

# Autoregressive Models

- Pixel RNN/CNN, WaveNet

Hao Dong

Peking University

## Autoregressive Models

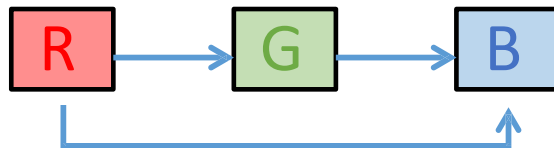
- Assumption of Autoregressive Models
- Fully visible Sigmoid Belief Network (FVSBN)
- Neural Autoregressive Density Estimation (NADE)
- Real-valued Neural Autoregressive Density Estimation (RNADE)
- Autoregressive Autoencoders
- Masked Autoencoder for Distribution Estimation (MADE)
- Recurrent Neural Networks
- Pixel RNN: Pixel CNN, Row LSTM and Diagonal BiLSTM
- Gated PixelCNN
- WaveNet

- Assumption of Autoregressive Models
- Fully visible Sigmoid Belief Network (FVSBN)
- Neural Autoregressive Density Estimation (NADE)
- Real-valued Neural Autoregressive Density Estimation (RNADE)
- Autoregressive Autoencoders
- Masked Autoencoder for Distribution Estimation (MADE)
- Recurrent Neural Networks
- **Pixel RNN: Pixel CNN, Row LSTM and Diagonal BiLSTM**
- Gated PixelCNN
- WaveNet

# Autoregressive Image Modeling

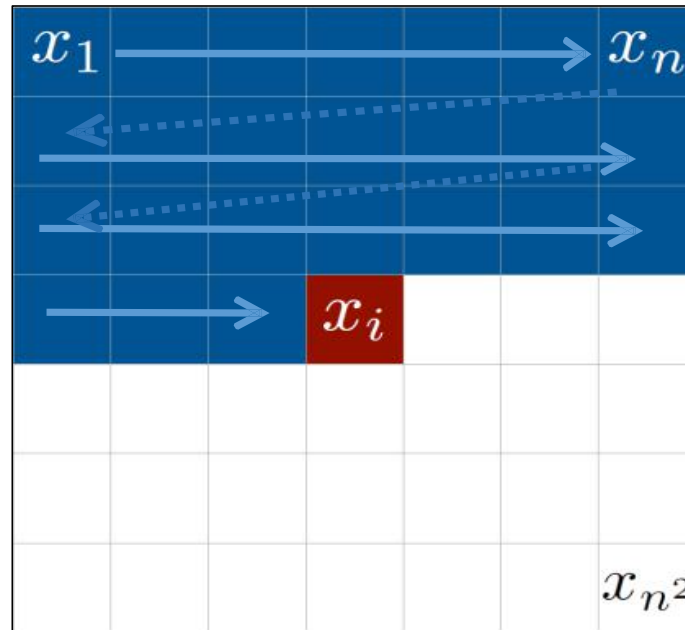
-Autoregressive models train a network that model the conditional distribution of every individual pixel given previous pixels (raster scan orderdependencies).

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



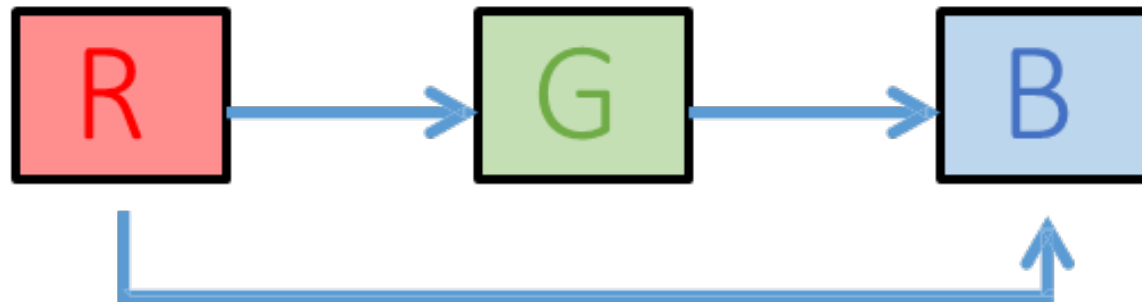
# Autoregressive Image Modeling

**Sequentially predict pixels** rather than predicting the whole image at once (like as GAN, VAE)



# Autoregressive Image Modeling

- For color image, 3 channels are generated successive conditioning, blue given red and green, green given red, and red given only the pixels above and to the left of all channels



---

## Pixel Recurrent Neural Networks

---

**Aäron van den Oord**  
**Nal Kalchbrenner**  
**Koray Kavukcuoglu**

Google DeepMind

AVDNOORD@GOOGLE.COM  
NALK@GOOGLE.COM  
KORAYK@GOOGLE.COM

PixelCNN  
PixelRNN

- Row LSTM
- Diagonal BiLSTM

(ICML 2016 Best Paper)

---

## Conditional Image Generation with PixelCNN Decoders

---

Gated PixelCNN

**Aäron van den Oord**  
Google DeepMind  
avdnoord@google.com

**Nal Kalchbrenner**  
Google DeepMind  
nalk@google.com

**Oriol Vinyals**  
Google DeepMind  
vinyals@google.com

**Lasse Espeholt**  
Google DeepMind  
espeholt@google.com

**Alex Graves**  
Google DeepMind  
gravesa@google.com

**Koray Kavukcuoglu**  
Google DeepMind  
korayk@google.com

WaveNet

## WAVENET: A GENERATIVE MODEL FOR RAW AUDIO

**Aäron van den Oord**

**Sander Dieleman**

**Heiga Zen<sup>†</sup>**

**Karen Simonyan**

**Oriol Vinyals**

**Alex Graves**

**Nal Kalchbrenner**

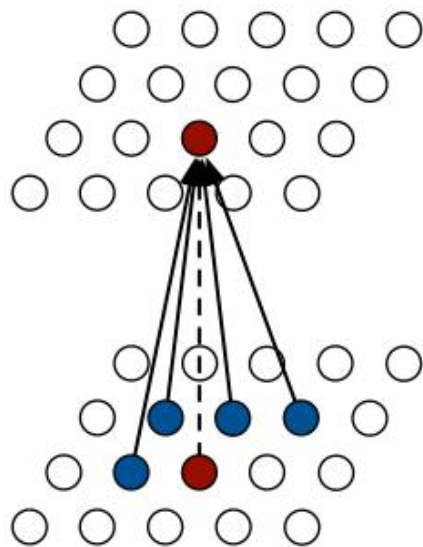
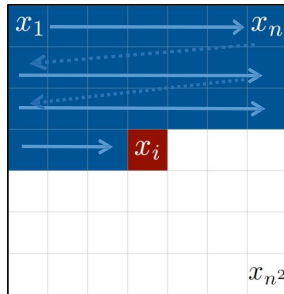
**Andrew Senior**

**Koray Kavukcuoglu**

- Assumption of Autoregressive Models
- Fully visible Sigmoid Belief Network (FVSBN)
- Neural Autoregressive Density Estimation (NADE)
- Real-valued Neural Autoregressive Density Estimation (RNADE)
- Autoregressive Autoencoders
- Masked Autoencoder for Distribution Estimation (MADE)
- Recurrent Neural Networks
- Pixel RNN: **Pixel CNN**, Row LSTM and Diagonal BiLSTM
- Gated PixelCNN
- WaveNet



## PixelCNN Network Structure

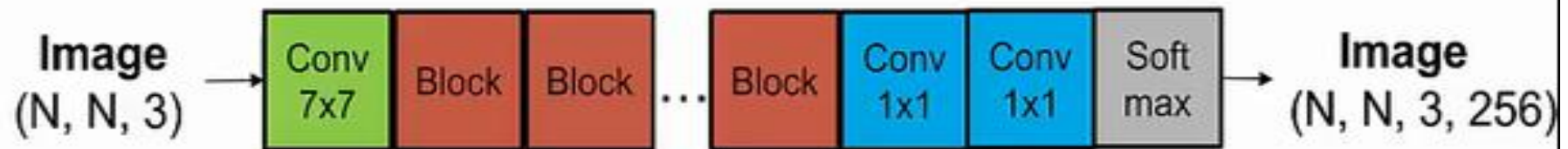
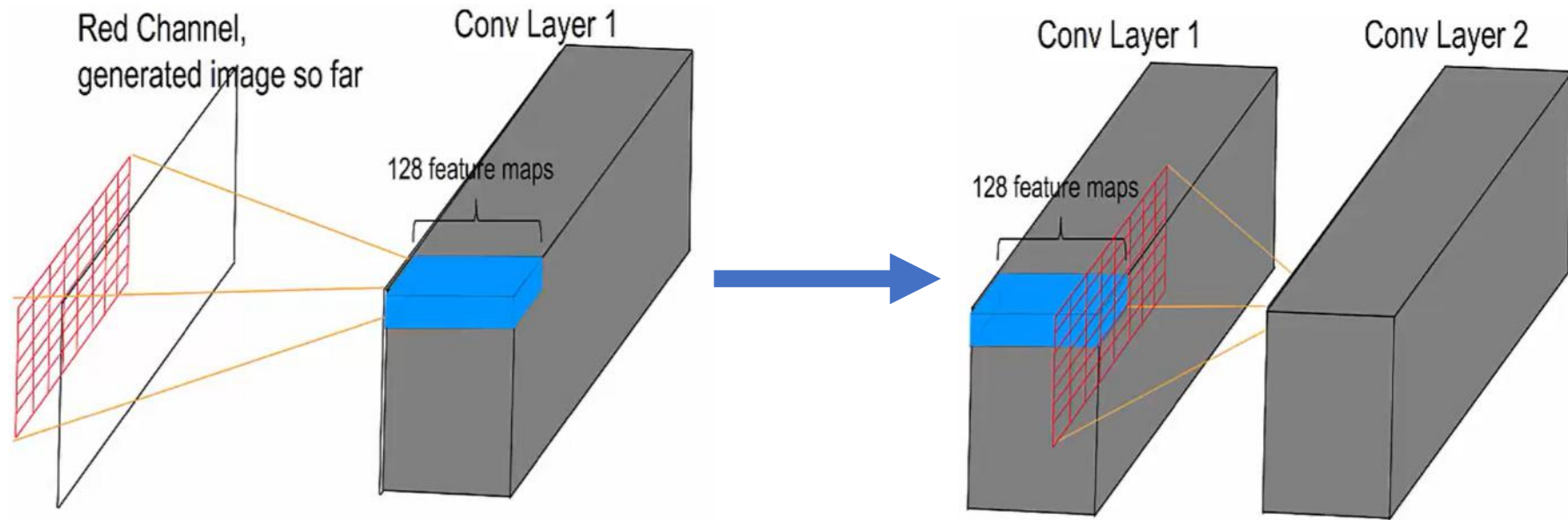


