Conditional Image Generation with PixelCNN Decoders

Aäron van den Oord
Nal Kalchbrenner
Oriol Vinyals
Lasse Espeholt
Alex Graves
Koray Kavukcuoglu

Google DeepMind

Yohei Sugawara
BrainPad Inc.

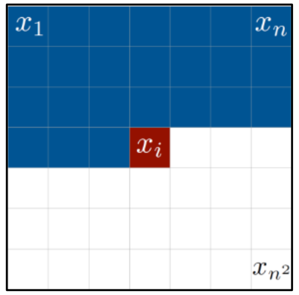
NIPS2016読み会(@Preffered Networks)

January 19, 2017

Preview

【Background: Autoregressive Image Modeling】

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



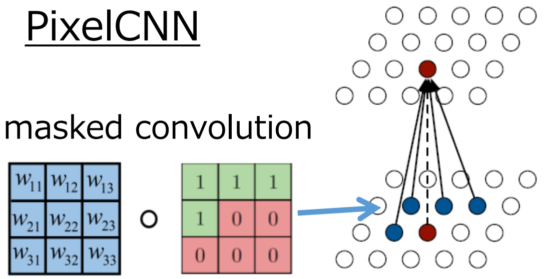
- Pixel dependencies = Raster scan order
- Autoregressive models **sequentially predict pixels** rather than predicting the whole image at once (like as GAN, VAE)

【Previous work: Pixel Recurrent Neural Networks】

- CNN based model and RNN based models are proposed.

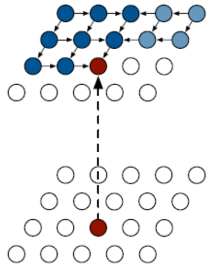
PixelCNN

masked convolution


w_{11} w_{12} w_{13}
w_{21} w_{22} w_{23}
w_{31} w_{32} w_{33}
∘
1 1 1
1 0 0
0 0 0

Pros: easier to parallelize
Cons: bounded dependency

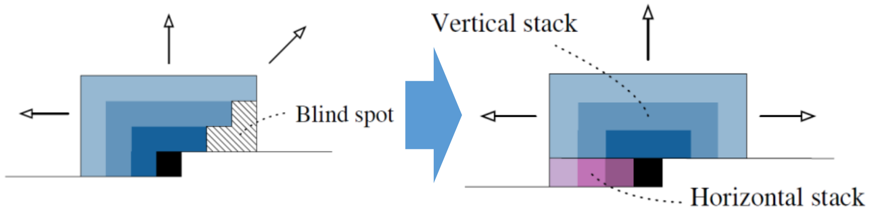
PixelRNN
(Diagonal BiLSTM)



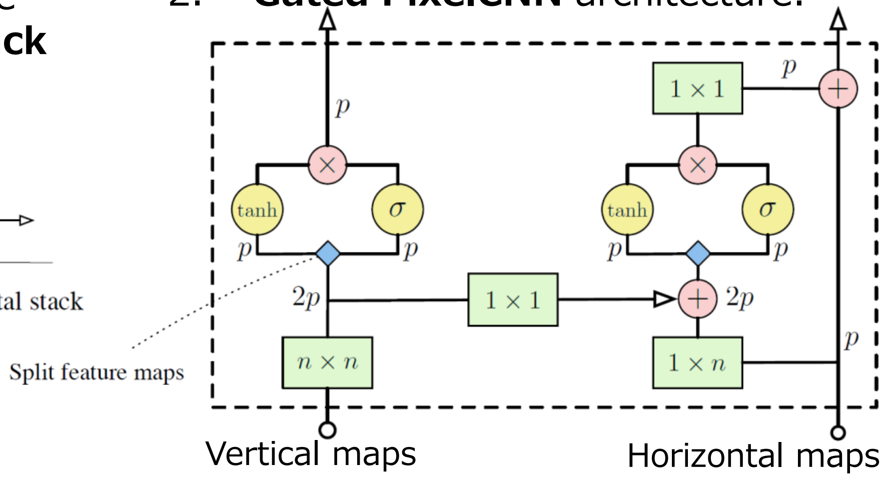
Pros: full dependency field
Cons: sequential training.

【Proposed approach: Gated PixelCNN】

1. Removal of blind spots in the receptive field by combining **the horizontal stack** and **the vertical stack**.



2. **Gated PixelCNN** architecture.



