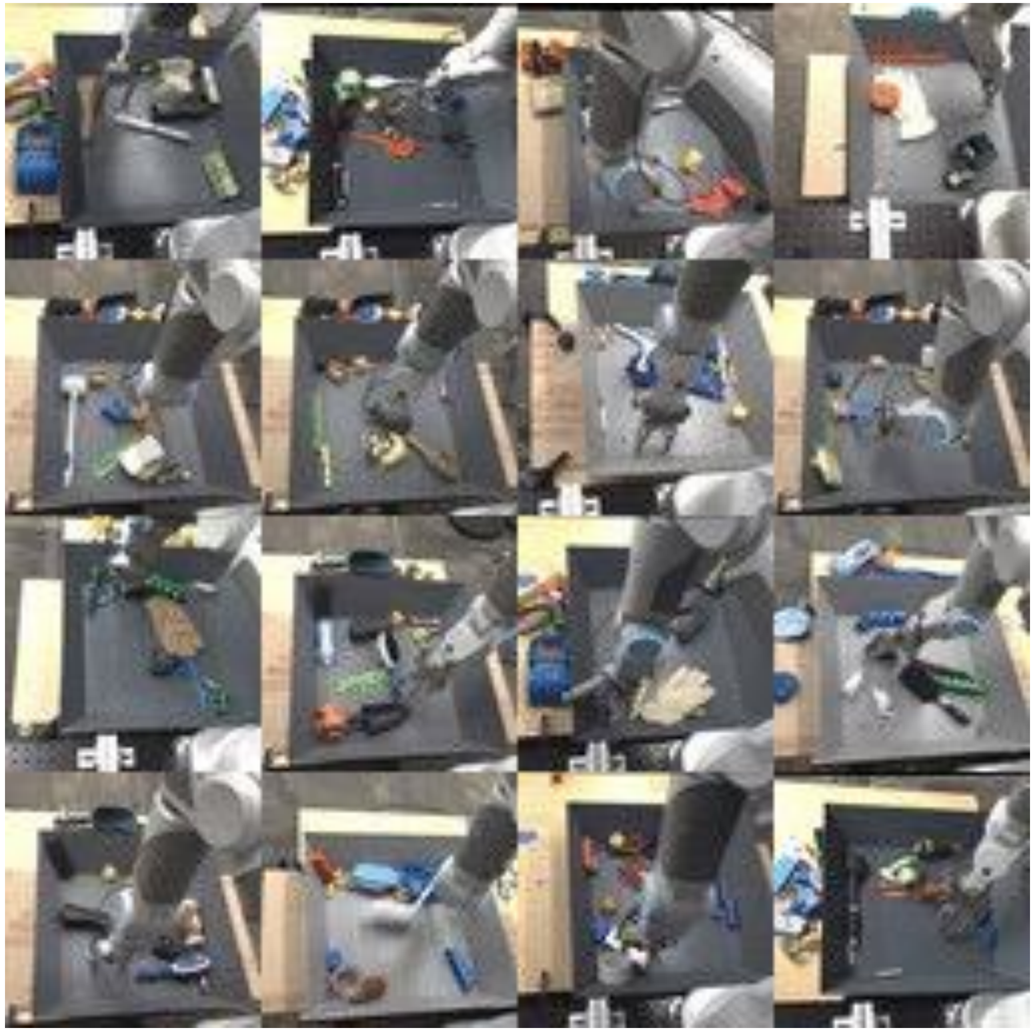
VPN Samples for Robotic Pushing

Video Pixel Net (VPN)



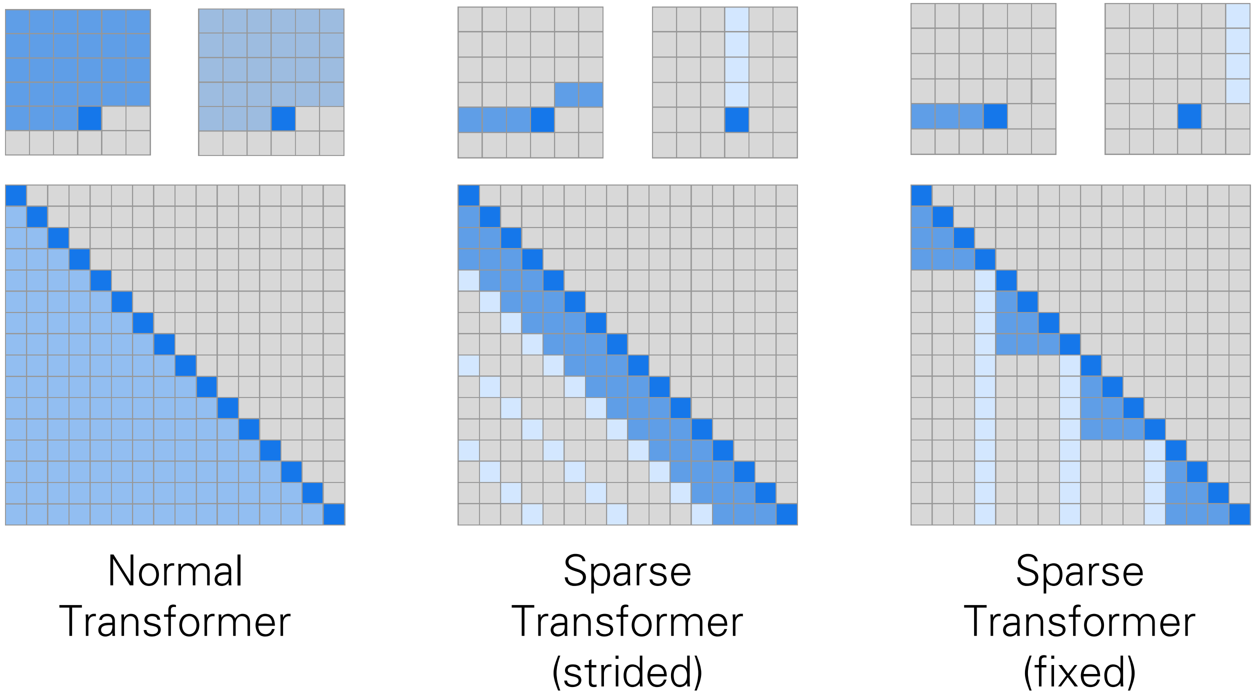
PixelCNN Decoders

Resolution Preserving CNN Encoders



VPN Samples for Robotic Pushing

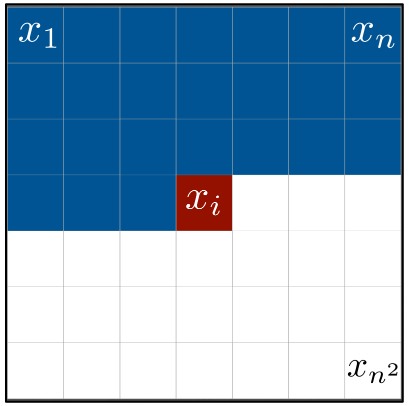
Sparse Transformers



- Strided attention is roughly equivalent to each position attending to its row and its column
- Fixed attention attends to a fixed column and the elements after the latest column element (especially used for text).

[Child, Gray, Radford, Sutskever, 2019]

Autoregressive Models



- Explicitly model conditional probabilities:

$$p_{\text{model}}(\boldsymbol{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i \mid x_1, \ldots, x_{i-1})$$

Each conditional can be a complicated neural net

Advantages:

- $p_{\text{model}}(x)$ is tractable (easy to train and sampl

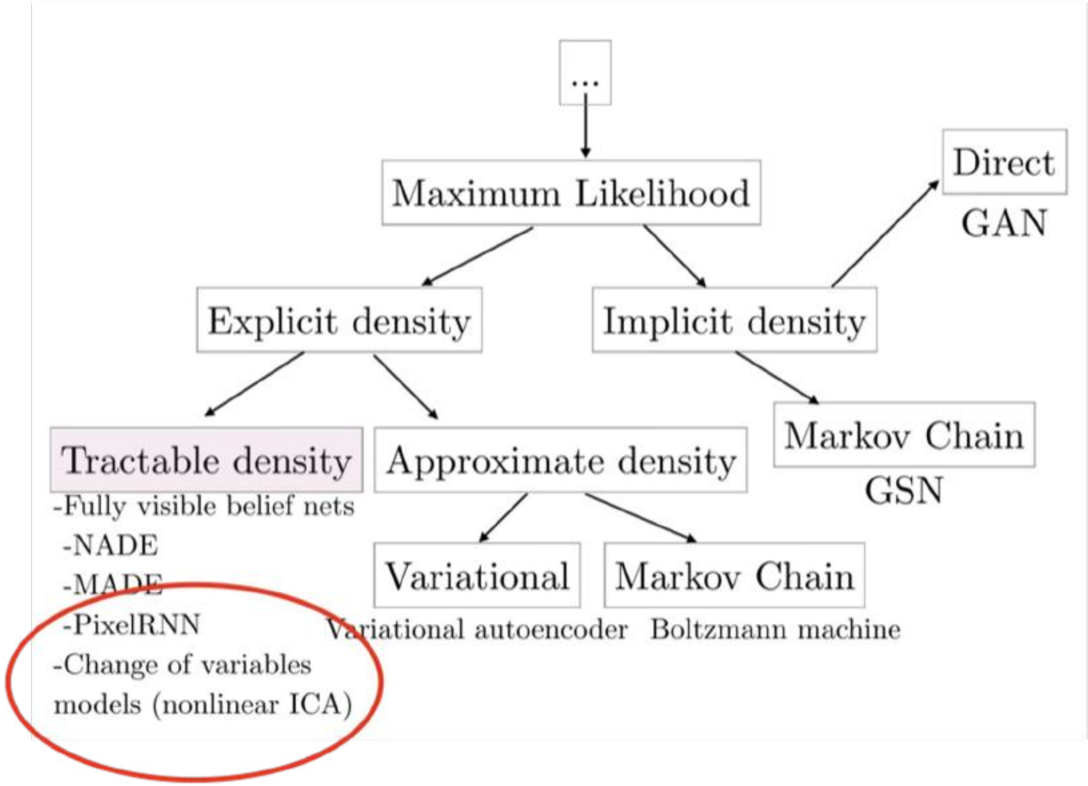
Disadvantages:

- Generation can be too costly
- Generation can not be controlled by a latent code

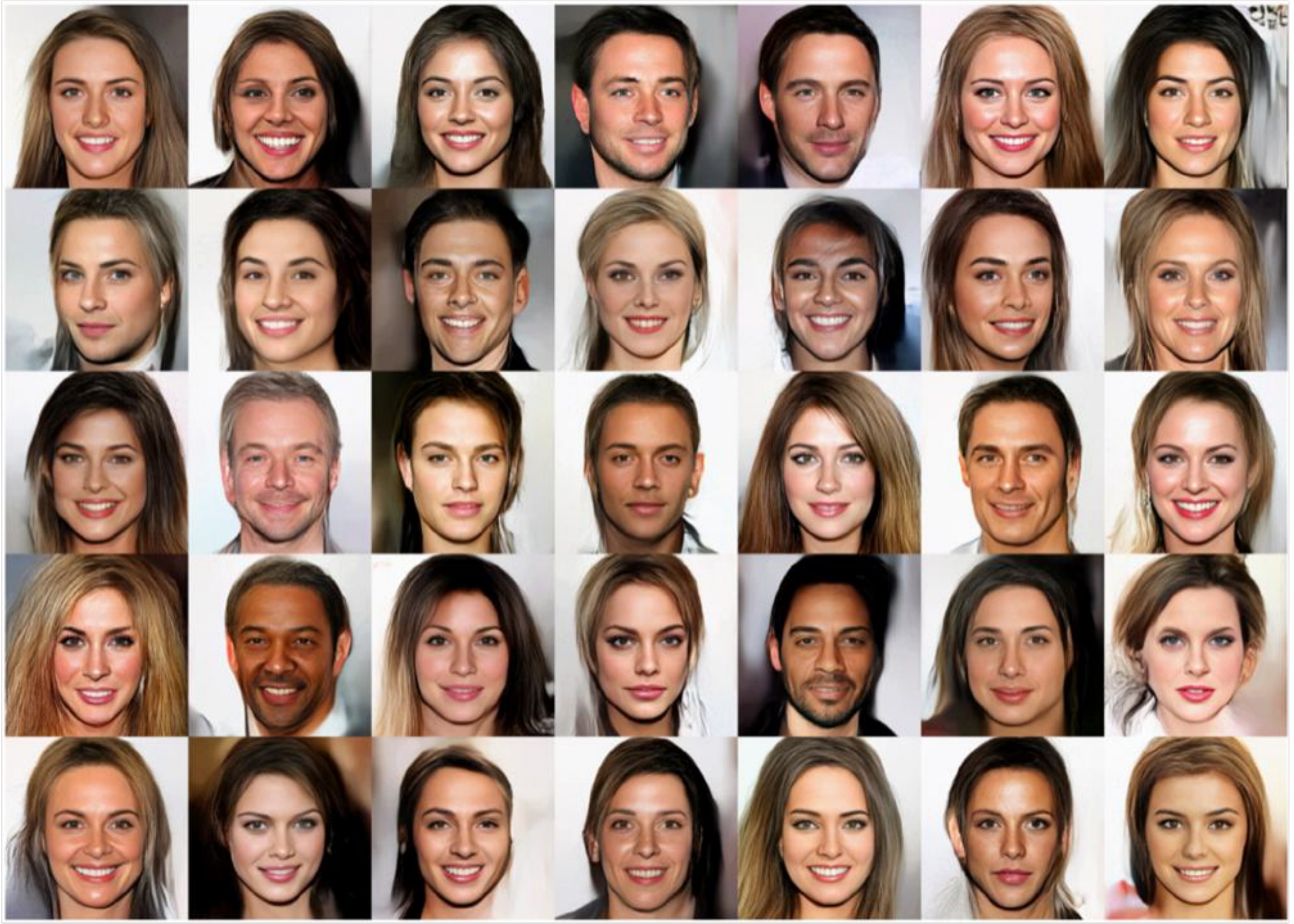
PixelCNN elephants
(van den Ord et al. 2016)