PixelCNN

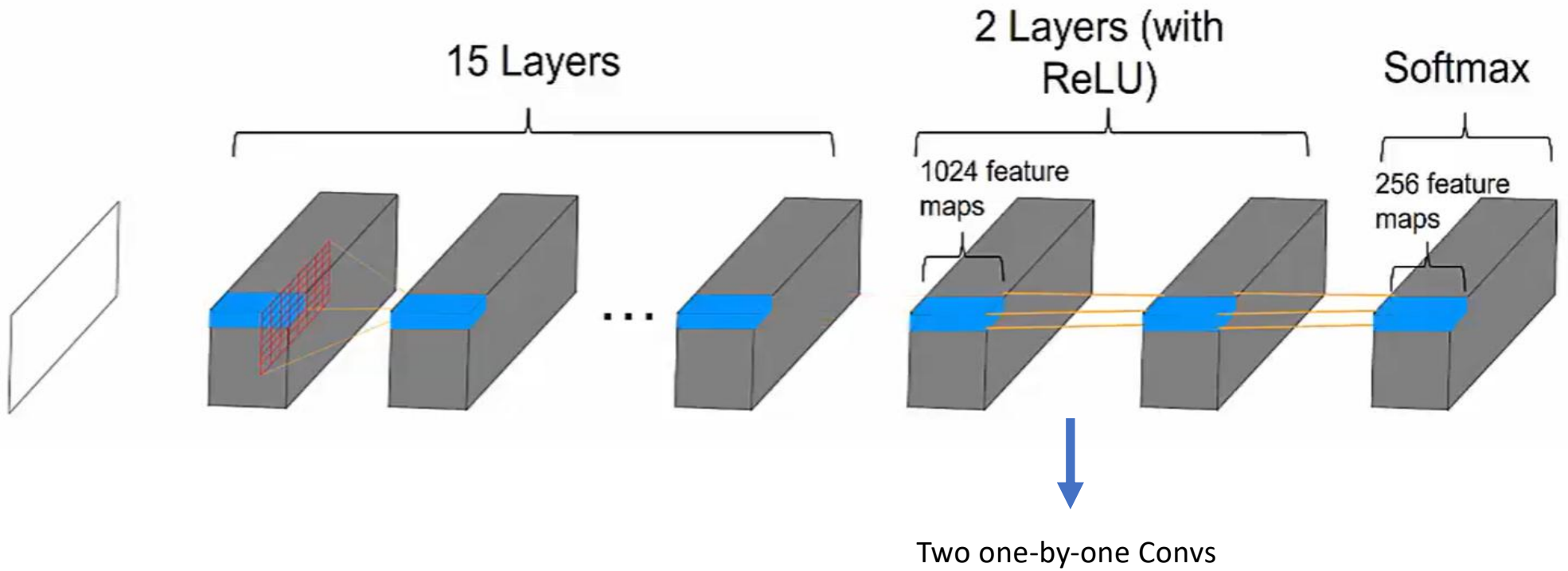
Goal:

- Use the neighbor pixels to predict the new pixel

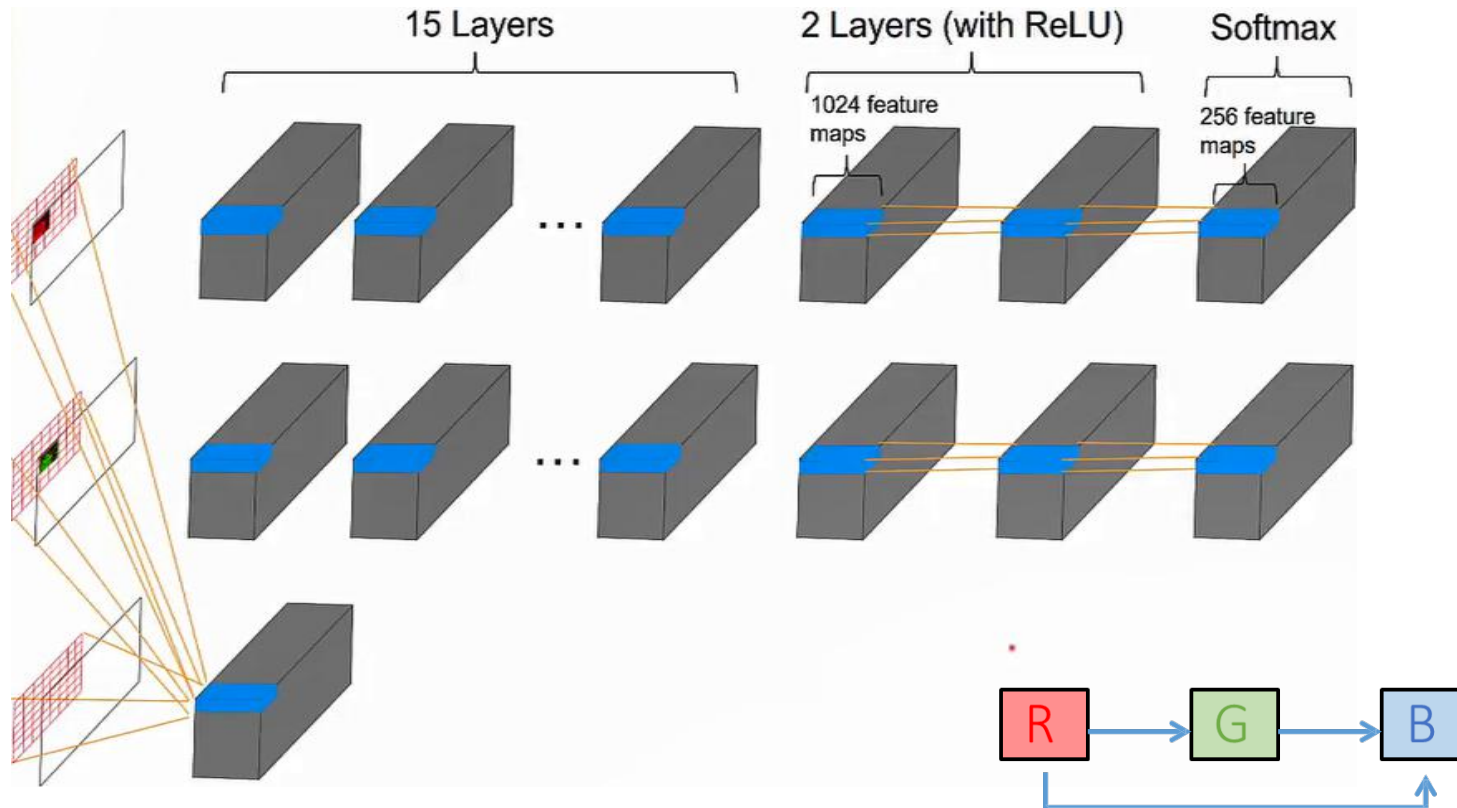
# PixelCNN Network Structure



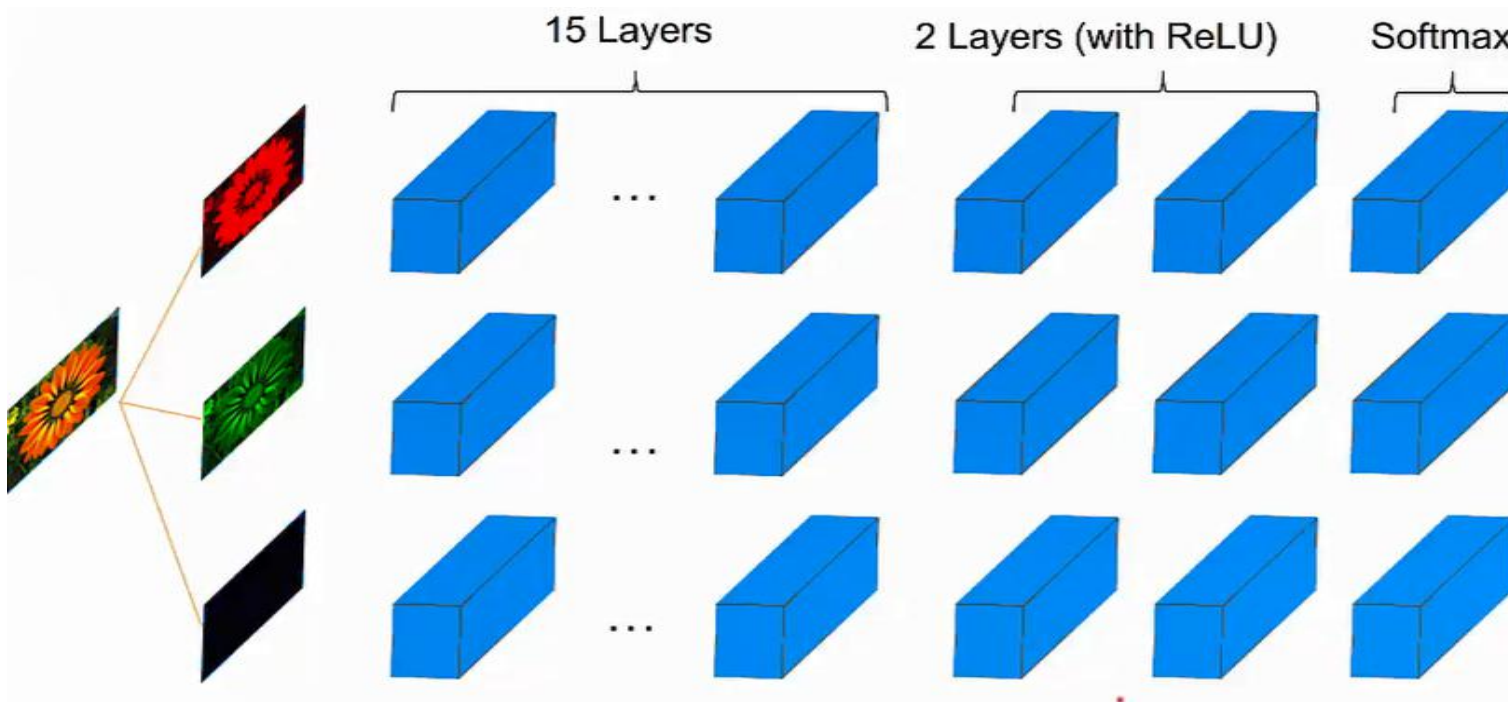
# PixelCNN Network Structure



# PixelCNN Network Structure: Repeat for 3 times for RGB

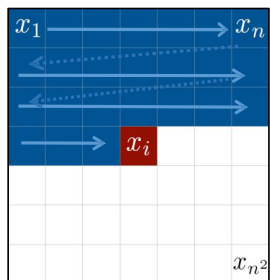


When training, something differs...

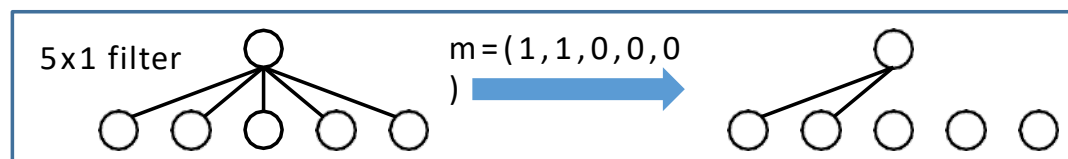


## Details of “Masked Convolution” & “Blind Spot”

- To generate next pixel, the model can only condition on the previously generated pixels.
- Then, to make sure CNN can only use information about pixels above and to the left of current pixel, the filters of the convolution need to be masked.



### Case 1 D

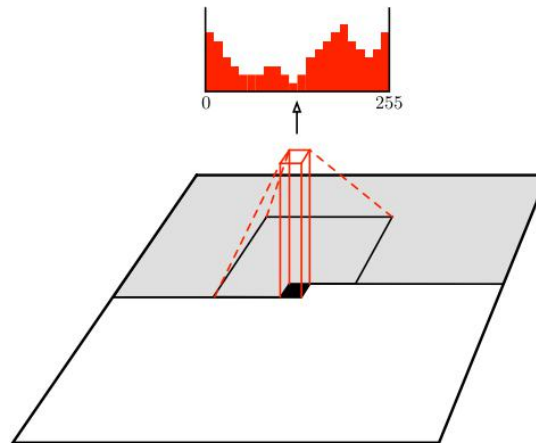


- Right figure shows 5x1 convolutional filters after multiplying them by mask.
- The filters connecting the input layer to the first hidden layer are in this case multiplied by  $m=(1,1,0,0,0)$ , to ensure the model is causal.