Flow-Based Models

Invertible Neural Networks

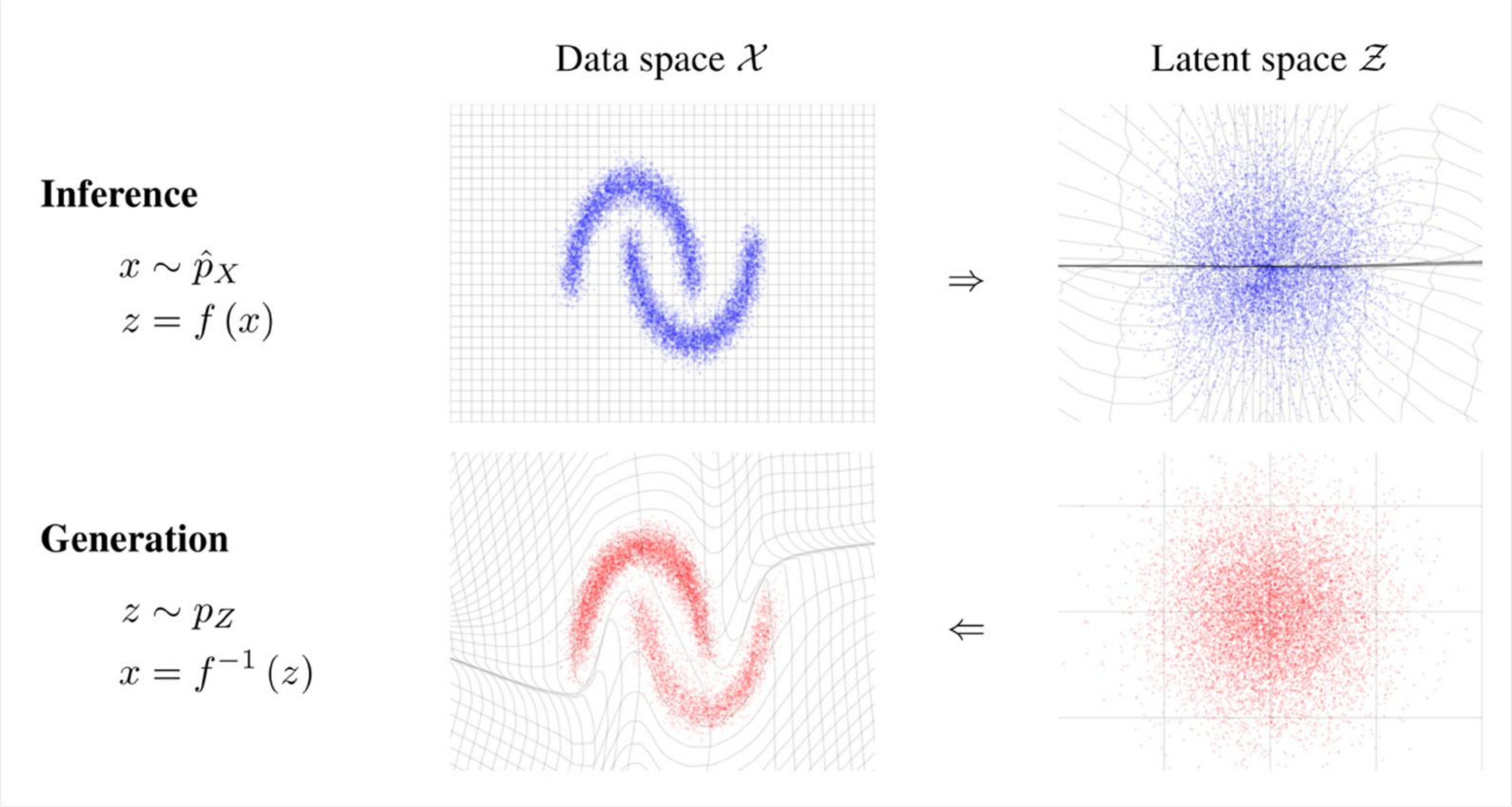


Goodfellow et al. 2016



Kingma and Dhariwal 2018

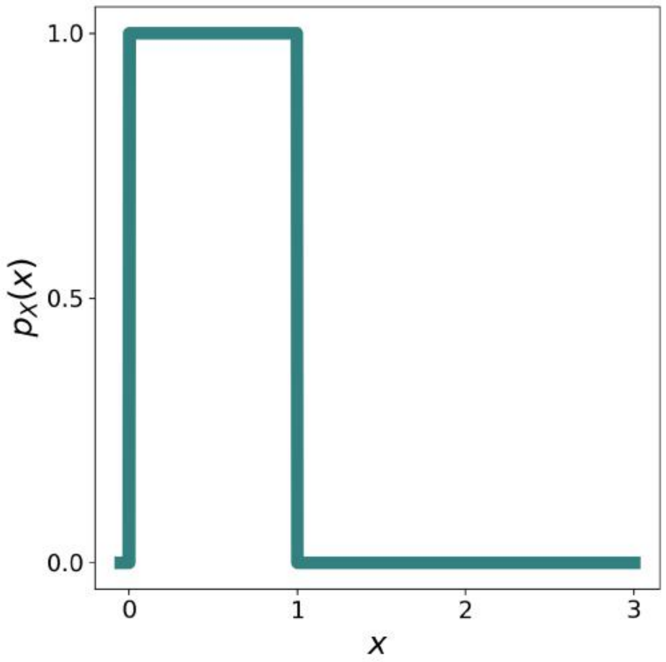
Normalizing Flows: Translating Probability Distributions

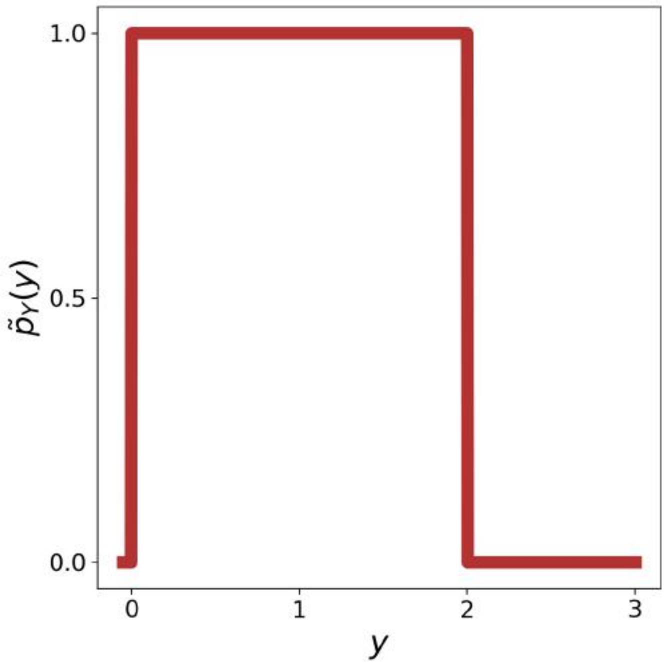


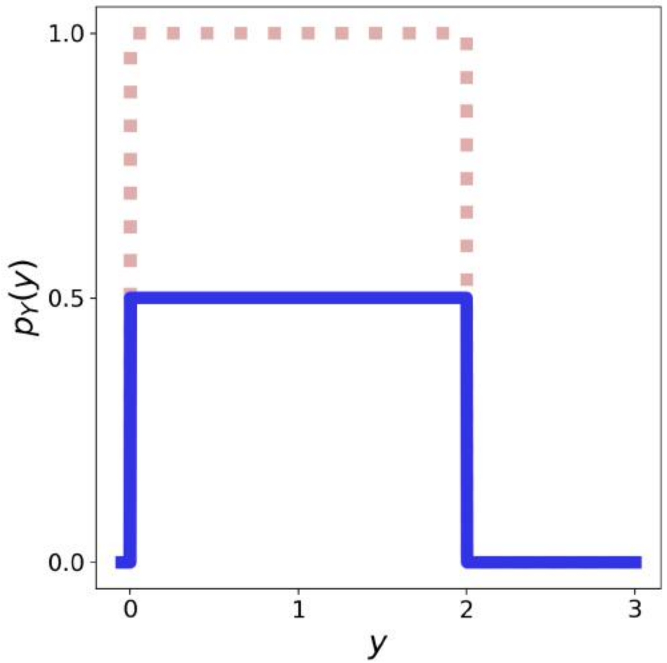
Change of Variable Density Needs to Be Normalized

$$X \sim p_X \qquad Y := 2X$$

$$p_X(x) = \begin{cases} 1 & \text{for } 0 \le x \le 1 \\ 0 & \text{else} \end{cases} \qquad \tilde{p}_Y(y) = p_X(y/2) \qquad p_Y(y) = p_X(y/2)/2$$



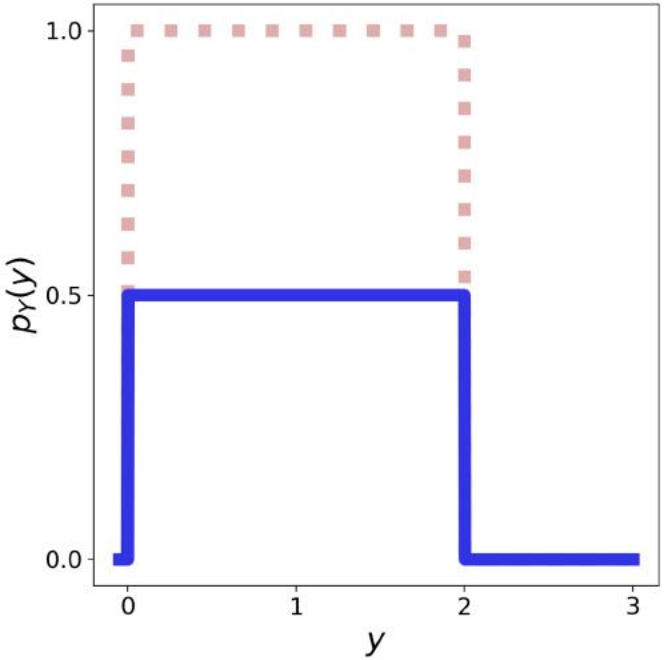




Change of Variable Density (m-Dimensional)

For a multivariable invertible mapping $f : \mathbb{R}^m \to \mathbb{R}^m$ $\quad X \sim p_X \quad Y := f(X)$

$$p_Y(y) = p_X(f^{-1}(y)) \left| \det \frac{\partial f^{-1}(y)}{\partial y} \right|$$



$Y := 2X$

$p_Y(y) = p_X(y/2)/2$

Local change of volume

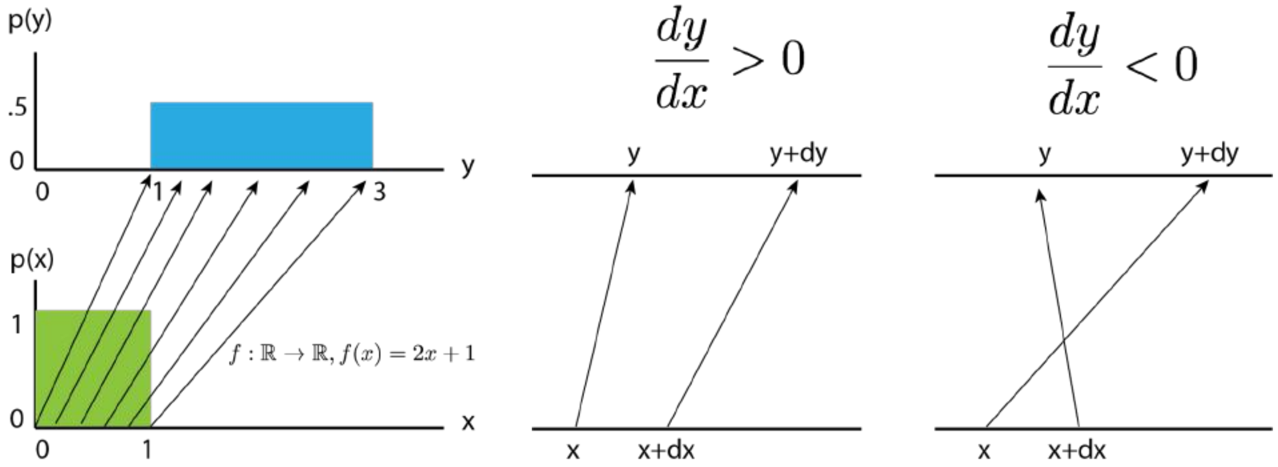
mass = density * volume

Change of Variable Density (m-Dimensional)

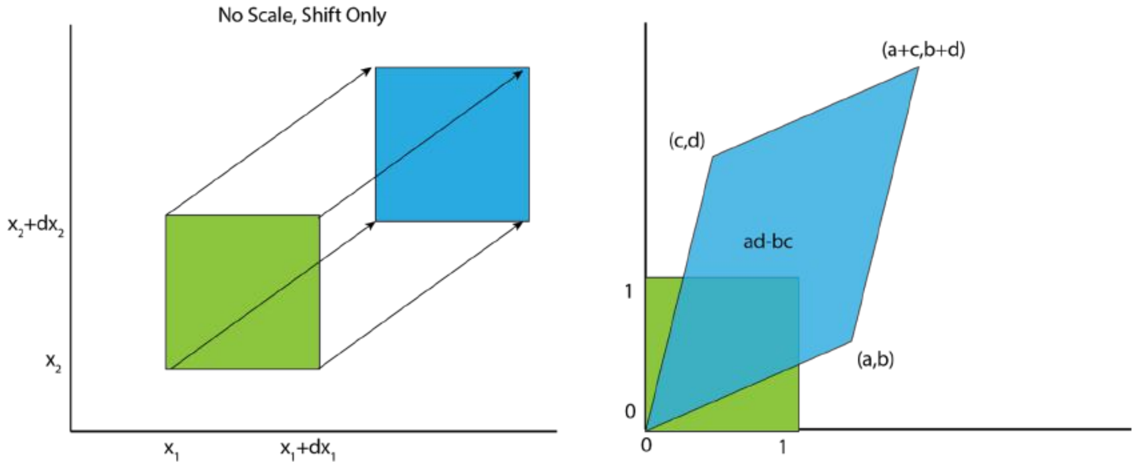
For a multivariable invertible mapping $f : \mathbb{R}^m \to \mathbb{R}^m \quad X \sim p_X \quad Y := f(X)$

$$p_Y(y) = p_X(f^{-1}(y)) \left| \det \frac{\partial f^{-1}(y)}{\partial y} \right|$$

1-D



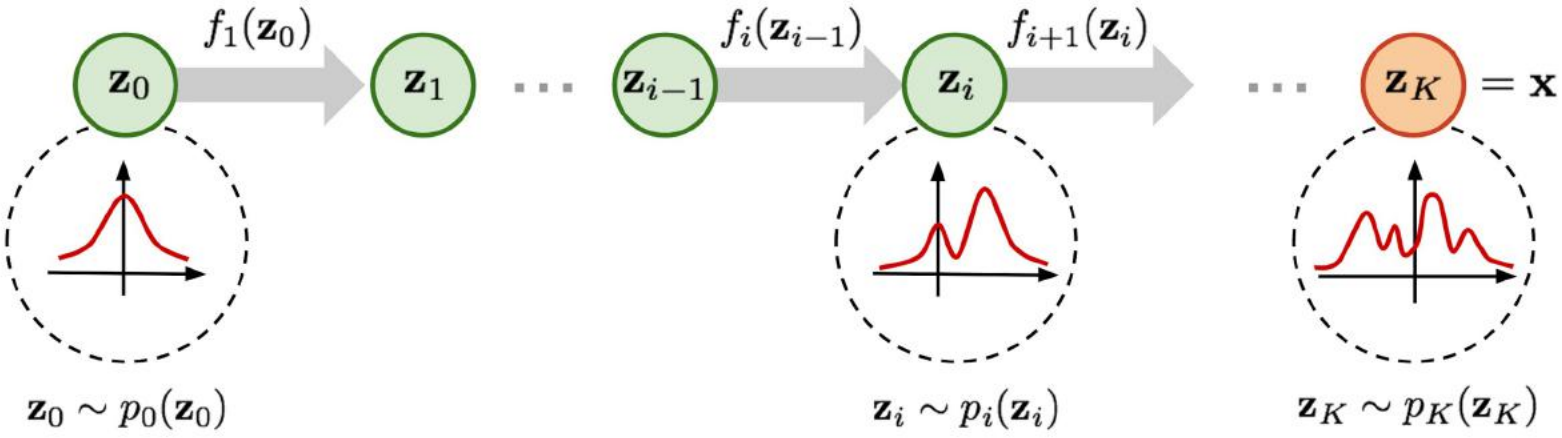
2-D



Figures from blog post: Normalizing Flows Tutorial, Part 1: Distributions and Determinants by Eric Jang

Chaining Invertible Mappings (Composition)

$$f = f_S \circ \cdots \circ f_2 \circ f_1 \qquad f(x) = f_S(\cdots f_2(f_1(x)))$$



$$\frac{\partial f(x)}{\partial x} = \frac{f_S(x_{S-1})}{\partial x_{S-1}} \cdots \frac{f_2(x_1)}{\partial x_1} \frac{f_1(x_0)}{\partial x_0} \qquad x_s = f_s(x_{s-1}) \quad x_0 = x$$

Chain rule

$$\det\left(\frac{\partial f(x)}{\partial x}\right) = \det\left(\frac{f_S(x_{S-1})}{\partial x_{S-1}}\right) \cdots \det\left(\frac{f_2(x_1)}{\partial x_1}\right) \det\left(\frac{f_1(x_0)}{\partial x_0}\right)$$

Determinant of matrix product

Figure from blog post: Flow-based Deep Generative Models by Lilian Weng, 2018

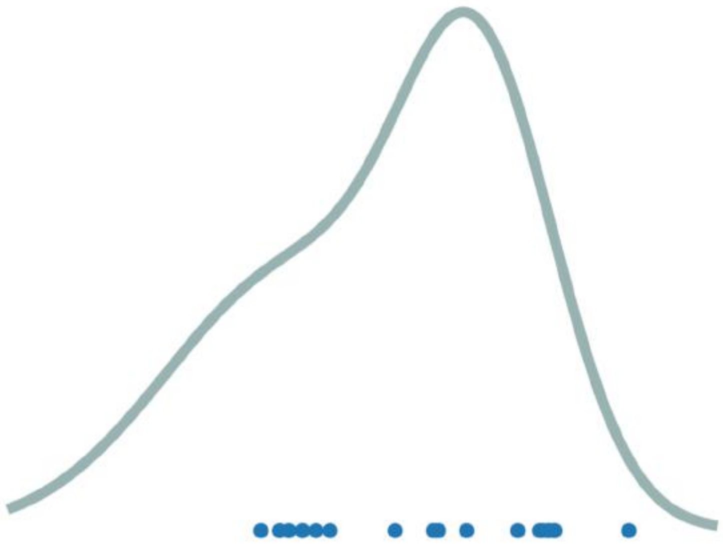
Training with Maximum Likelihood Principle

$$Z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad X = g(Z)$$

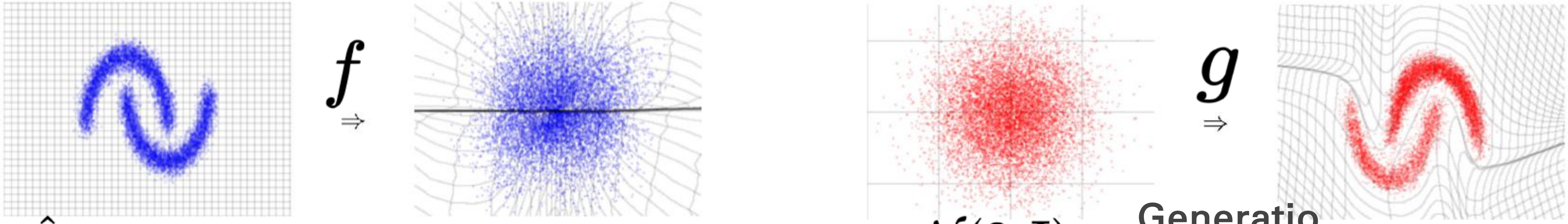
$$g = f^{-1} \text{ bijective}$$

$$\mathbb{E}_x\left[\log p(x)\right] = \mathbb{E}_x\left[\log \mathcal{N}(f(x); 0, I)\left|\det \frac{\partial f(x)}{\partial x}\right|\right]$$

Regularizes the entropy



Higher likelihood



$x \sim \hat{p}_X$ Inference $z \sim p_Z = \mathcal{N}(\mathbf{0}, \mathbf{I})$ Generation

Figures from Density Estimation Using Real NVP by Dinh et al., 2017

Pathways to Designing a Normalizing Flow

1. Require an invertible architecture.
   - Coupling layers, autoregressive, etc.

2. Require efficient computation of a change of variables equation.

$$\log p(x) = \log p(f(x)) + \log \left| \det \frac{df(x)}{dx} \right|$$

Model distribution   Base distribution

$\mathcal{O}(m^3)$

(or a continuous version) $\log p(x(t_N)) = \log p(x(t_0)) + \int_{t_0}^{t_N} \operatorname{tr}\left(\frac{\partial f(x(t), t)}{\partial x(t)}\right) dt$

Architectural Taxonomy

Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

1. Block coupling

NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow

2. Autoregressive

IAF/MAF/NAF
SOS polynomial
UMNN

Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

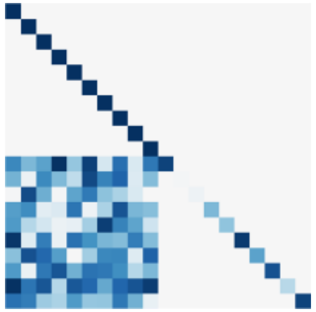
3. Det identity

Planar/Sylvester flows
Radial flow

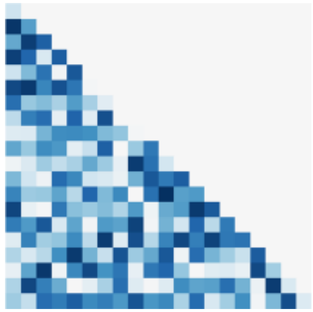
4. Stochastic estimation

Residual Flow
FFJORD

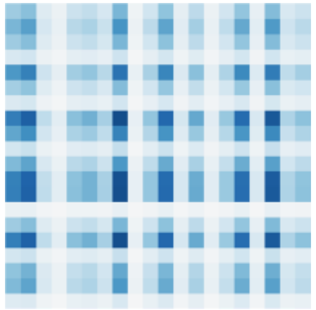
Jacobian



(Lower triangular + structured)



(Lower triangular)



(Low rank)

(Arbitrary)

Figures from Ricky Chen

Architectural Taxonomy

Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

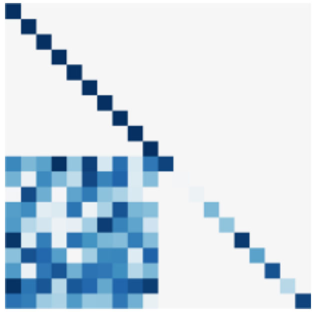
Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

1. Block coupling

NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow

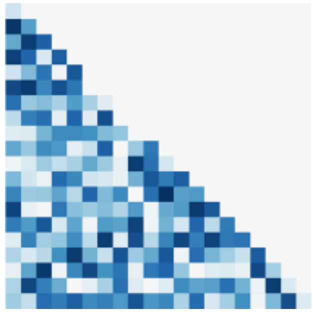
Jacobian



(Lower triangular + structured)

2. Autoregressive

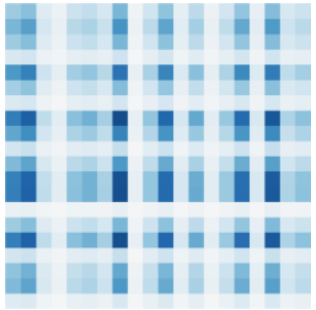
IAF/MAF/NAF
SOS polynomial
UMNN



(Lower triangular)

3. Det identity

Planar/Sylvester flows
Radial flow



(Low rank)