# Details of “Masked Convolution” & “BlindSpot”

## Case 2D

- In case of 2D, PixelCNNs have a **blind spot** in the receptive field that cannot be used to make predictions.
- Rightmost figure shows the growth of the masked receptive field. (3 layered network with 3x3 conv filters)



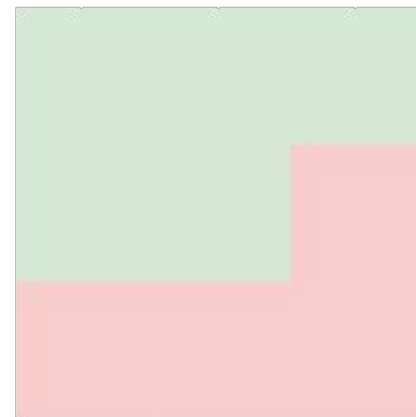
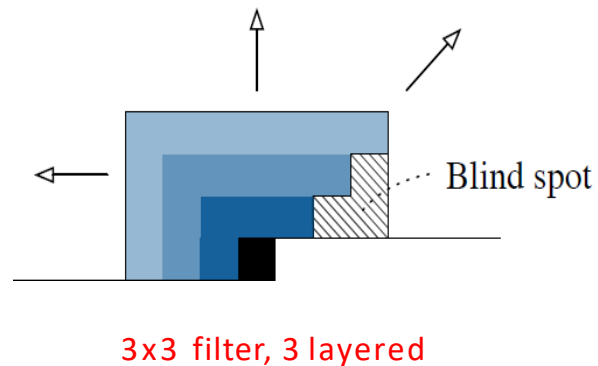
5x5 filter

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

# Details of “Masked Convolution” & “BlindSpot”

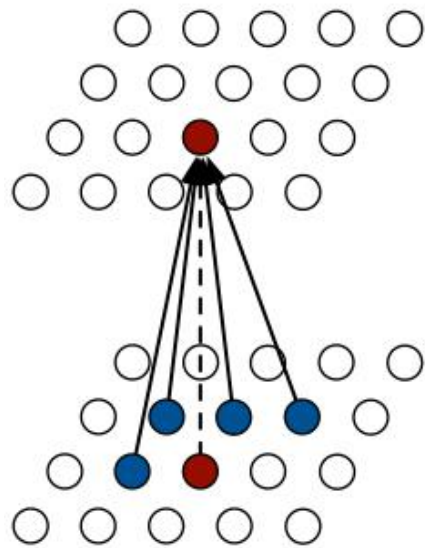
## Case 2D

- In case of 2D, PixelCNNs have a **blind spot** in the receptive field that cannot be used to make predictions.
- Rightmost figure shows the growth of the masked receptive field. (3 layered network with 3x3 conv filters)





## Pixel CNN Drawbacks



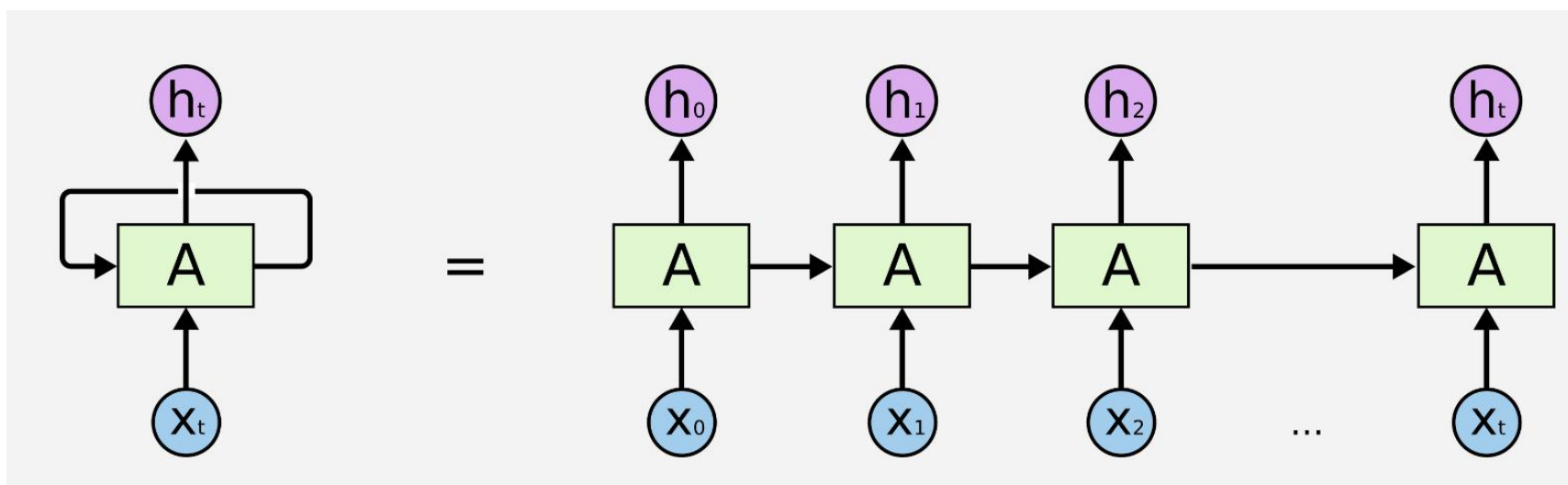
PixelCNN

Drawbacks:

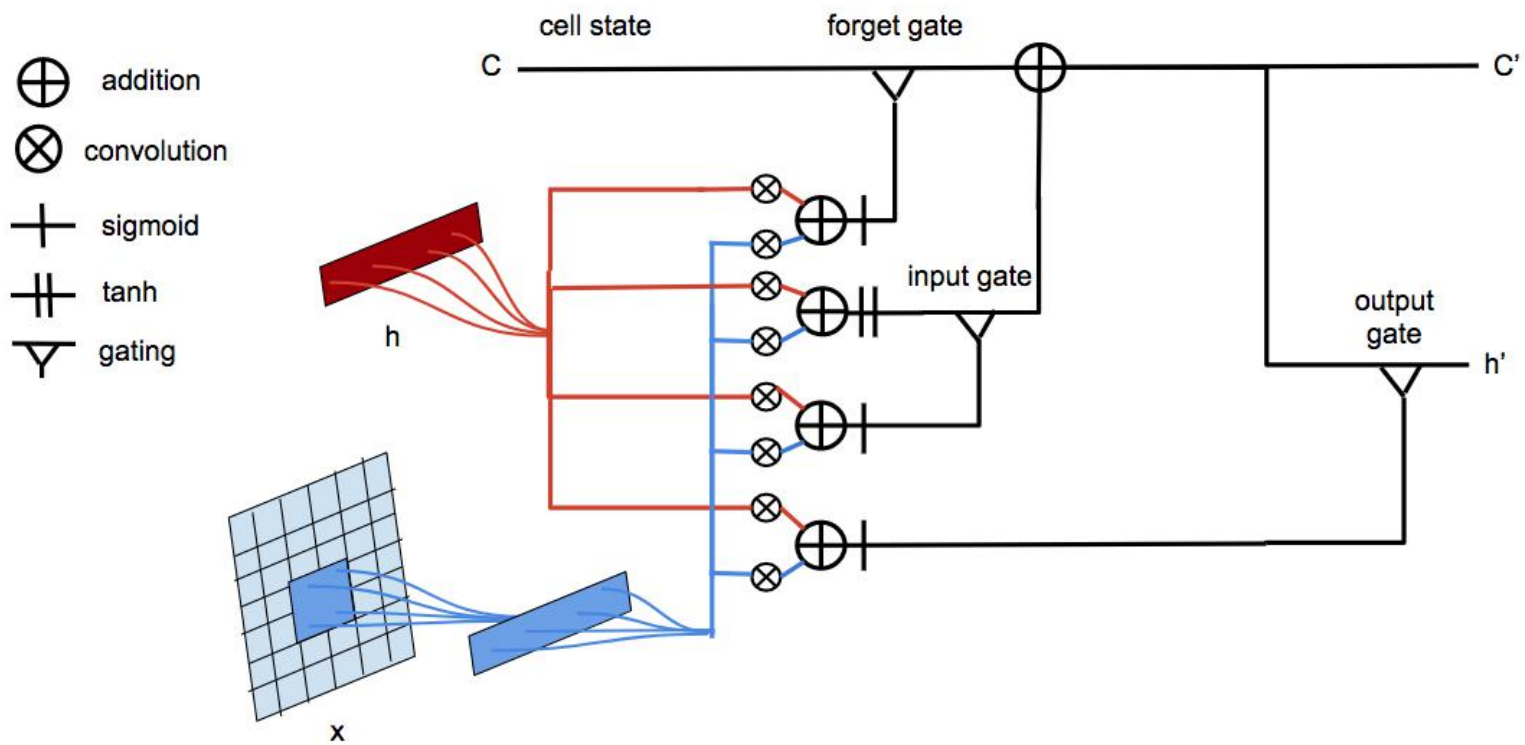
- Sequential generation is SLOW
- Blind spot problem

- Assumption of Autoregressive Models
- Fully visible Sigmoid Belief Network (FVSBN)
- Neural Autoregressive Density Estimation (NADE)
- Real-valued Neural Autoregressive Density Estimation (RNADE)
- Autoregressive Autoencoders
- Masked Autoencoder for Distribution Estimation (MADE)
- Recurrent Neural Networks
- **Pixel RNN**: Pixel CNN, Row LSTM and Diagonal BiLSTM
- Gated PixelCNN
- WaveNet

# Recap RNN...



# Convolutional LSTM...



# With the Convolutional LSTM, we change LSTM Equations from ... into ...

## LSTM Equations

$$\begin{aligned}
 i &= \sigma(x_i U^i + \mathbf{h}_{i-1} W^i) \\
 f &= \sigma(x_i U^f + \mathbf{h}_{i-1} W^f) \\
 o &= \sigma(x_i U^o + \mathbf{h}_{i-1} W^o) \\
 g &= \tanh(x_i U^g + \mathbf{h}_{i-1} W^g) \\
 c_i &= c_{i-1} \circ f + g \circ i \\
 \mathbf{h}_i &= \tanh(c_i) \circ o
 \end{aligned}$$

Gates - Control how much information is allowed through

States - Hold information about all time steps up till now {0, i, ..., i-1, i}

## LSTM Equations

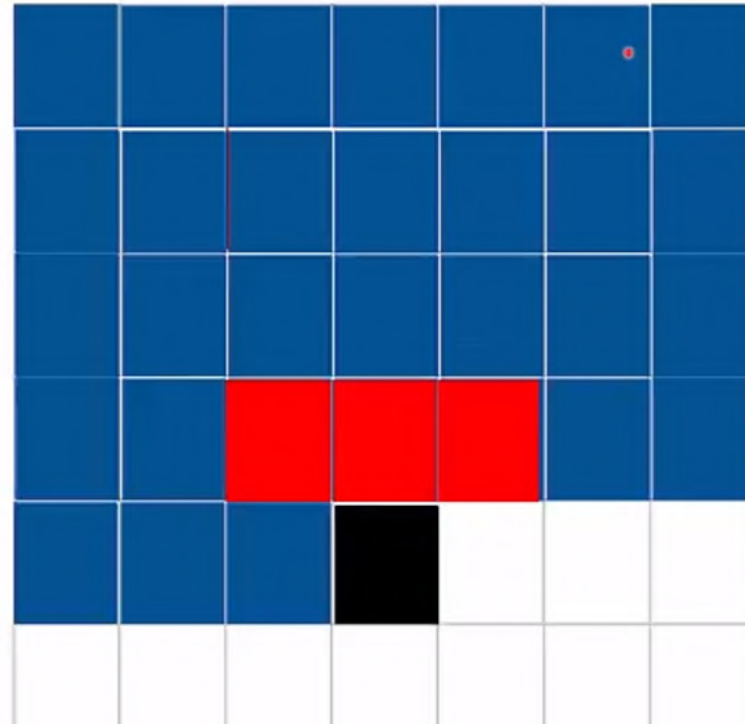
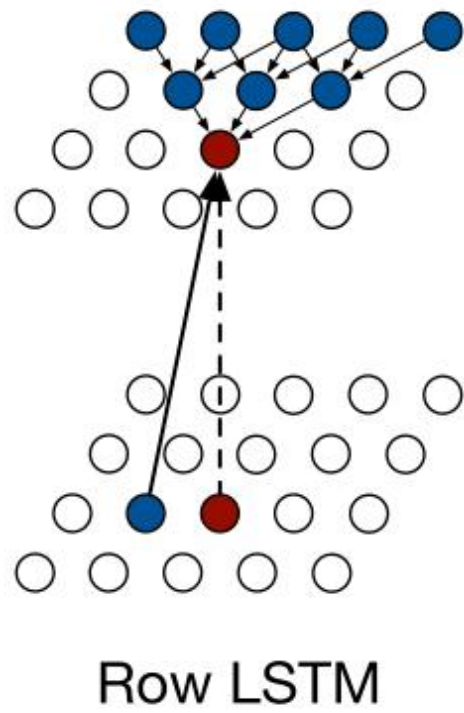
$$\begin{aligned}
 i &= \sigma(x_i \cancel{U^i} + \mathbf{h}_{i-1} \cancel{W^i}) \\
 f &= \sigma(x_i \cancel{U^f} + \mathbf{h}_{i-1} \cancel{W^f}) \\
 o &= \sigma(x_i \cancel{U^o} + \mathbf{h}_{i-1} \cancel{W^o}) \\
 g &= \tanh(x_i \cancel{U^g} + \mathbf{h}_{i-1} \cancel{W^g}) \\
 c_i &= c_{i-1} \circ f + g \circ i \\
 \mathbf{h}_i &= \tanh(c_i) \circ o
 \end{aligned}$$