4. Stochastic estimation

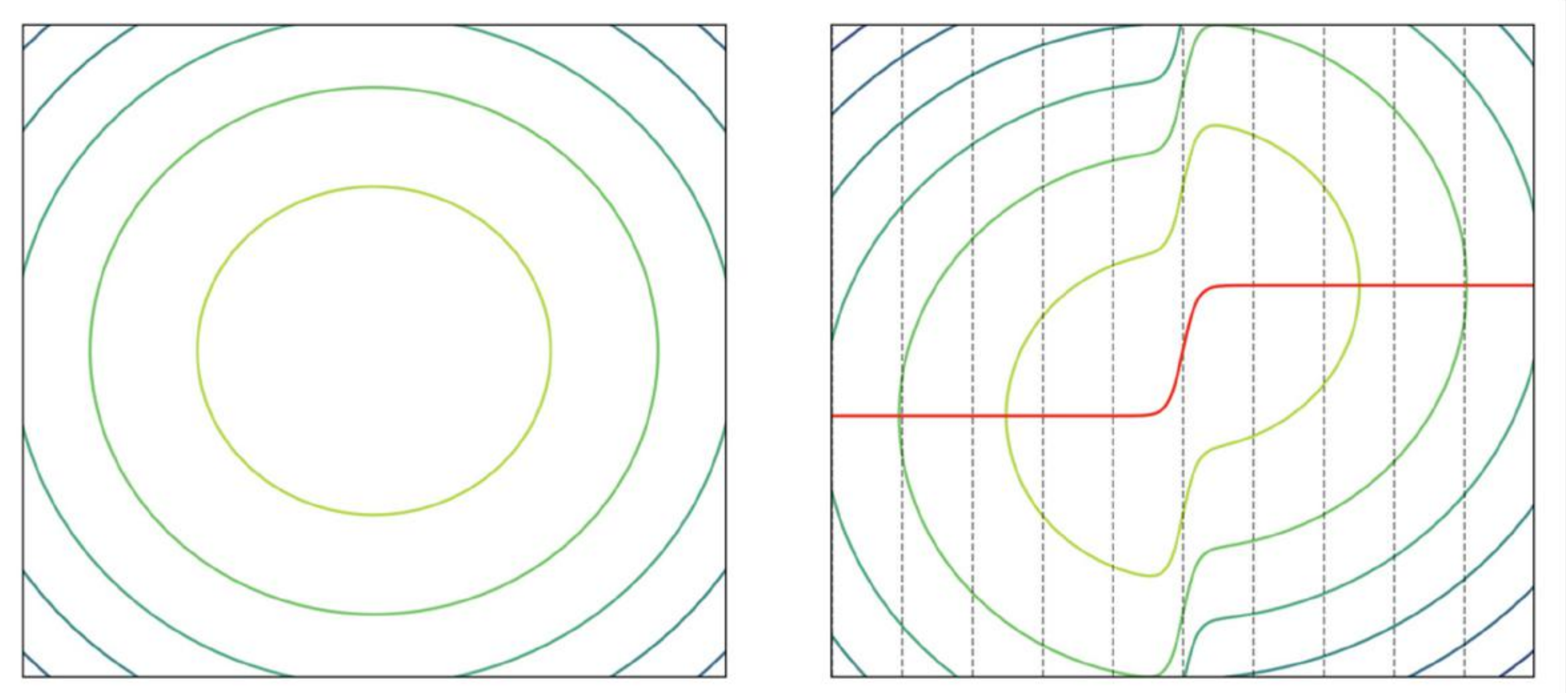
Residual Flow
FFJORD

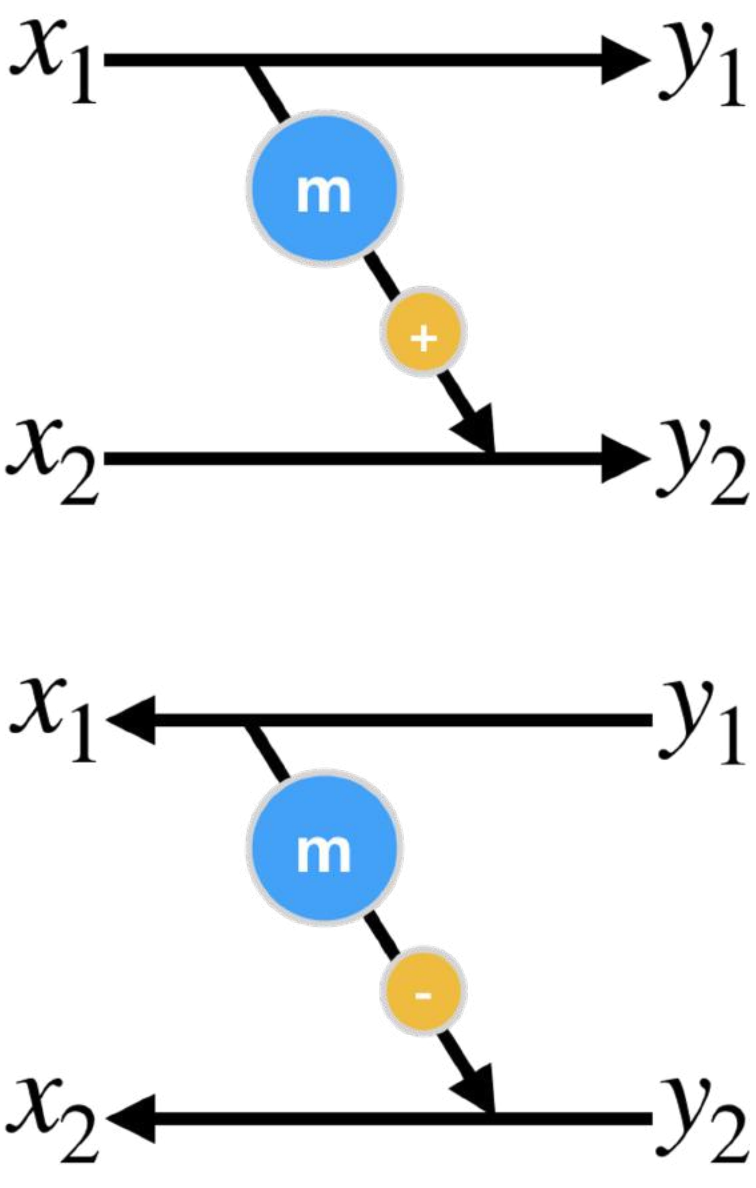
(Arbitrary)

Figures from Ricky Chen

Coupling Law - NICE

- General form $f(\boldsymbol{x})_1 = \boldsymbol{x}_1, \quad f(\boldsymbol{x})_2 = \boldsymbol{x}_2 + \mathcal{F}(\boldsymbol{x}_1)$
- Invertibility no constraint
- Jacobian determinant =1 (volume preserving)



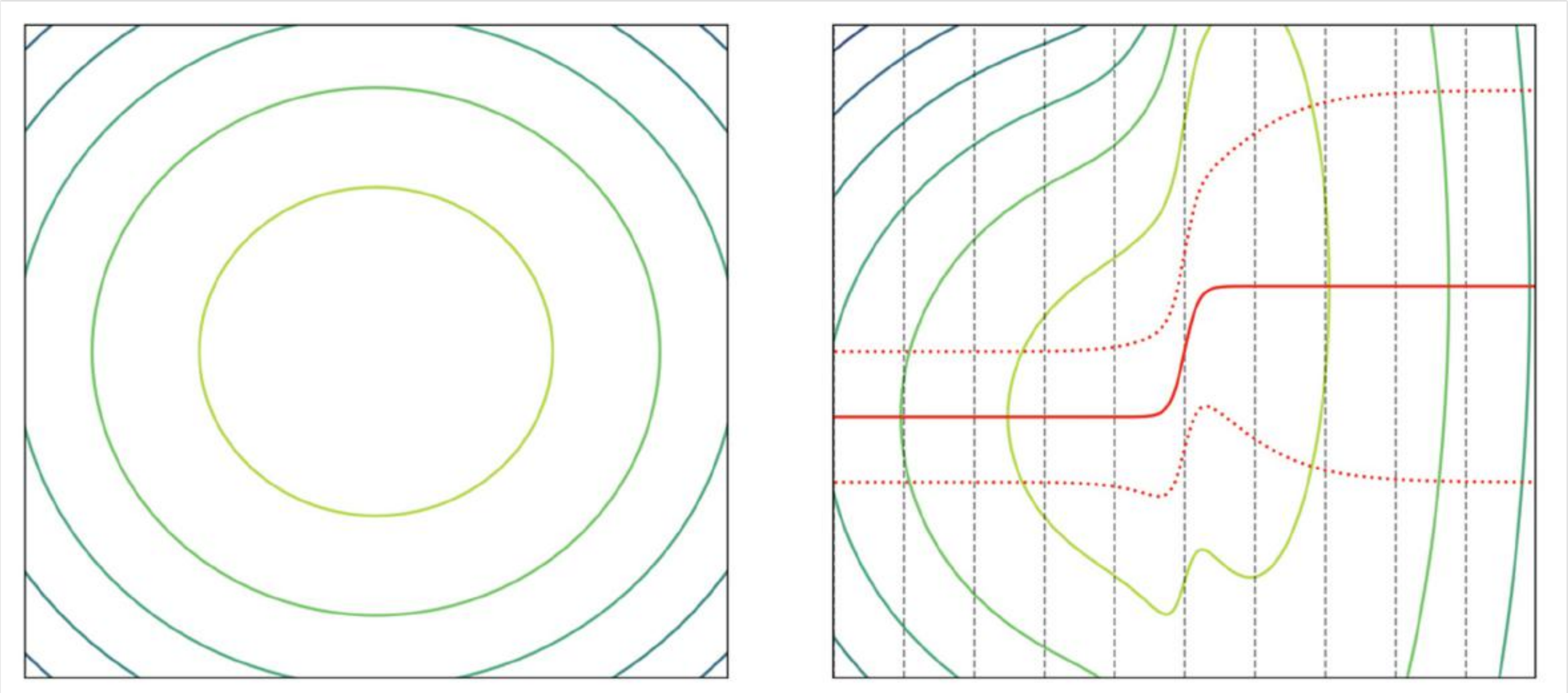


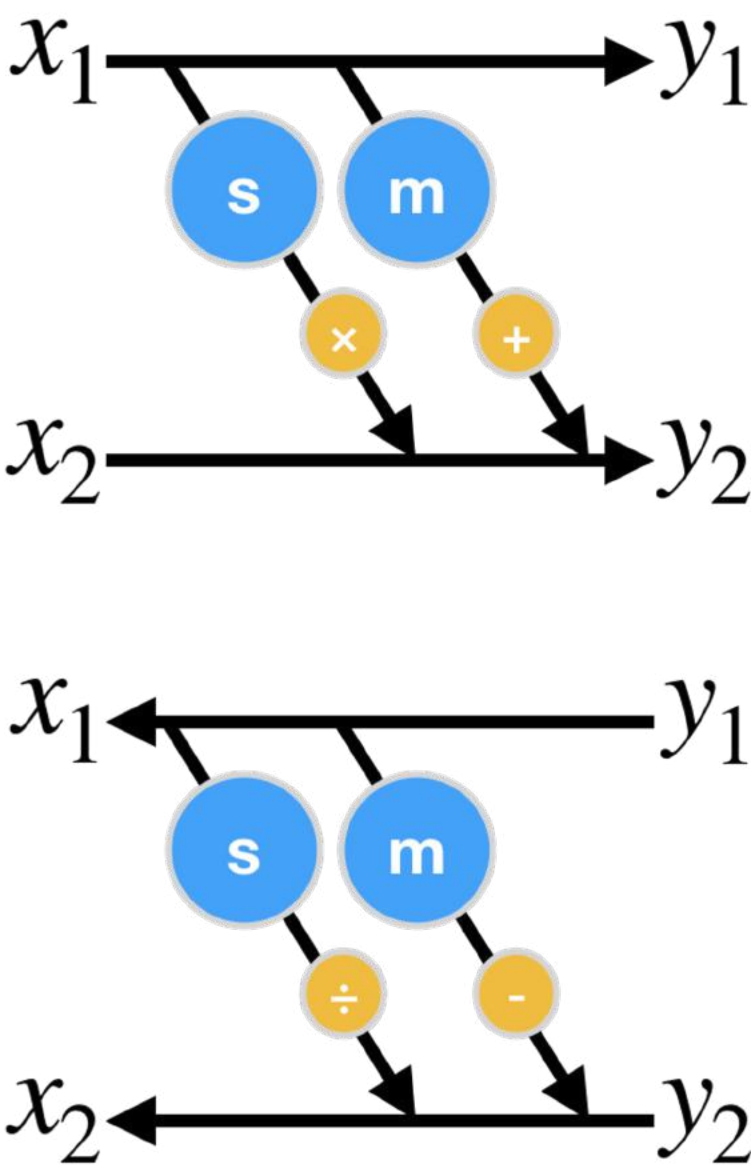
Dinh et al. 2014

Coupling Law - RealNVP

Real-valued
Non-Volume
Preserving

- General form
$$f(\boldsymbol{x})_1 = \boldsymbol{x}_1,$$
$$f(\boldsymbol{x})_2 = s(\boldsymbol{x}_1) \odot \boldsymbol{x}_2 + m(\boldsymbol{x}_1)$$
- Invertibility — s>0 (or simply non-zero)
- Jacobian determinant — product of s





Dinh et al. 2016

Real NVP via Masked Convolution

Partitioning can be implemented using a binary mask b, and using the functional form for y

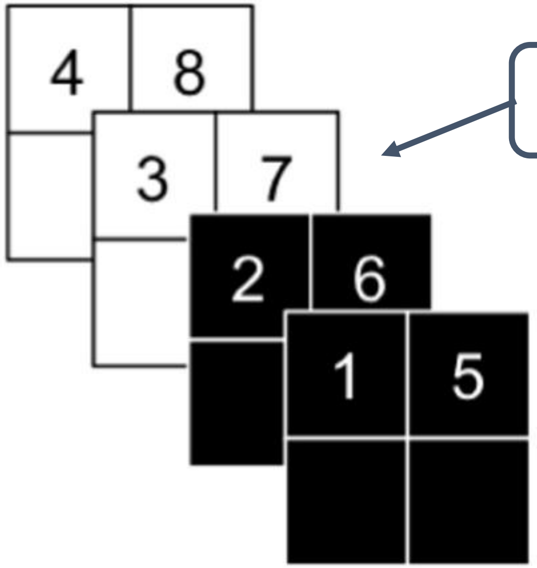
$$f(x) = b \odot x + (1 - b) \odot (x \odot \exp(s_{-}(b \odot x)) + m(b \odot x))$$

Real NVP via Masked Convolution

Partitioning can be implemented using a binary mask b, and using the functional form for y

$$f(x) = b \odot x + (1 - b) \odot (x \odot \exp(s\_(b \odot x)) + m(b \odot x))$$

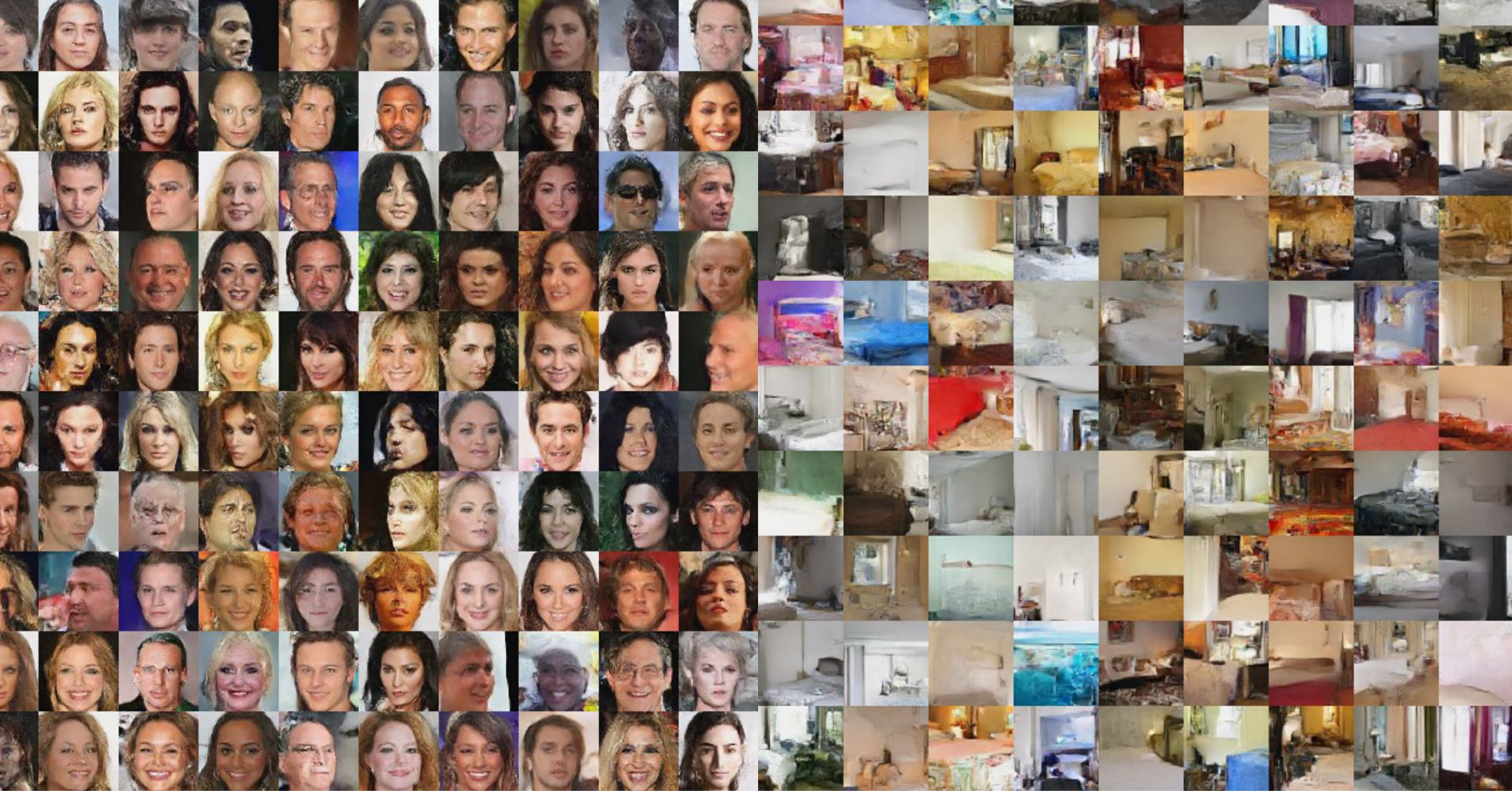
The **spatial checkerboard pattern mask** has value 1 where the sum of spatial coordinates is odd, and 0 otherwise.



After a "**squeeze**" operation

The channel-wise mask b is 1 for the first half of the channel dimensions and 0 for the second half.

Figures from Density Estimation Using Real NVP by Dinh et al., 2017

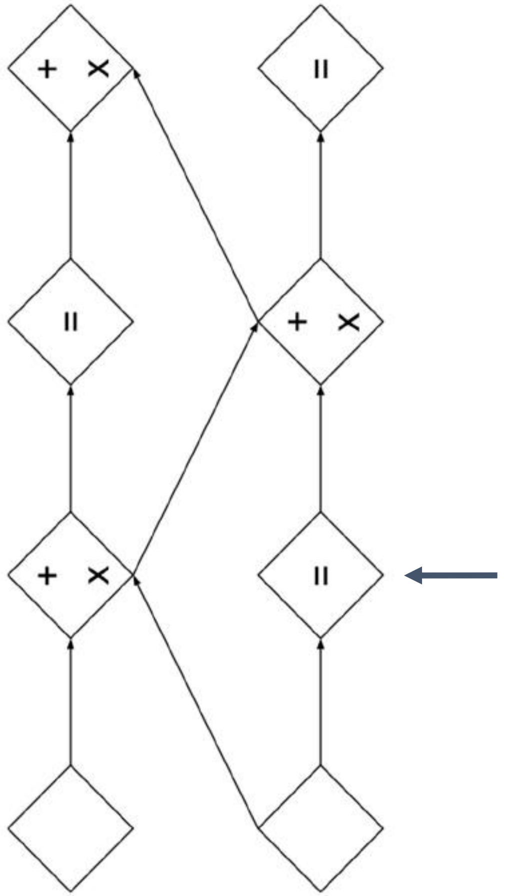


Celeba-64 (left) and LSUN bedroom (right)

Figures from Density Estimation Using Real NVP by Dinh et al., 2017

Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)

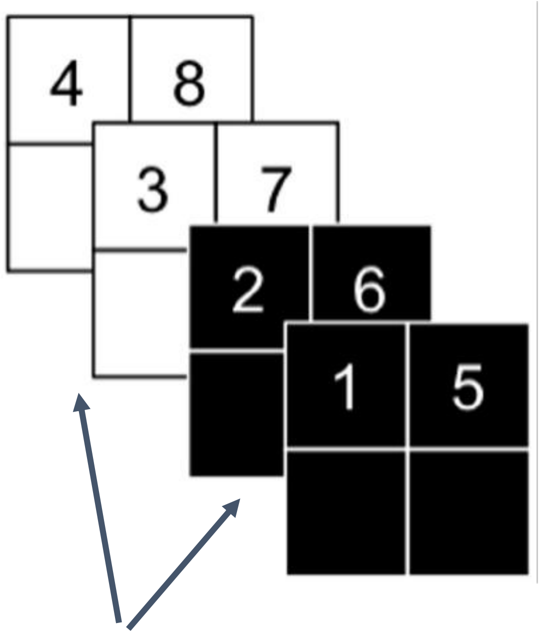


Unchanged in the first transform

Figure from Density Estimation Using Real NVP by Dinh et al., 2017

Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)

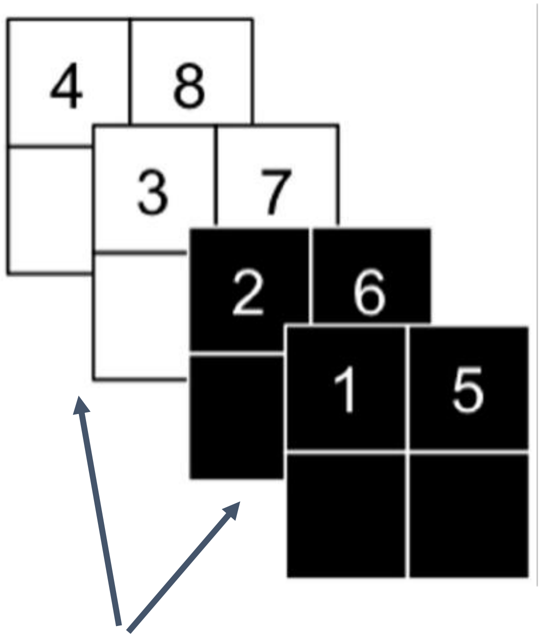


Alternating masks

Figure from Density Estimation Using Real NVP by Dinh et al., 2017

Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)



Alternating masks

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ x_1 \\ x_2 \end{bmatrix}$$

Replace with a general invertible matrix W

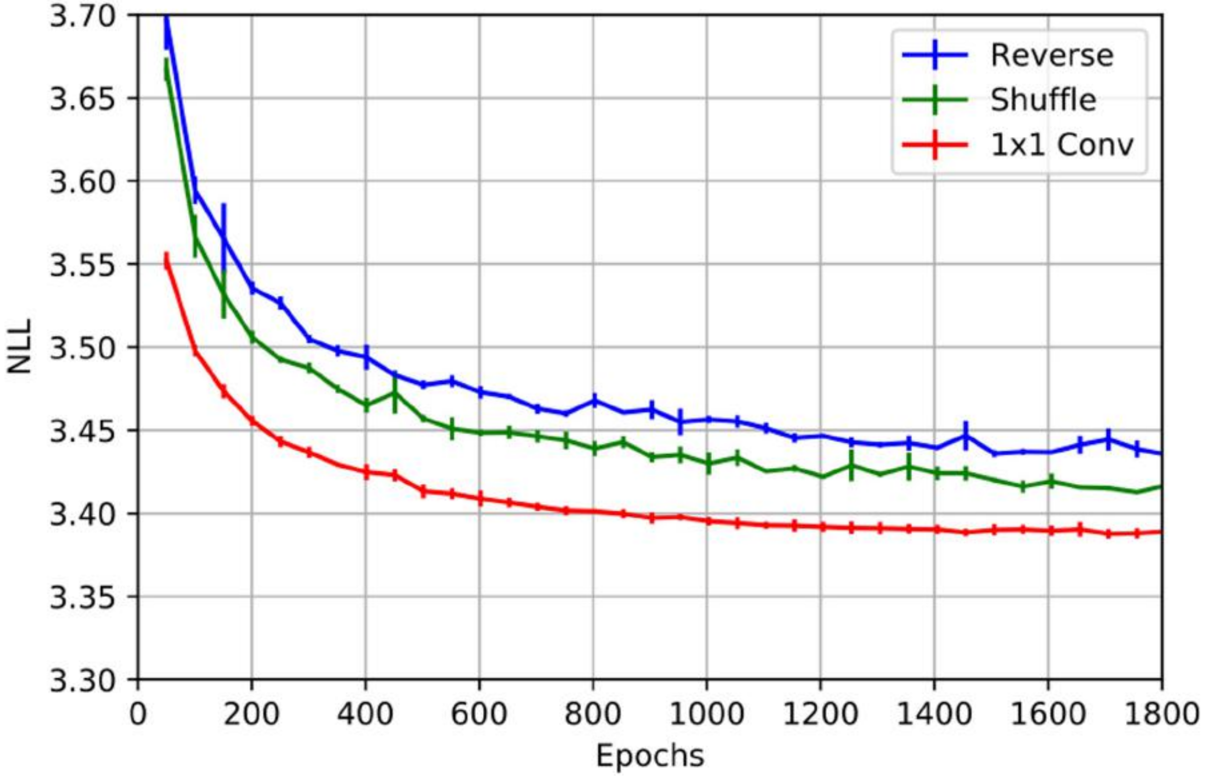
Represent W as a 1x1 convolutional kernel of shape [c, c, 1, 1]; c being # channels