Like Convolutional LSTM - replaced fully-connected layer with convolutional layer

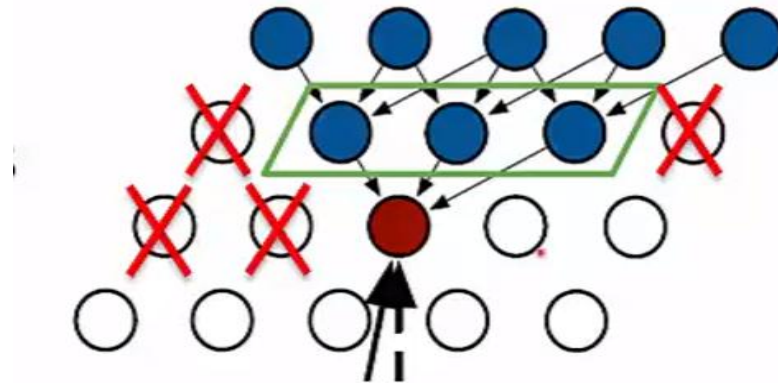
$$[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] = \sigma(\mathbf{K}^{ss} \otimes \mathbf{h}_{i-1} + \mathbf{K}^{is} \otimes \mathbf{x}_i)$$

$$\begin{aligned}
 \mathbf{c}_i &= \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i \\
 \mathbf{h}_i &= \mathbf{o}_i \odot \tanh(\mathbf{c}_i)
 \end{aligned}$$

# PixelRNN 1: Row LSTM



So the receptive field...

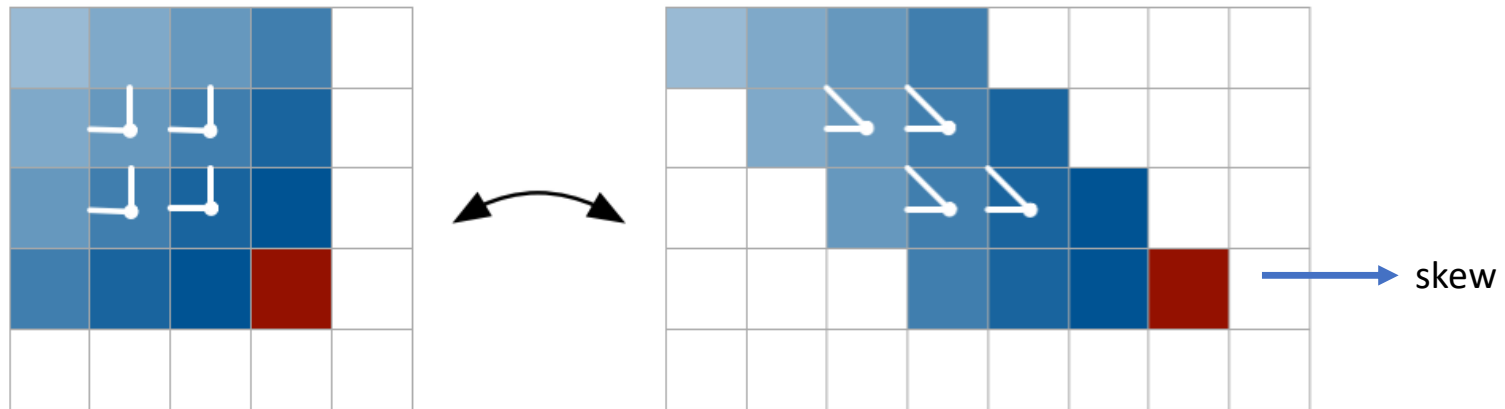


Triangular receptive  
field

- Assumption of Autoregressive Models
- Fully visible Sigmoid Belief Network (FVSBN)
- Neural Autoregressive Density Estimation (NADE)
- Real-valued Neural Autoregressive Density Estimation (RNADE)
- Autoregressive Autoencoders
- Masked Autoencoder for Distribution Estimation (MADE)
- Recurrent Neural Networks
- **Pixel RNN: Pixel CNN, Row LSTM and Diagonal BiLSTM**
- Gated PixelCNN
- WaveNet



## PixelRNN2: Diagonal BiLSTM

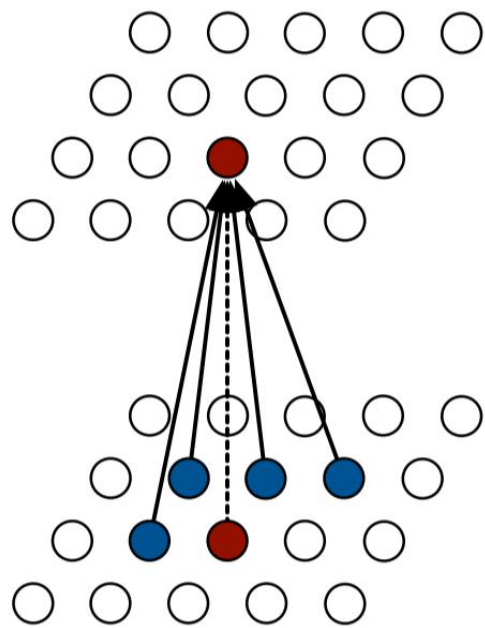


In the Diagonal BiLSTM, to allow for parallelization along the diagonals, the input map is skewed by offsetting each row by one position with respect to the previous row. When the spatial layer is computed left to right and column by column, the output map is shifted back into the original size. The convolution uses a kernel of size  $2 \times 1$ .

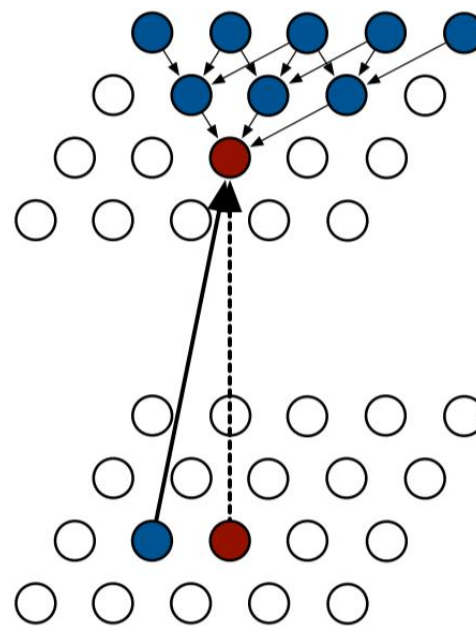
# Diagonal BiLSTM: why “Bi”?



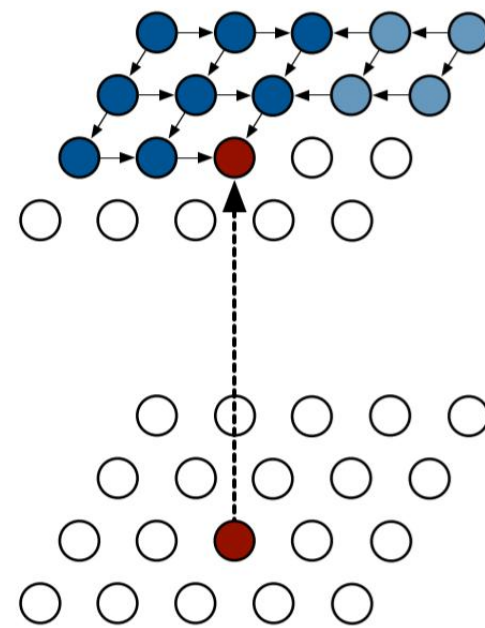
# Comparison between the 3 network above...