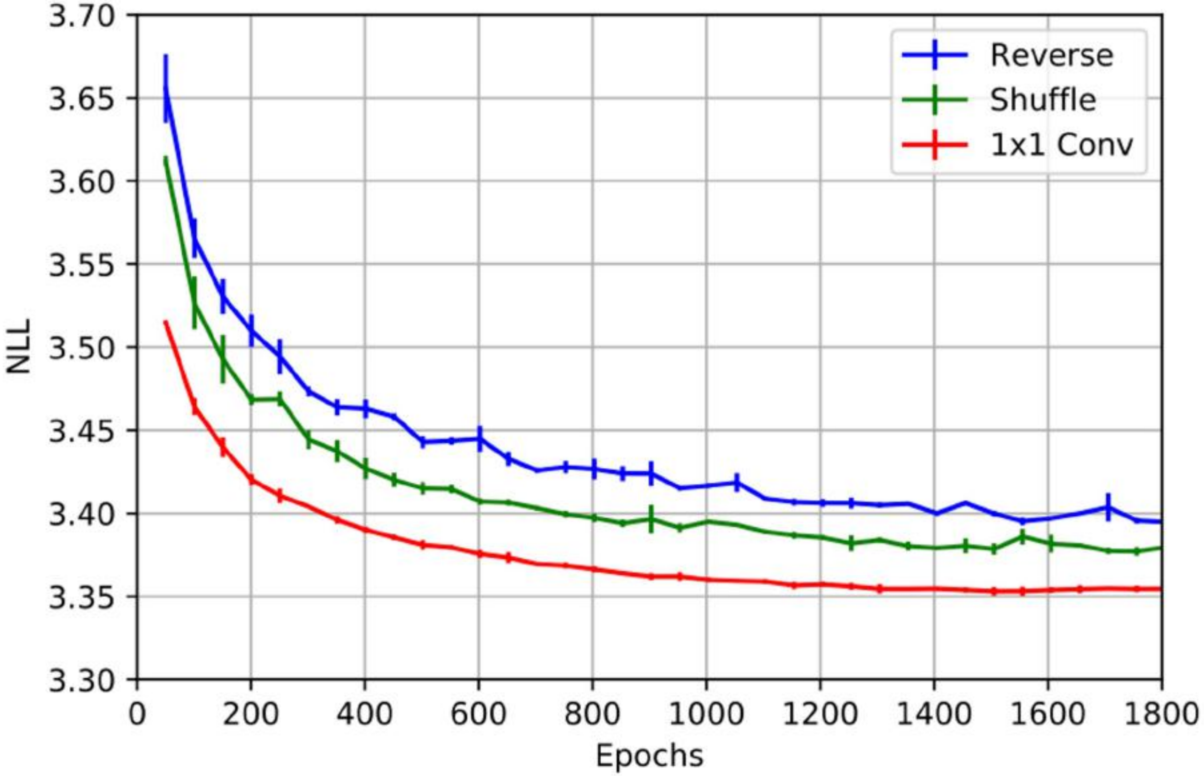
$$\log\left|\det\left(\frac{\partial \, \text{conv2D}(h; W)}{\partial h}\right)\right| = h \cdot w \cdot \log|\det(W)|$$

Figure from Density Estimation Using Real NVP by Dinh et al., 2017

Ablation: Permutation vs 1x1 Convolution

Model	CIFAR-10	ImageNet 32x32	ImageNet 64x64	LSUN (bedroom)	LSUN (tower)	LSUN (church outdoor)
RealNVP	3.49	4.28	3.98	2.72	2.81	3.08
Glow	**3.35**	**4.09**	**3.81**	**2.38**	**2.46**	**2.67**

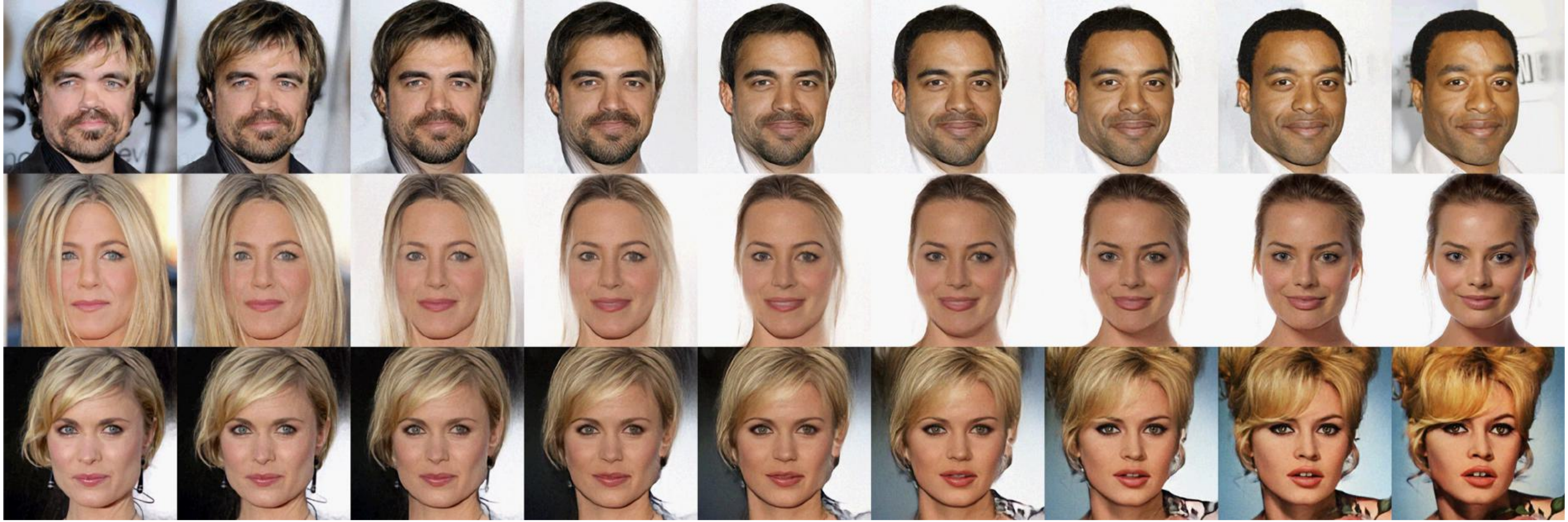




Bits-per-dim on CIFAR: left: additive, right: affine

Results from Glow: Generative Flow with Invertible 1×1 Convolutions by Kingma and Dhariwal, 2018

Figure from Glow: Generative Flow with Invertible 1×1 Convolutions by Kingma and Dhariwal, 2018



Interpolation with Generative Flows

Figure from *Glow: Generative Flow with Invertible 1×1 Convolutions* by Kingma and Dhariwal, 2018
Video from Durk Kingma's youtube channel

Architectural Taxonomy

Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

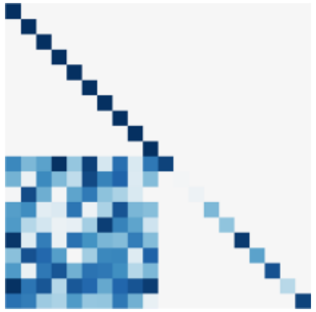
Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

1. Block coupling

NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow

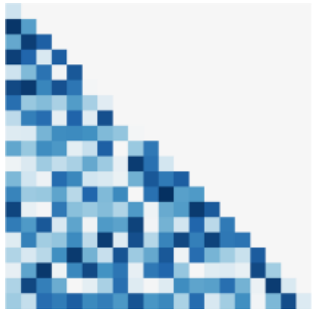
Jacobian



(Lower triangular + structured)

2. Autoregressive

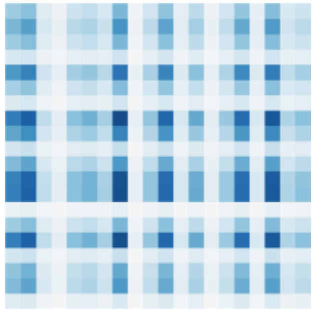
IAF/MAF/NAF
SOS polynomial
UMNN



(Lower triangular)

3. Det identity

Planar/Sylvester flows
Radial flow



(Low rank)

4. Stochastic estimation

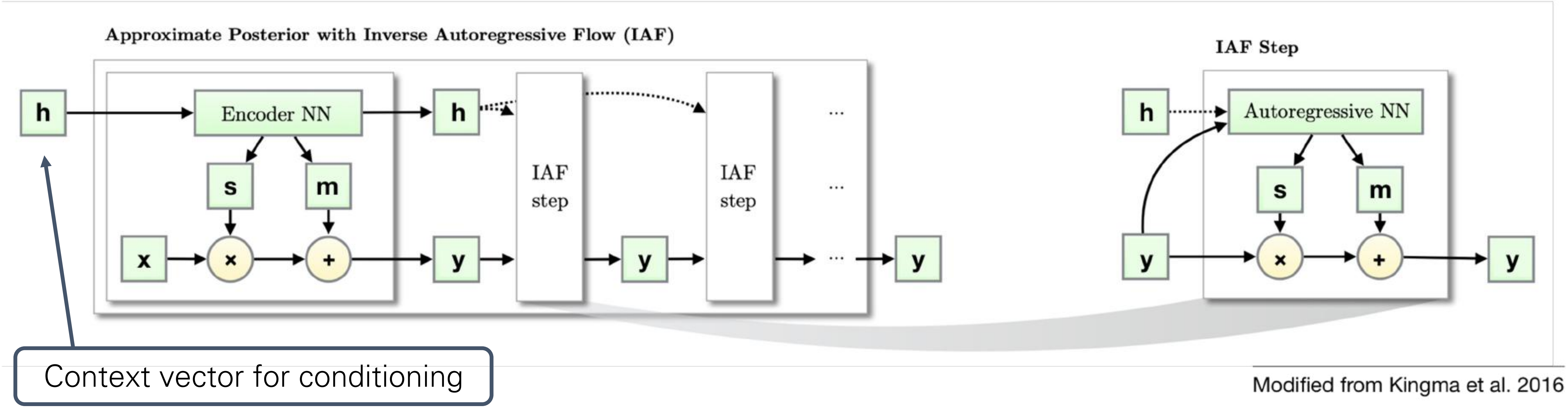
Residual Flow
FFJORD

(Arbitrary)

Figures from Ricky Chen

Inverse (Affine) Autoregressive Flows

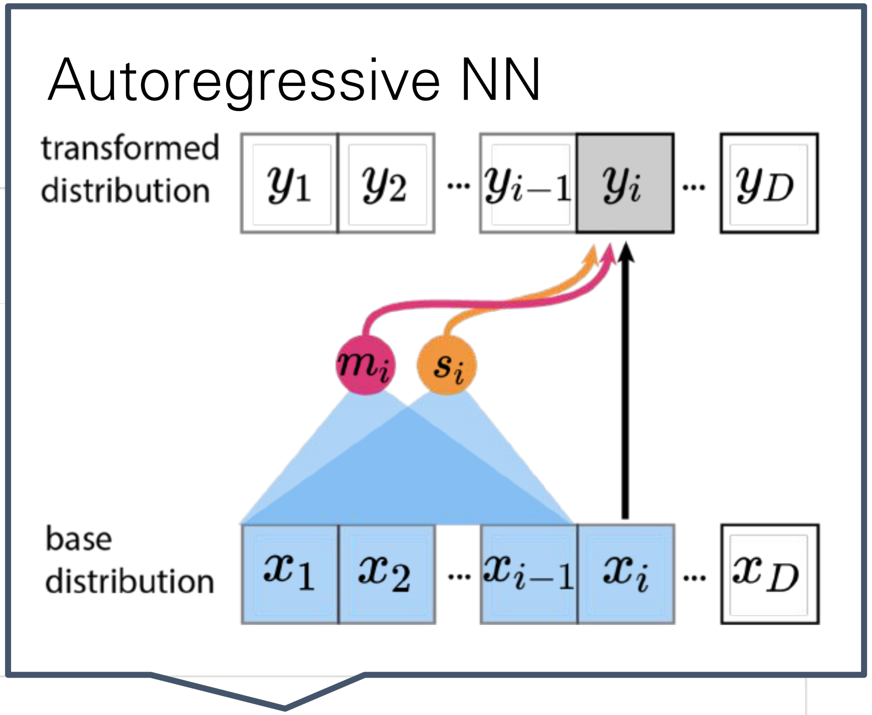
- General form $f(\boldsymbol{x})_t = s(\boldsymbol{x}_{<t}) \cdot \boldsymbol{x}_t + m(\boldsymbol{x}_{<t})$
- Invertibility s>0 (or simply non-zero)
- Jacobian determinant product of s

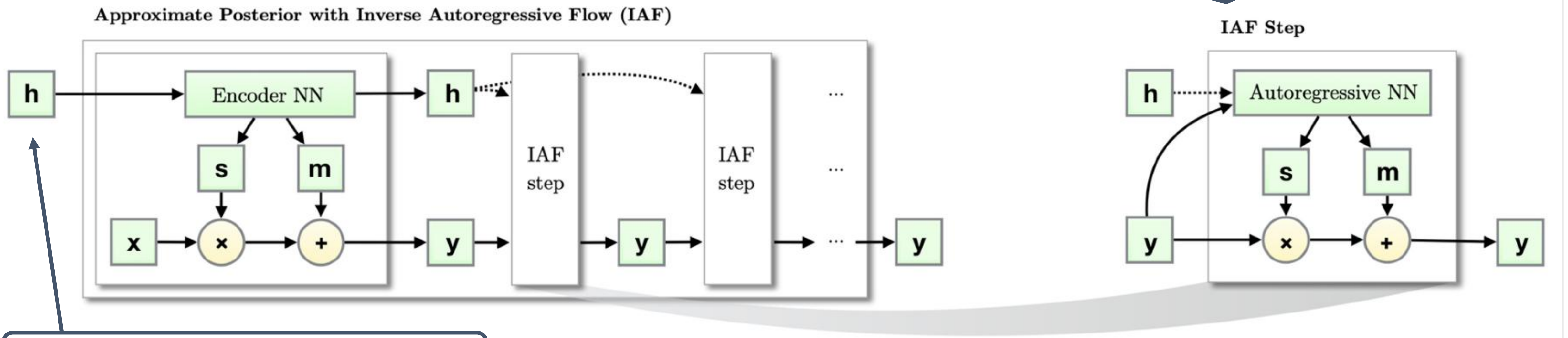


Modified from Kingma et al. 2016

Inverse Autoregressive Flows

- General form $$f(\boldsymbol{x})_t = s(\boldsymbol{x}_{<t}) \cdot \boldsymbol{x}_t + m(\boldsymbol{x}_{<t})$$
- Invertibility: s>0 (or simply non-zero)
- Jacobian determinant: product of s





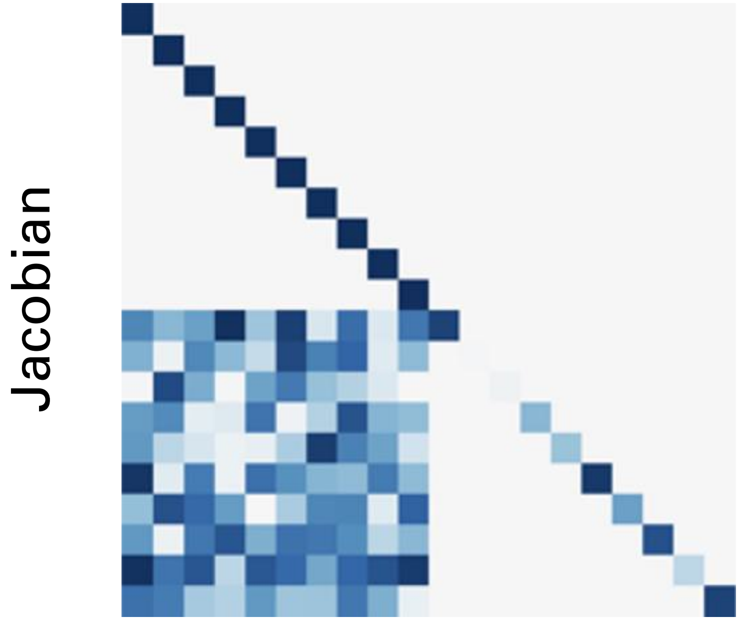
Context vector for conditioning

Modified from Kingma et al. 2016

Trade-off between Expressivity and Inversion Cost

Block autoregressive

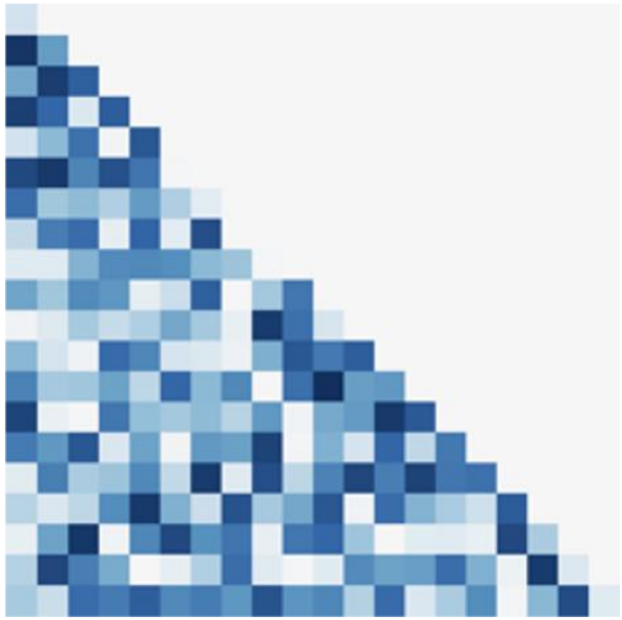
- Limited capacity
- Inverse takes constant time



(Block triangular)

Autoregressive

- Higher capacity
- Inverse takes linear time (dimensionality)

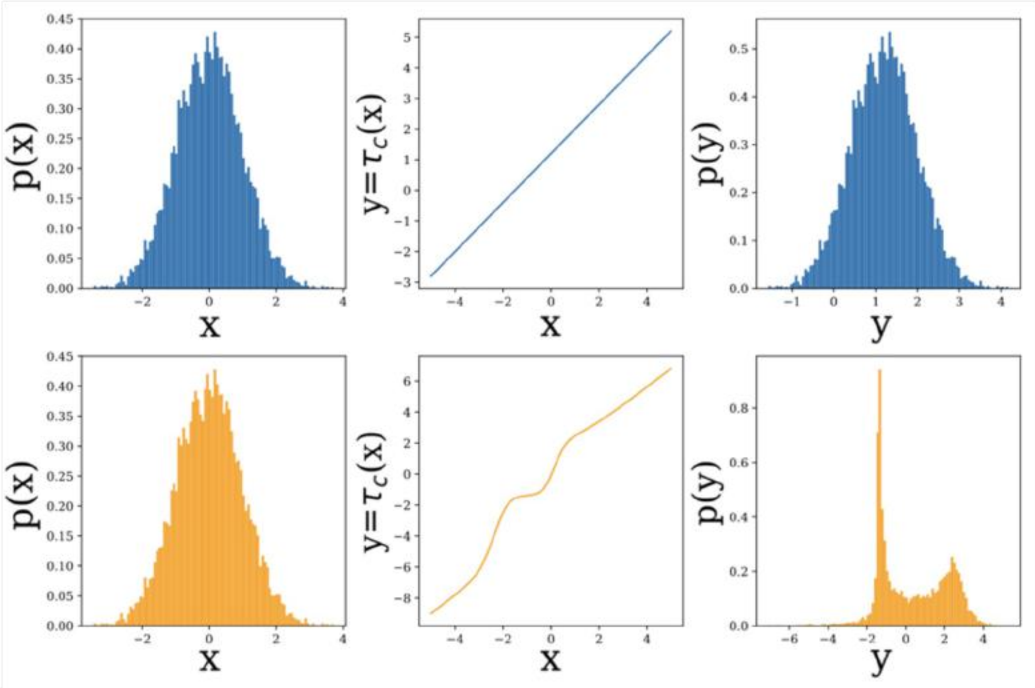


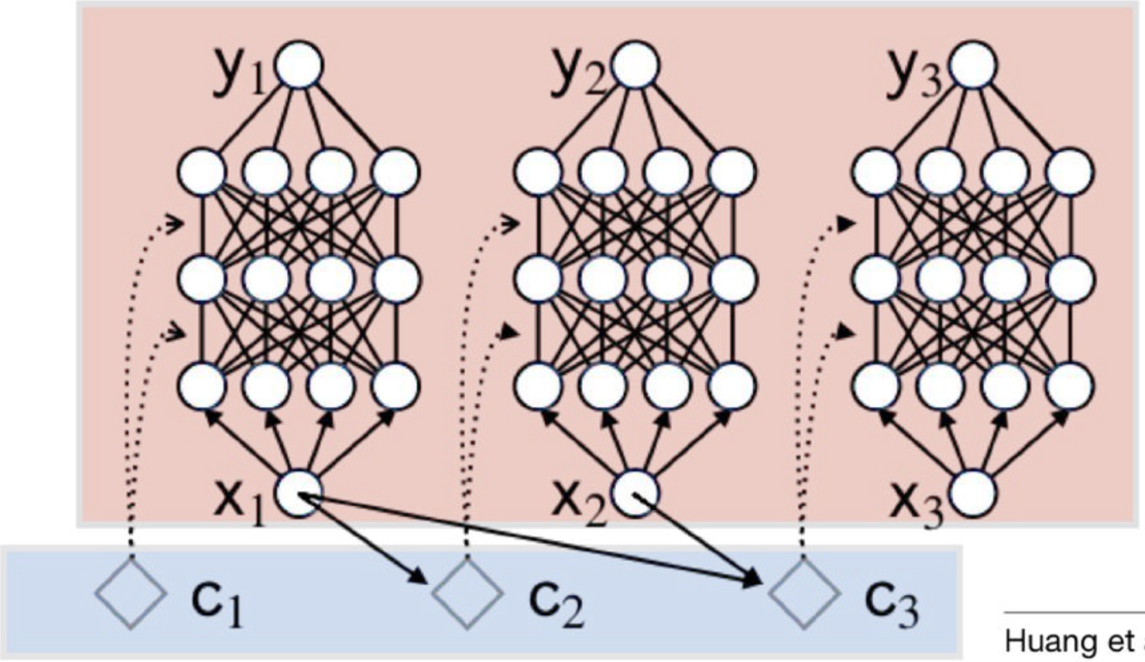
(Triangular)

Figures from Ricky Chen

Neural Autoregressive Flows

- General form $f(\boldsymbol{x})_t = \mathcal{P}(\boldsymbol{x}_t; \mathcal{H}(\boldsymbol{x}_{<t}))$
- Invertibility monotonic activation and positive weight in $\mathcal{P}$
- Jacobian determinant product of derivatives (elementwise)





Huang et al. 2018

Architectural Taxonomy

Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

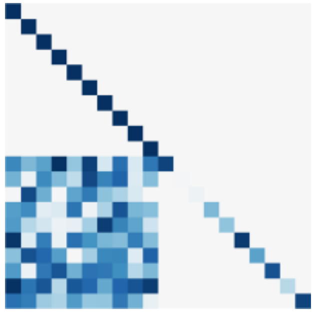
Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

1. Block coupling

NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow

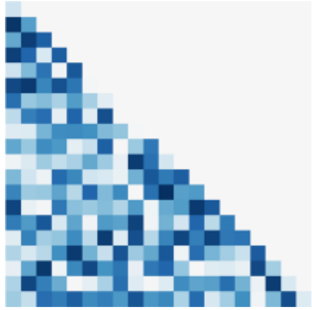
Jacobian



(Lower triangular + structured)

2. Autoregressive

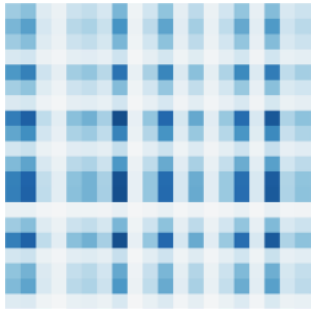
IAF/MAF/NAF
SOS polynomial
UMNN



(Lower triangular)

3. Det identity

Planar/Sylvester flows
Radial flow



(Low rank)