PixelCNN



Row LSTM

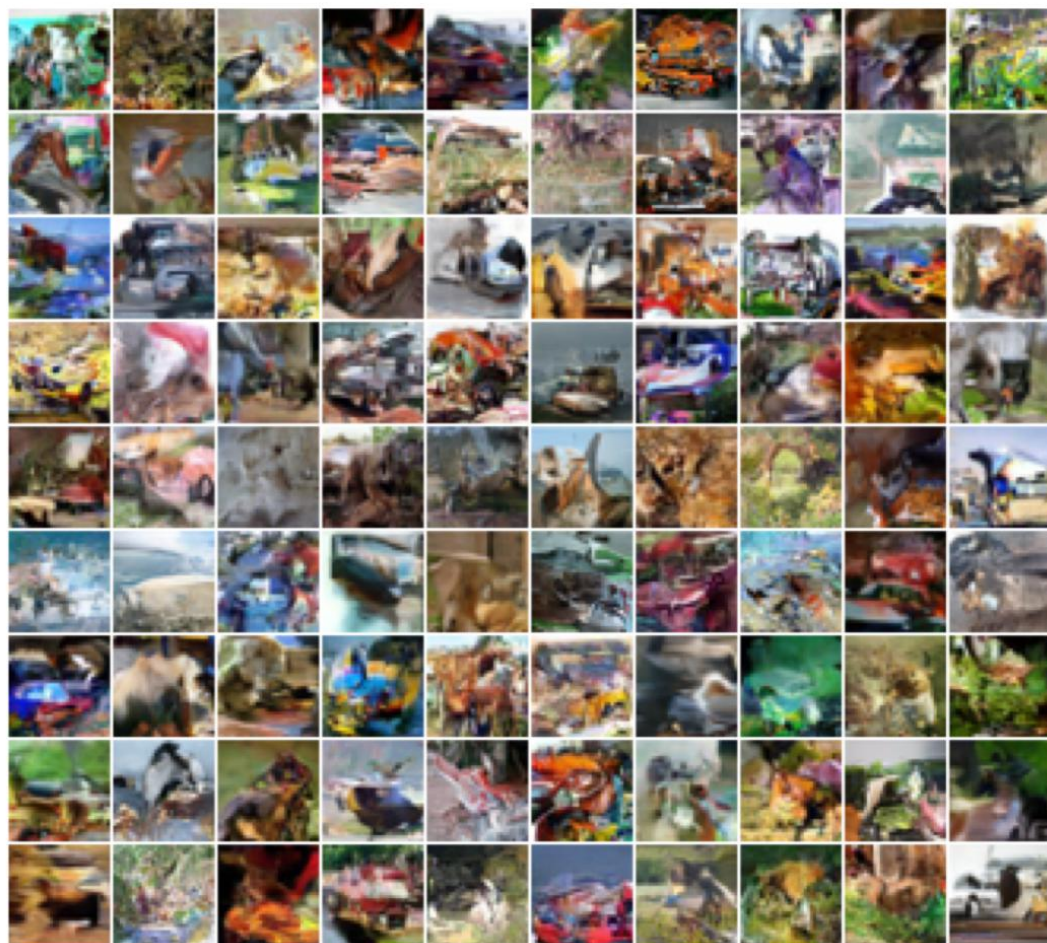


Diagonal BiLSTM

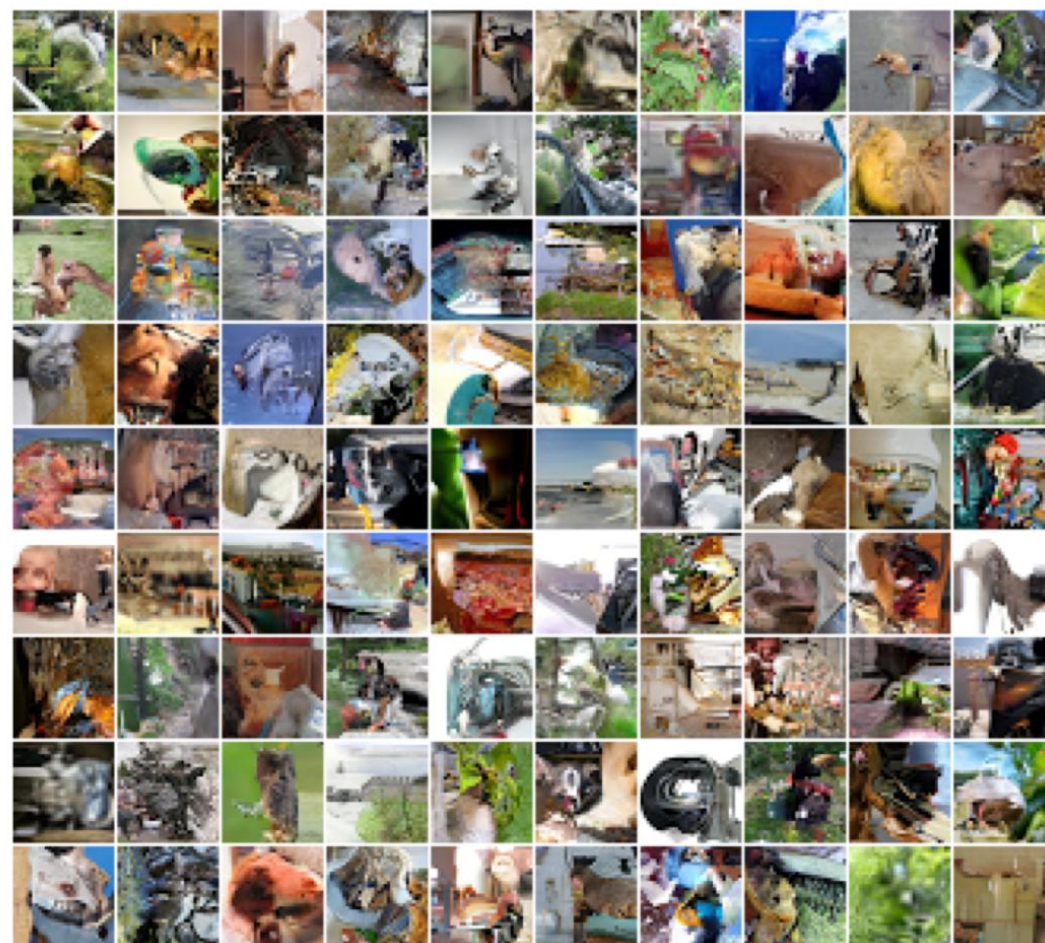


# Results of PixelRNN

CIFAR-10



ImageNet 32x32





# Results of PixelRNN



*Figure 1.* Image completions sampled from a PixelRNN.

## PixelRNN vs. PixelCNN

	PixelRNN	PixelCNN
Pros.	<ul style="list-style-type: none"> <li>Effectively handles long-range dependencies</li> <li><b>Good performance</b></li> </ul>	Convolutions are easier to parallelize ⇒ <b>Much faster</b> to train
Cons.	<ul style="list-style-type: none"> <li>Each state needs to be computed sequentially.</li> <li><b>Computationally expensive</b></li> </ul>	Bounded receptive field ⇒ <b>Inferior performance</b> <b>Blind spot problem</b> (due to the masked convolution) needs to be eliminated.

- Assumption of Autoregressive Models
- Fully visible Sigmoid Belief Network (FVSBN)
- Neural Autoregressive Density Estimation (NADE)
- Real-valued Neural Autoregressive Density Estimation (RNADE)
- Autoregressive Autoencoders
- Masked Autoencoder for Distribution Estimation (MADE)
- Recurrent Neural Networks
- Pixel RNN: Pixel CNN, Row LSTM and Diagonal BiLSTM
- **Gated PixelCNN**
- WaveNet

## Gated PixelCNN

An improved version of PixelCNN, major improvements are as follows :

- Removal of blind spots in the receptive field by combining **the horizontal stack** and **the vertical stack**.
- Replacement of the ReLU activations between the masked convolutions in the original PixelCNN with the **gated activation unit**.



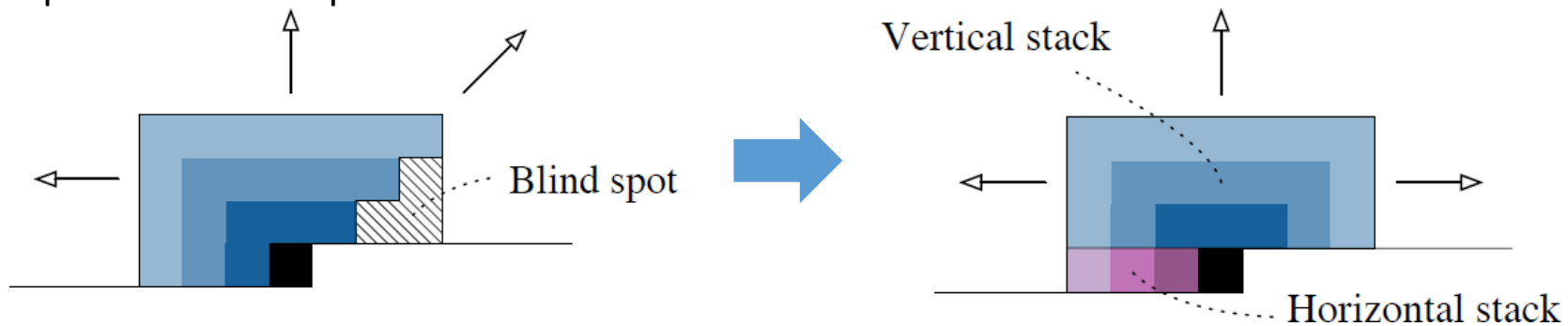
## Gated PixelCNN

- Given a latent vector, they modeled the conditional distribution of images, **Conditional PixelCNN**.
  - conditioning on class-label
  - conditioning on embedding from trained model
- From a convolutional auto-encoder, they replaced the deconvolutional decoder with conditional PixelCNN, named **PixelCNN Auto-Encoders**

## First improvement:

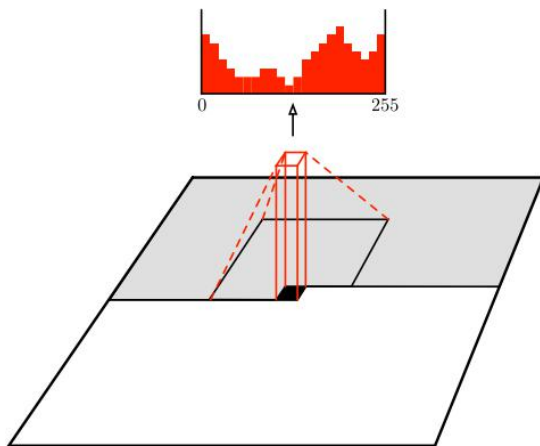
### Horizontal stack and Vertical stack

- The removal of blind spots in the receptive field are important for PixelCNN's performance, because the blind spot can cover as much as a quarter of the potential receptive field.



- The **vertical stack** conditions on all rows above the current row.
- The **horizontal stack** conditions on current row.

# First improvement: Horizontal stack and Vertical stack



Mask for PixelCNN

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

They cause blind spot

Mask for Gated PixelCNN

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

Vertical stack

Why they can be 1?

1	1	0	0	0
---	---	---	---	---

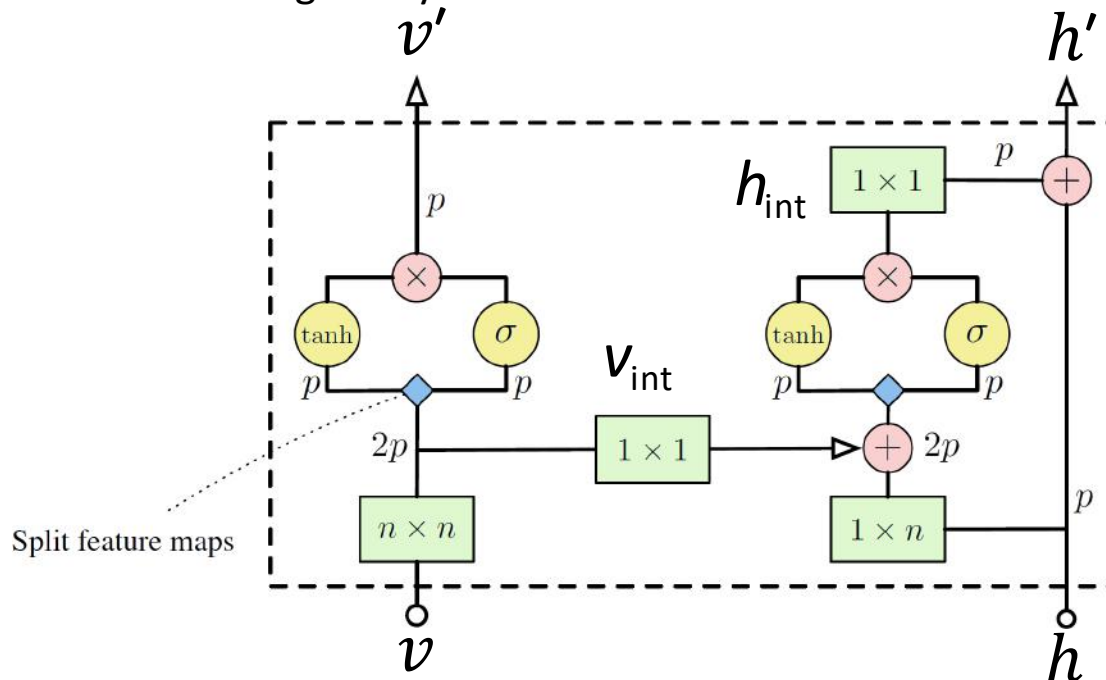
Horizontal stack

## Second improvement: Gated Activation and Architecture

- Gated activation unit:  $\mathbf{y} = \tanh(W_{k,f} * \mathbf{x}) \odot \sigma(W_{k,g} * \mathbf{x})$

( $\sigma$ : sigmoid,  $k$ : number of layer,  $\odot$ : element-wise product,  $*$ : convolutional operator)

- Single layer block of a GatedPixelCNN



- Masked convolutions are shown in green.
- Element-wise operations are shown in red.
- Convolutions with  $W_f$ ,  $W_g$  are combined into a single operation shown in blue.

$p = \#$ feature maps

$v$  = vertical activation maps

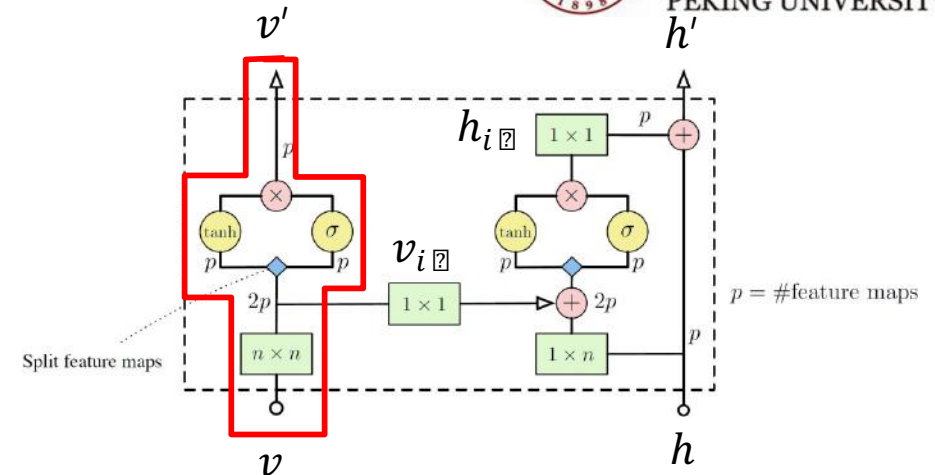
$h$  = horizontal activation maps

# Details of Gated PixelCNN architecture

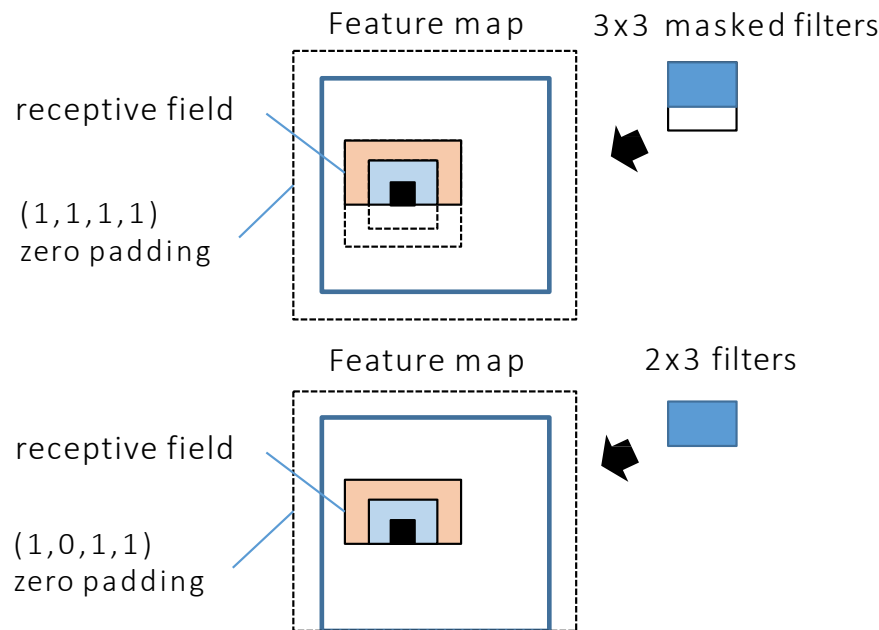
- Break down operations into four steps.
- ① Calculate vertical feature maps  
...  $n \times n$  convolutions are calculated with gated activation.

Input:  $v$  (= input image if 1<sup>st</sup> layer)

Output:  $v'$  (ex.  $n = 3$ )



Two types of equivalent implementation:



Next problem:

In this case,  $(i, j)$ th pixel depends on  $(i, j+k)$ th (future) pixels

Solution:

Shift down vertical feature maps when to feed into horizontal stack.

# Details of Gated PixelCNN architecture

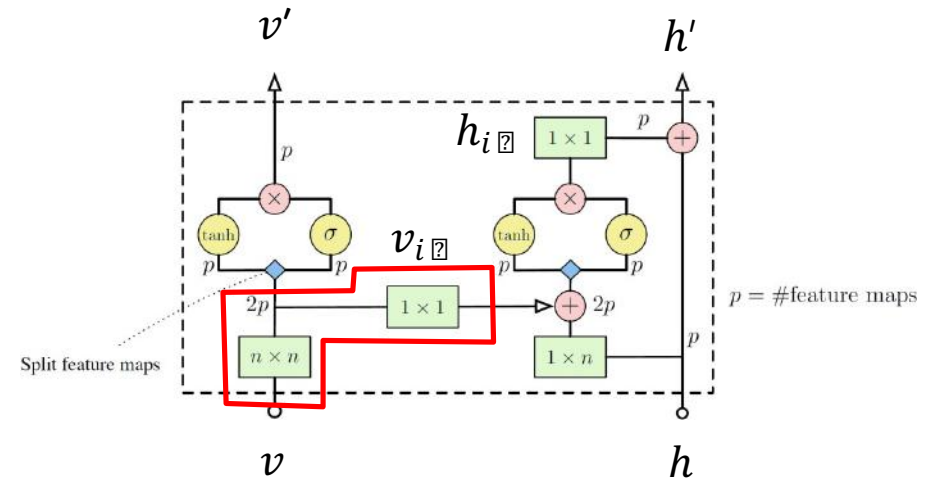


## ② Feed vertical maps into horizontal stack

1.  $n \times n$  masked convolution
2. shifting down operation (as below)
3.  $1 \times 1$  convolution

Input:  $v$  (= input image if 1<sup>st</sup> layer)

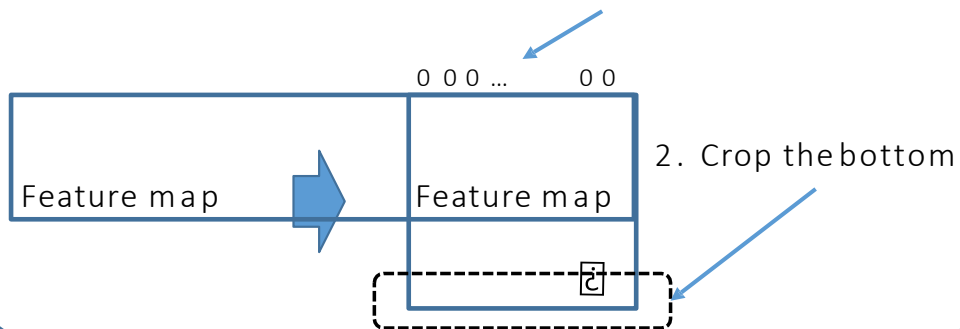
Output:  $v_{int}$



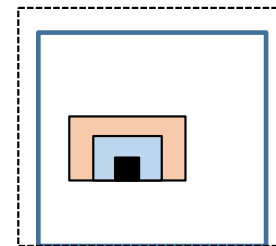
Shift down vertical feature maps when to feed into horizontal stack.

Left operations can be interpreted as below.

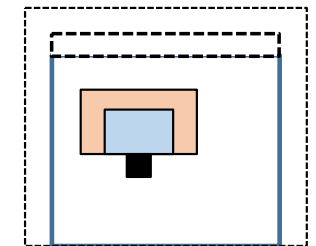
1. Add zero padding on the top



violate causality



ensure causality

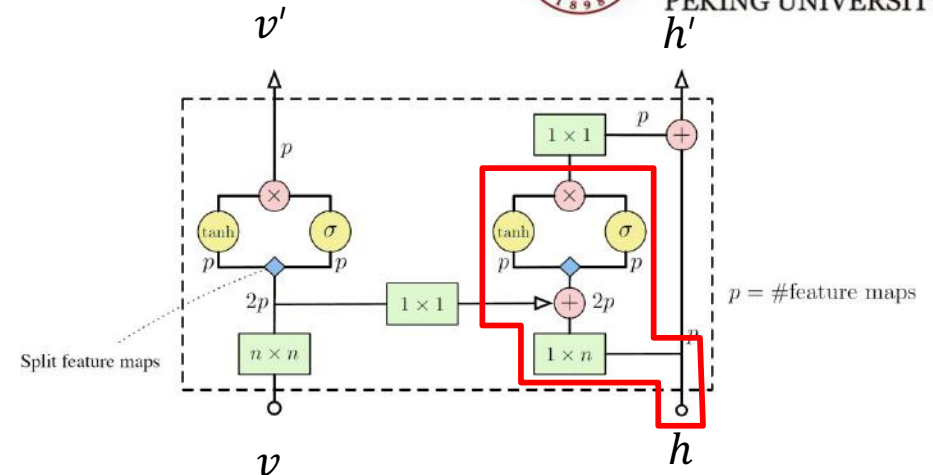


# Details of Gated PixelCNN architecture

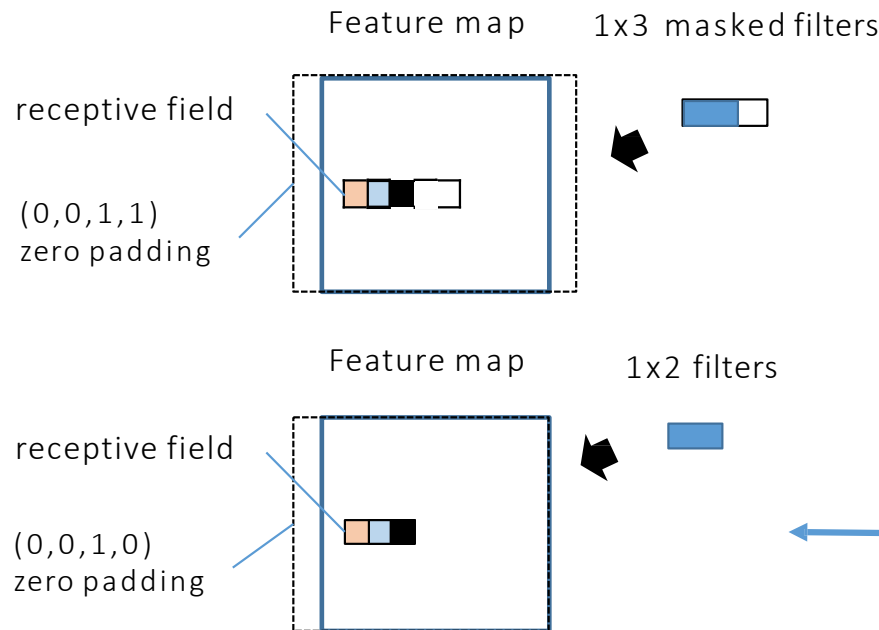
- ③ Calculate horizontal feature maps  
 ...  $1 \times n$  convolutions are calculated with gated activation.  
 (vertical maps are added before activation.)

Input:  $v_{int}$  ,  $h$  (input image if 1<sup>st</sup> layer)

Output:  $h_{int}$  (ex.  $n = 3$ )



Two types of equivalent implementation :



Next problem:  
 ➤ Mask A vs Mask B

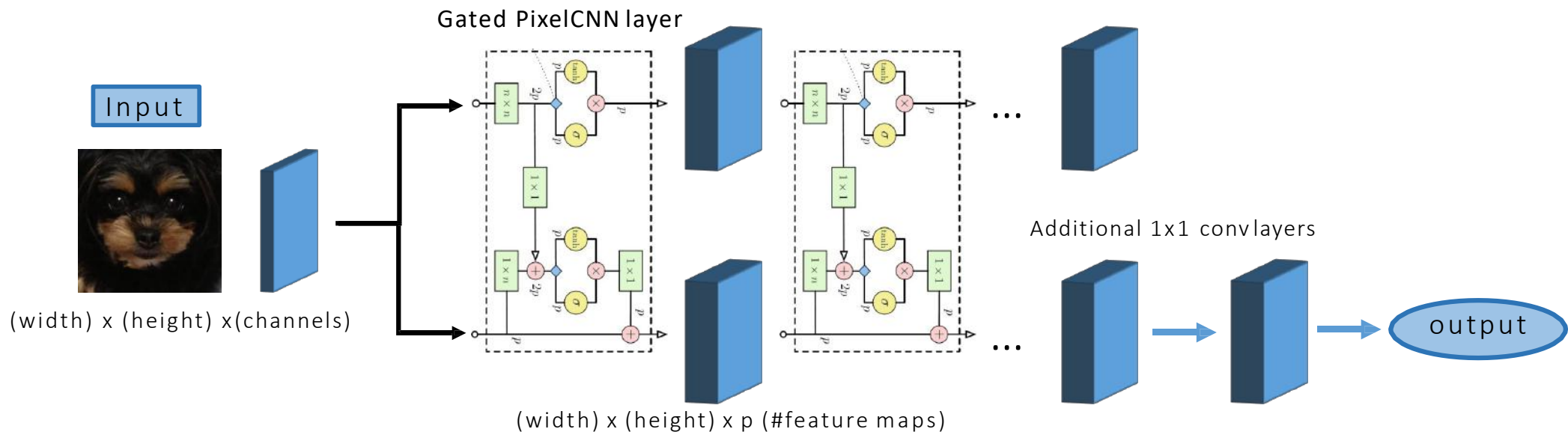
- Mask A (restrict connection from itself) is applied to only to the first convolution.
- Mask B (allow connection from itself) is applied to all the subsequent convolution.

# Output layer and whole architecture

## ■ Output layer

- Using a softmax on **discrete pixel values** ([0-255] = 256 way) instead of a mixture density approach. (same approach as PixelRNN)
- Although without prior information about the meaning or relations of the 256 color categories, the distributions predicted by the model are meaningful.