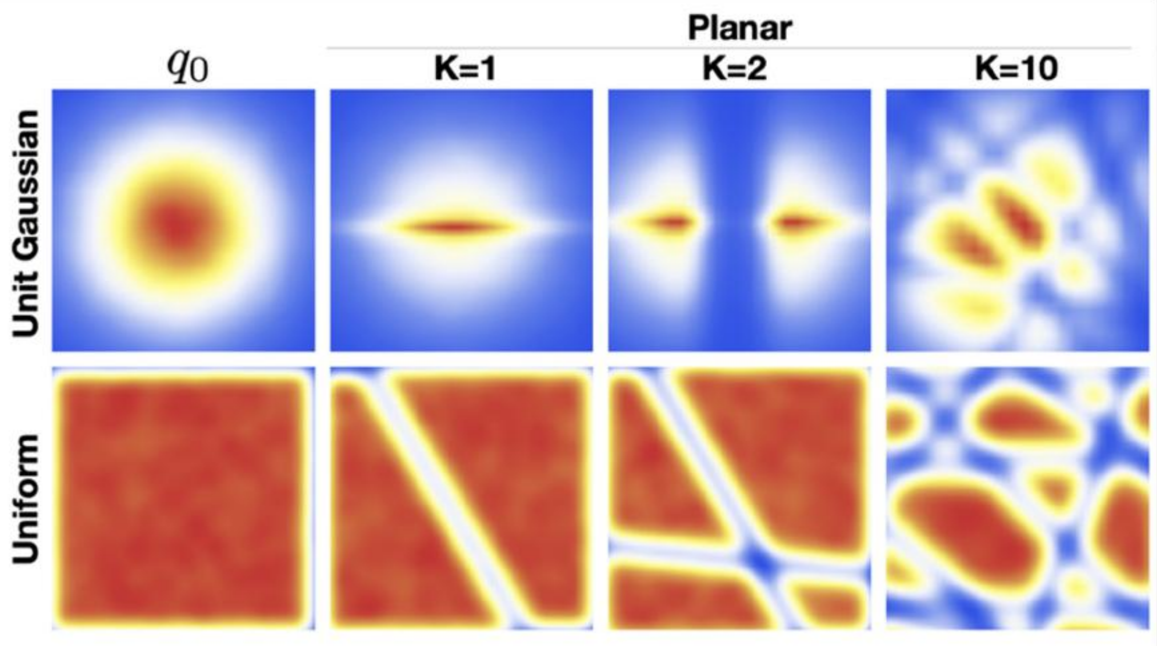
4. Stochastic estimation

Residual Flow
FFJORD

(Arbitrary)

Determinant Identity – Planar Flows

- General form $$f(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}h(\boldsymbol{w}^\top \boldsymbol{x} + b)$$
- Invertibility $$\boldsymbol{u}^\top \boldsymbol{w} > -1 \text{ if } h = \tanh$$
- Jacobian determinant $$\left|\det \frac{\partial f}{\partial \boldsymbol{x}}\right| = \left|\det\left(\boldsymbol{I} + h'(\boldsymbol{w}^\top \boldsymbol{x} + b)\boldsymbol{u}\boldsymbol{w}^\top\right)\right| = \left|1 + h'(\boldsymbol{w}^\top \boldsymbol{x} + b)\boldsymbol{u}^\top \boldsymbol{w}\right|$$



VAE on binary MNIST

Model	$-\ln p(\mathbf{x})$
DLGM diagonal covariance	≤ 89.9
DLGM+NF (k = 10)	≤ 87.5
DLGM+NF (k = 20)	≤ 86.5
DLGM+NF (k = 40)	≤ 85.7
DLGM+NF (k = 80)	≤ 85.1

Rezende et al. 2015

Determinant Identity – Sylvester Flows

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{A}h(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{b})$ $\boldsymbol{A} \in \mathbb{R}^{m \times d}$, $\boldsymbol{B} \in \mathbb{R}^{d \times m}$, $\boldsymbol{b} \in \mathbb{R}^{d}$, and $d \leq m$
- Invertibility Similar to planar flows
- Jacobian determinant Using Sylvester's Thm: $\det(\boldsymbol{I}_m + \boldsymbol{A}\boldsymbol{B}) = \det(\boldsymbol{I}_d + \boldsymbol{B}\boldsymbol{A})$

Model	**Freyfaces**		**Omniglot**		**Caltech 101**	
	-ELBO	NLL	-ELBO	NLL	-ELBO	NLL
VAE	4.53 ± 0.02	4.40 ± 0.03	104.28 ± 0.39	97.25 ± 0.23	110.80 ± 0.46	99.62 ± 0.74
Planar	$\mathbf{4.40 \pm 0.06}$	$\mathbf{4.31 \pm 0.06}$	102.65 ± 0.42	96.04 ± 0.28	109.66 ± 0.42	98.53 ± 0.68
IAF	4.47 ± 0.05	4.38 ± 0.04	102.41 ± 0.04	96.08 ± 0.16	111.58 ± 0.38	99.92 ± 0.30
O-SNF	4.51 ± 0.04	4.39 ± 0.05	99.00 ± 0.29	93.82 ± 0.21	106.08 ± 0.39	94.61 ± 0.83
H-SNF	4.46 ± 0.05	4.35 ± 0.05	$\mathbf{99.00 \pm 0.04}$	$\mathbf{93.77 \pm 0.03}$	$\mathbf{104.62 \pm 0.29}$	$\mathbf{93.82 \pm 0.62}$
T-SNF	4.45 ± 0.04	4.35 ± 0.04	99.33 ± 0.23	93.97 ± 0.13	105.29 ± 0.64	94.92 ± 0.73

Van den Berg et al. 2018

Architectural Taxonomy

Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

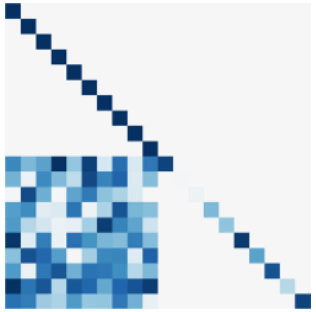
Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

1. Block coupling

NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow

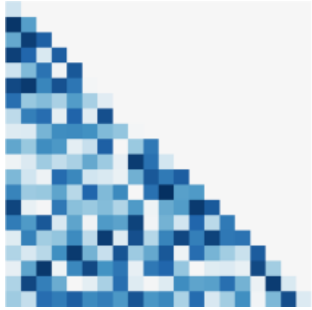
Jacobian



(Lower triangular + structured)

2. Autoregressive

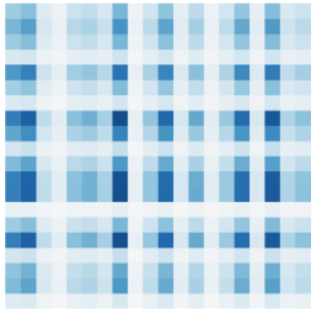
IAF/MAF/NAF
SOS polynomial
UMNN



(Lower triangular)

3. Det identity

Planar/Sylvester flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual Flow
FFJORD

(Arbitrary)

Figures from Ricky Chen

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \operatorname{tr}\left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log\left|\det \frac{\partial f(x)}{\partial x}\right| = \operatorname{tr}\left(\log \frac{\partial f(x)}{\partial x}\right)$$

Jacobi's formula

$$\operatorname{tr}\left(\log\left(\mathbf{I} + \frac{\partial g(x)}{\partial x}\right)\right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \operatorname{tr}\left(\frac{\partial g(x)}{\partial x}^{k}\right)$$

Power series expansion

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \mathrm{tr}\left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

$$\mathrm{tr}\left(\log \left(\mathbf{I} + \frac{\partial g(x)}{\partial x} \right) \right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \mathrm{tr}\left(\frac{\partial g(x)}{\partial x}^{k} \right)$$

Power series expansion

$$\approx \mathbb{E}_v \left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} v^{\top} \left(\frac{\partial g(x)}{\partial x}^{k} \right) v \right]$$

Truncation &
Hutchinson trace estimator

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \operatorname{tr} \left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

$$\operatorname{tr} \left(\log \left(\mathbf{I} + \frac{\partial g(x)}{\partial x} \right) \right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \operatorname{tr} \left(\frac{\partial g(x)}{\partial x} \right)^k$$

Power series expansion

$$\approx \mathbb{E}_v \left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} v^\top \left(\frac{\partial g(x)}{\partial x} \right)^k v \right]$$

Truncation &
Hutchinson trace estimator

Bias

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \operatorname{tr} \left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

$$\operatorname{tr}\left(\log\left(\mathbf{I} + \frac{\partial g(x)}{\partial x}\right)\right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \operatorname{tr}\left(\frac{\partial g(x)}{\partial x}^{k}\right)$$

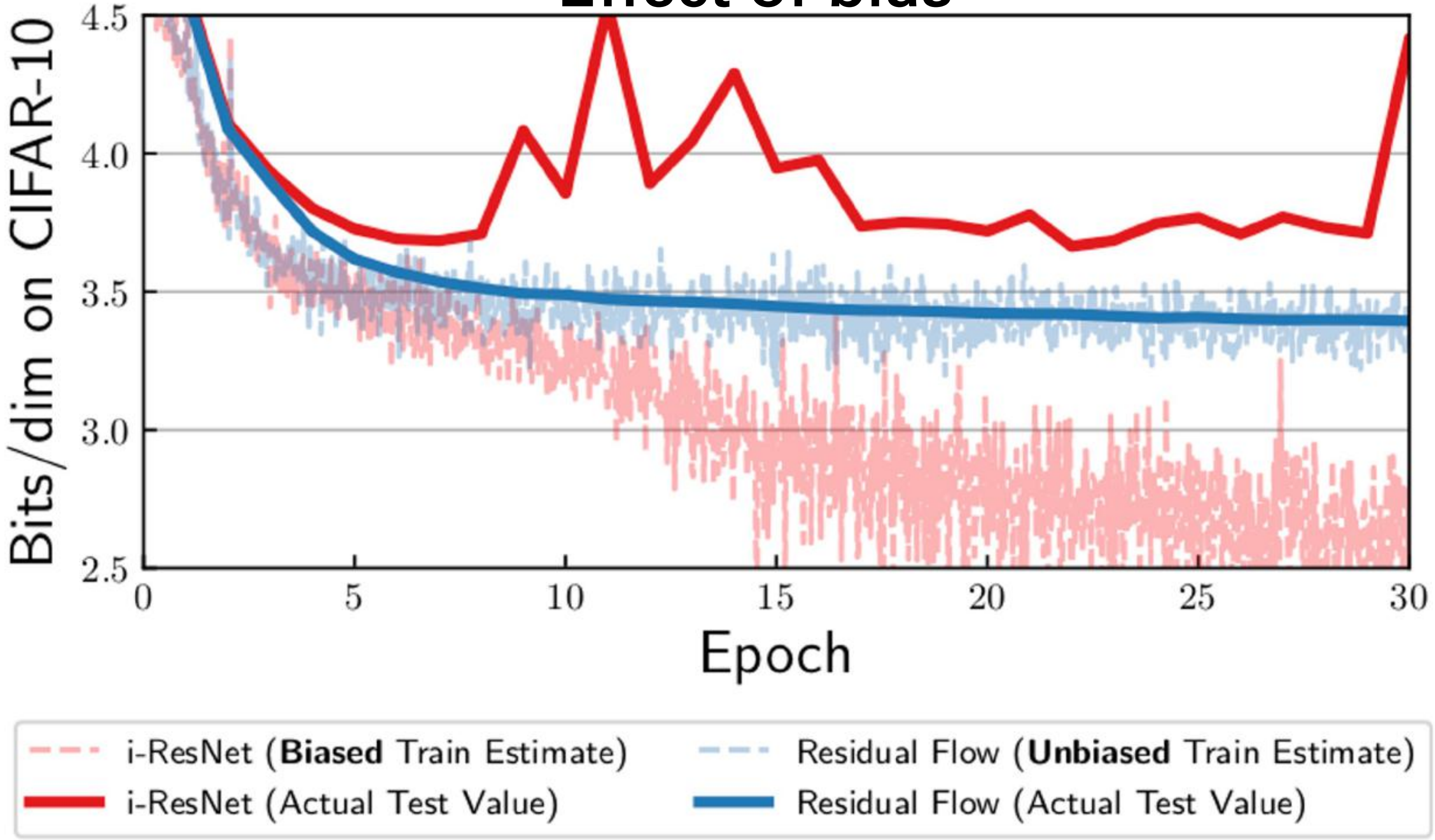
Power series expansion

$$= \mathbb{E}_{v,n}\left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k \cdot \mathbb{P}(N \geq k)} v^{\top} \left(\frac{\partial g(x)}{\partial x}^{k}\right) v\right]$$

Russian roulette estimator & Hutchinson trace estimator

Behrmann et al. 2018

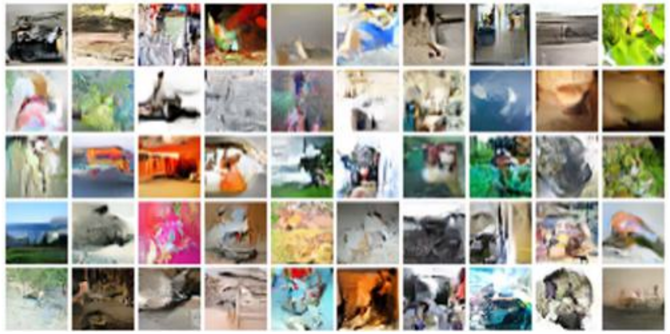
Effect of bias



CelebA samples

Cifar10 samples

Imagenet-32 samples



Figures from Residual Flows for Invertible Generative Modeling by Chen et al., 2019

Next lecture:
Variational Autoencoders
and Denoising Diffusion Models