## ■ Whole architecture





## Third improvements:

### Conditional PixelCNN & PixelCNN AutoEncoder

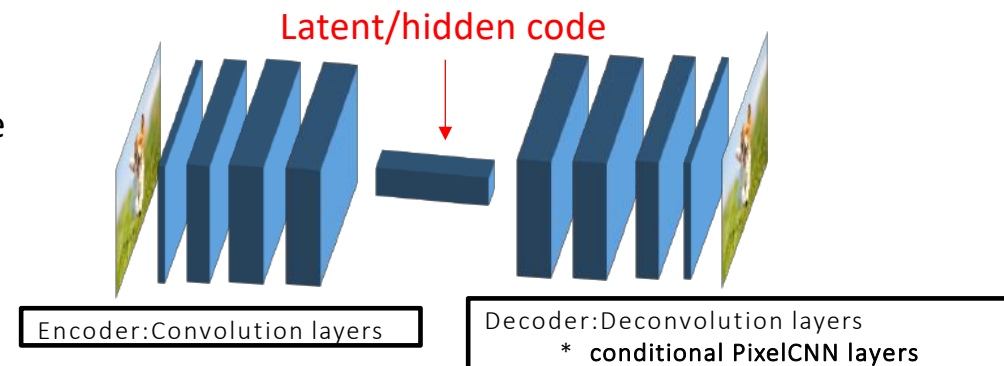
#### ■ Conditional PixelCNN

	original	conditional
Model	$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i   x_1, \dots, x_{i-1}).$	$p(\mathbf{x}   \mathbf{h}) = \prod_{i=1}^{n^2} p(x_i   x_1, \dots, x_{i-1}, \mathbf{h})$
Gated activation unit	$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x}) \odot \sigma(W_{k,g} * \mathbf{x})$	$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x} + \underline{V_{k,f}^T \mathbf{h}}) \odot \sigma(W_{k,g} * \mathbf{x} + \underline{V_{k,g}^T \mathbf{h}})$

- They modeled the conditional distribution by adding terms that depend on  $\mathbf{h}$  to the activations before the nonlinearities

#### ■ PixelCNN AutoEncoder

- From a convolutional auto-encoder, they replaced the deconvolutional decoder with conditional PixelCNN



# Experimental Results (Unconditional)

- Score: Negative log-likelihood score (bits/dim)

- Data: CIFAR-10 dataset

Model	NLL Test (Train)
Uniform Distribution: [30]	8.00
Multivariate Gaussian: [30]	4.70
NICE: [4]	4.48
Deep Diffusion: [24]	4.20
DRAW: [9]	4.13
Deep GMMs: [31, 29]	4.00
Conv DRAW: [8]	3.58 (3.57)
RIDE: [26, 30]	3.47
PixelCNN: [30]	3.14 (3.08)
PixelRNN: [30]	3.00 (2.93)
<b>Gated PixelCNN:</b>	<b>3.03 (2.90)</b>

- Gated PixelCNN outperforms the PixelCNN by 0.11 bits/dim, which has a very significant effect on the visual quality, and close to the performance of PixelRNN

- Data: ImageNet dataset

32x32	Model	NLL Test (Train)
	Conv Draw: [8]	4.40 (4.35)
	PixelRNN: [30]	3.86 (3.83)
	<b>Gated PixelCNN:</b>	<b>3.83 (3.77)</b>
64x64	Model	NLL Test (Train)
	Conv Draw: [8]	4.10 (4.04)
	PixelRNN: [30]	3.63 (3.57)
	<b>Gated PixelCNN:</b>	<b>3.57 (3.48)</b>

- Gated PixelCNN outperforms PixelRNN.
- Achieve similar performance to the PixelRNN in **less than half the training time.**

# Experimental Results

## ■ Conditioning on ImageNet classes

- Given a one-hot encoding  $h_i$ , for the  $i$ -th class, model  $p(x | h)_i$



Lhasa Apso (dog)



Sorrel horse

(part of results.)

## ■ Conditioning on Portrait Embeddings

- Embeddings are taken from top layer of a conv network trained on a large database of portraits from Flickr images.
- After the supervised net was trained,  $\{x:\text{image}, h:\text{embedding}\}$  tuples are taken and trained conditional PixelCNN to model  $p(x | h)$
- Given a new image of a person that was not in the training set, they computed  $h$  and generate new portraits of same person.



- And experimented with reconstructions conditioned on linear interpolations between embeddings of pairs of images.

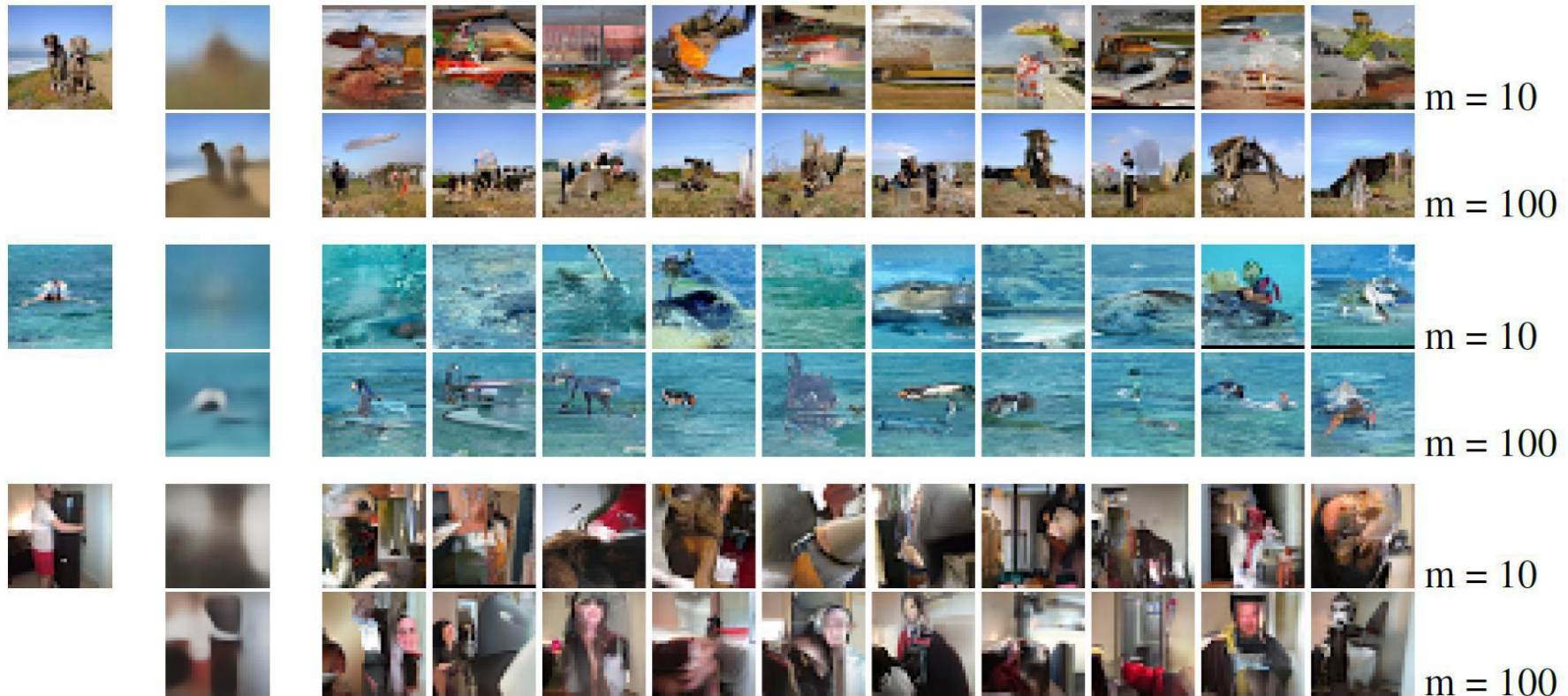




# Experimental Results (PixelCNN Auto-Encoder)

■ Data: 32x32 ImageNet patches

(m: dimensional bottleneck)



(Left to right: original image, reconstruction by auto-encoder, conditional samples from PixelCNN auto-encoder)

- Assumption of Autoregressive Models
- Fully visible Sigmoid Belief Network (FVSBN)
- Neural Autoregressive Density Estimation (NADE)
- Real-valued Neural Autoregressive Density Estimation (RNADE)
- Autoregressive Autoencoders
- Masked Autoencoder for Distribution Estimation (MADE)
- Recurrent Neural Networks
- Pixel RNN: Pixel CNN, Row LSTM and Diagonal BiLSTM
- Gated PixelCNN
- **WaveNet**

# Motivation



The Autoregressive model (e.g. PixelCNN) has been very successful.

→ What about **voice**?

I want to do that with a CNN which is more efficient than an RNN.

# WaveNet



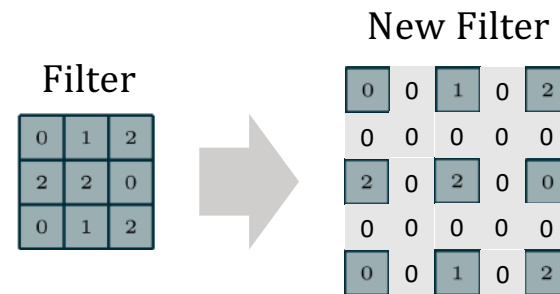
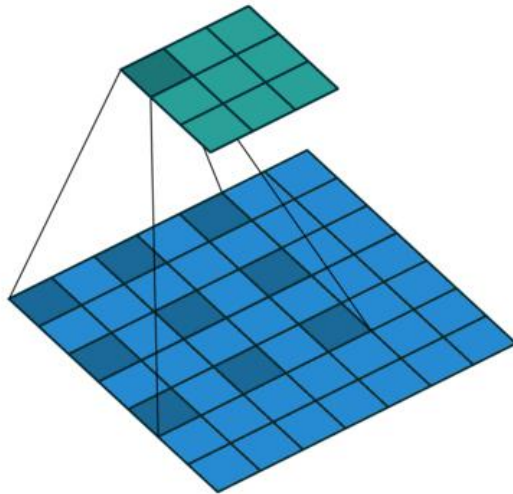
This network is similar to PixelCNN.

Here are the contributions:

- Unprecedented quality speech synthesis.
- Efficient architecture with a large receptive field using dilated convolution
- (also voice recognition)

## Recap: Dilated convolution

Roughly speaking, if you really want to use a filter with a large kernel size, you can use this to get a large kernel and approximate results without increasing the amount of computation.



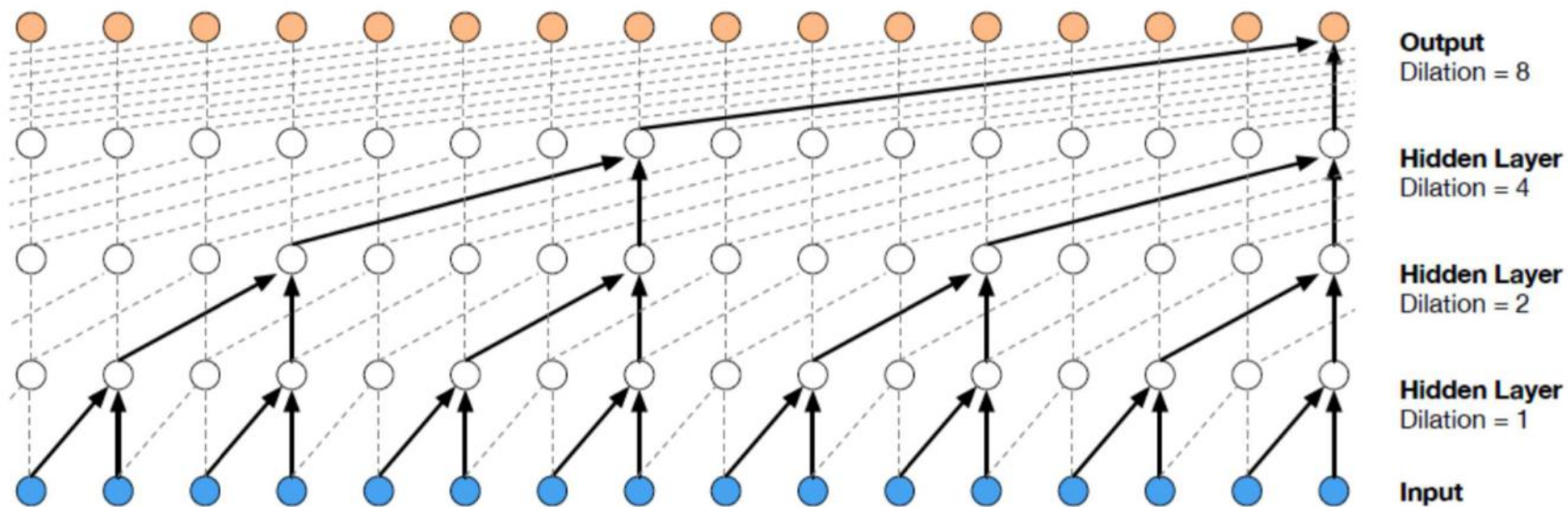
Dilation rate =  $2 \times 2$

Receptive field :  $3 \times 3 \rightarrow 5 \times 5$

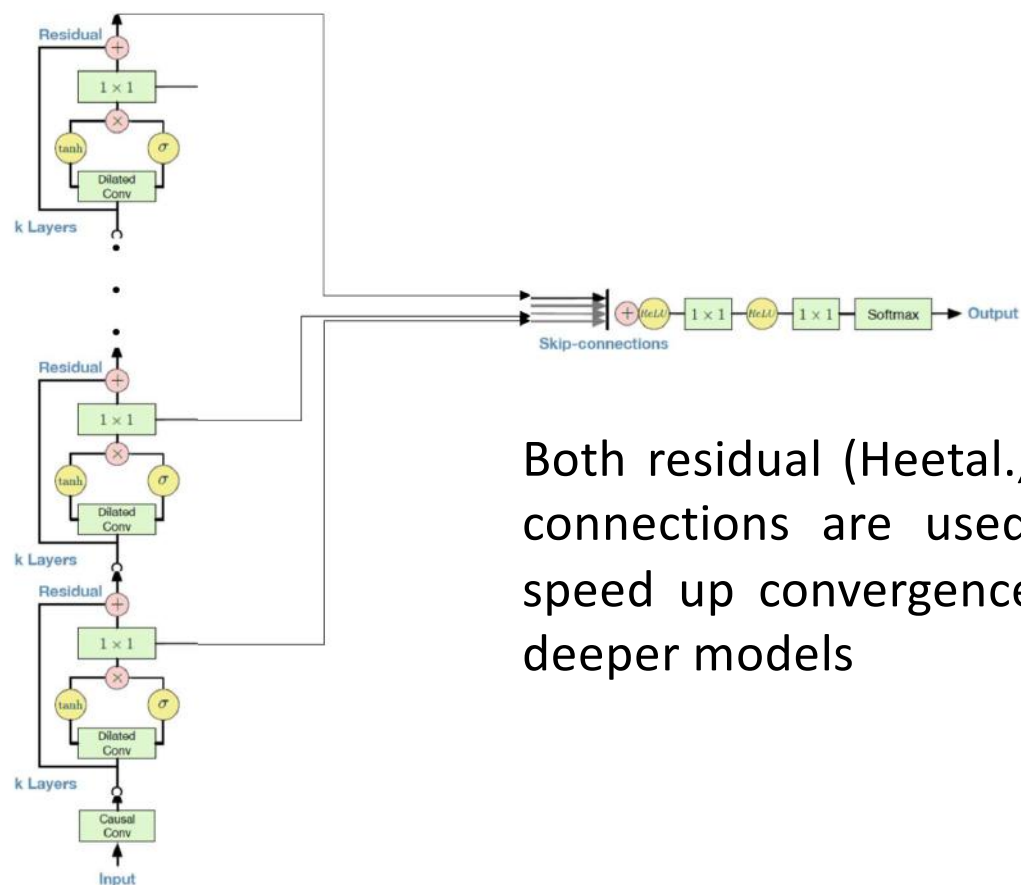


# Stacked dilated causal convolution

This is a conceptual diagram of the expansion of the receptive field.

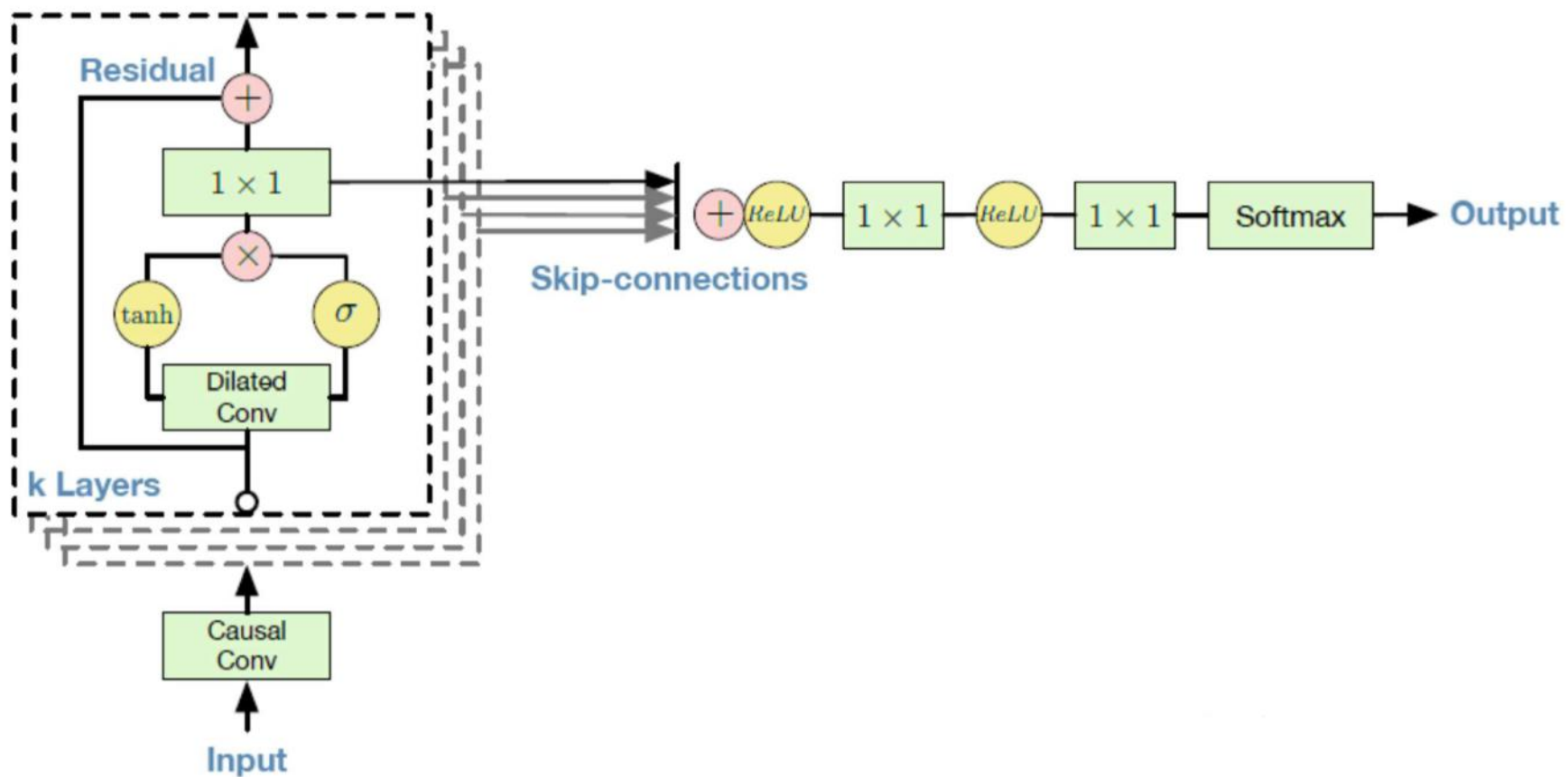


# Entire architecture



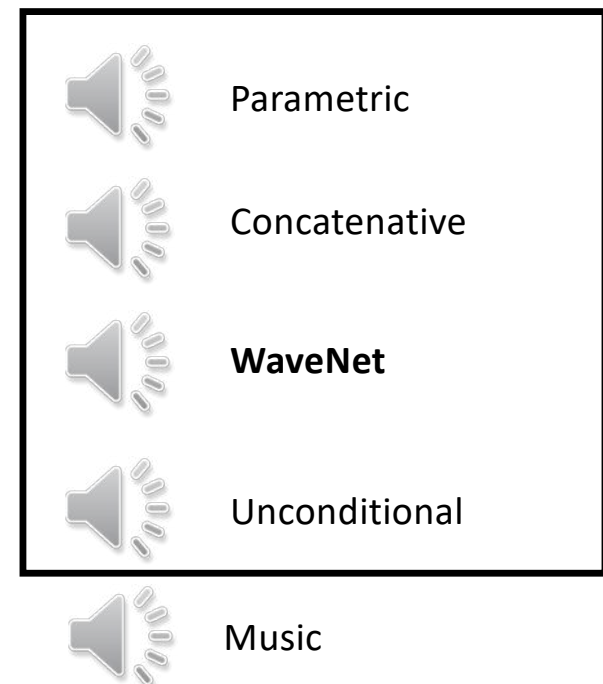
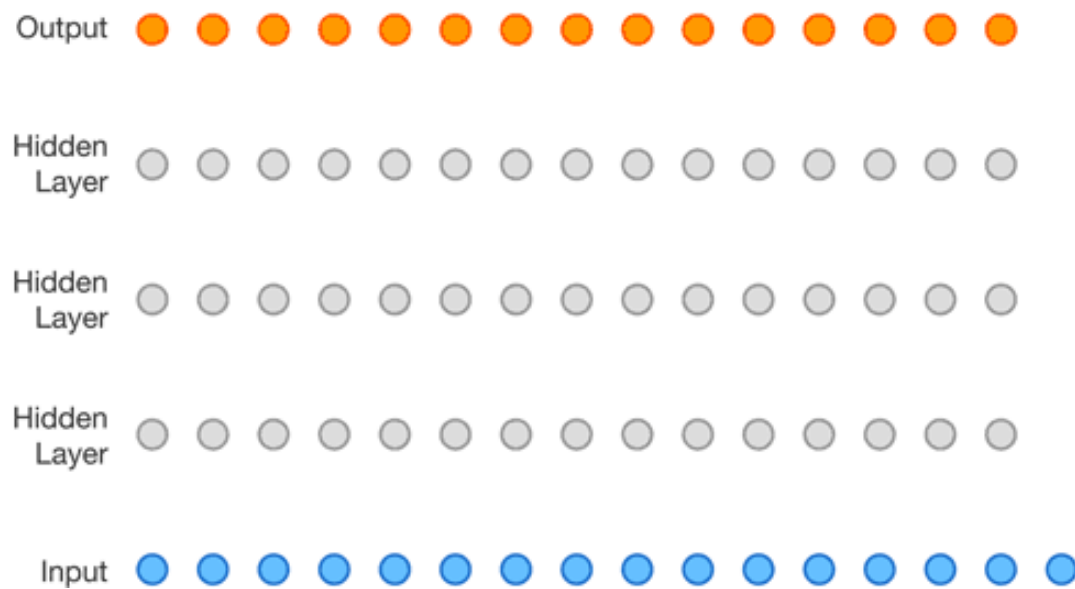
Both residual (Heetal.,2015) and parameterized skip-connections are used throughout the network, to speed up convergence and enable training of much deeper models

# Residual block



# Conditional Generation

$$P(X = \textit{speech} | Y = \textit{sentence})$$



## Summary of Autoregressive Models

Easy to sample from

Sample  $x_0 \sim p(x_0)$

Sample  $x_1 \sim p(x_1 | x_0 = x_0)$

-

Easy to compute probability  $p(x = x)$

Compute  $p(x_0 = x_0)$

Compute  $p(x_1 = x_1 | x_0 = x_0)$

Multiply together (sum their logarithms)

-

Ideally, can compute all these terms in parallel for fast training

Easy to extend to continuous variables. For example, can choose Gaussian conditionals  $p(x_t | x_{<t}) = N(\mu_{\vartheta}(x_{<t}), \Sigma_{\vartheta}(x_{<t}))$  or mixture of logistics

No natural way to get features, cluster points, do unsupervised learning

## References

Pixel Recurrent Neural Networks:

<https://arxiv.org/pdf/1609.03499.pdf>

WaveNet: A Generative Model for Raw Audio:

<https://arxiv.org/pdf/1601.06759.pdf>

Conditional Image Generation with PixelCNN Decoders:

<https://arxiv.org/pdf/1606.05328v2.pdf>

# Thanks