3. Conditional PixelCNN

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

conditioning on [one-hot encoding of the class-label / embedding from trained model

4. PixelCNN AutoEncoder

Encoder = Convolutional layers
Decoder = **Conditional PixelCNN layers**

【Experimental Result】

Data
- CIFAR-10 dataset (32x32 color images)

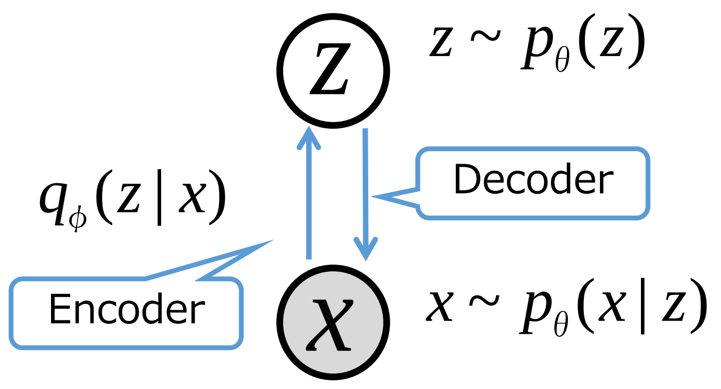
Performance (Unconditional)

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)
Gated PixelCNN:	3.03 (2.90)

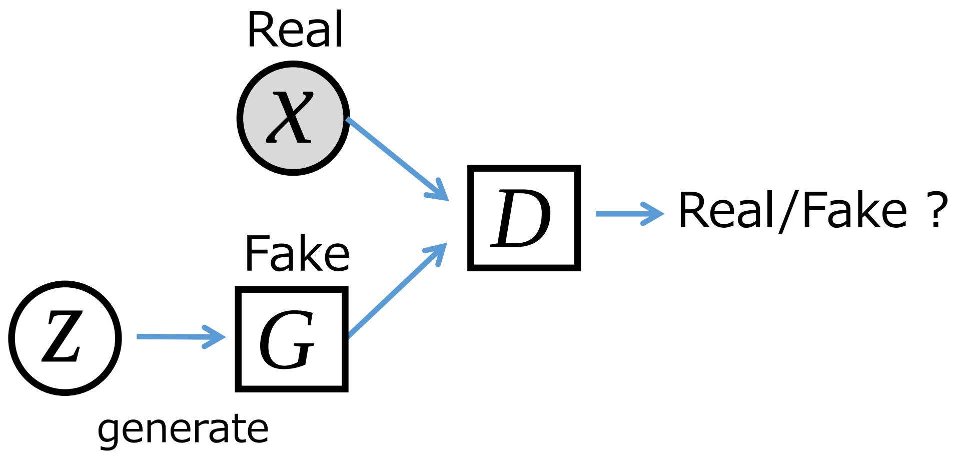
Image Generation Models

-Three image generation approaches are dominating the field:

Variational AutoEncoders (VAE)

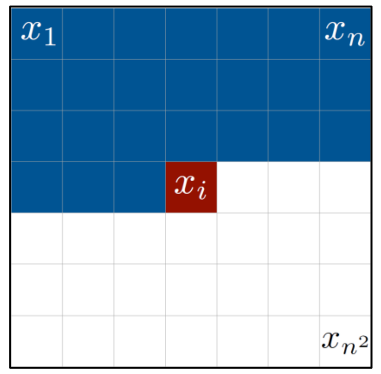


Generative Adversarial Networks (GAN)



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

Autoregressive Models



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

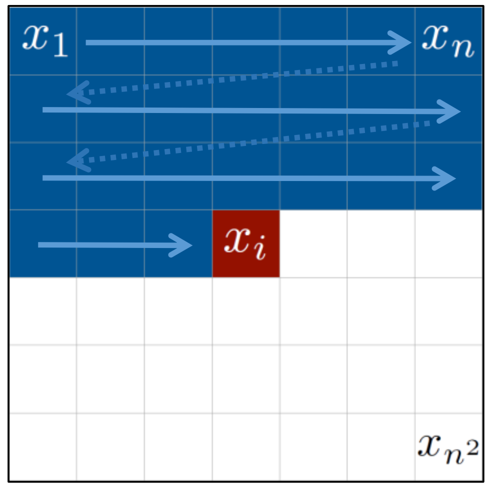
	VAE	GAN	Autoregressive Models
Pros.	- Efficient inference with approximate latent variables.	- generate sharp image. - no need for any Markov chain or approx networks during sampling.	- very simple and stable training process - currently gives the best log likelihood. - tractable likelihood
Cons.	- generated samples tend to be blurry.	- difficult to optimize due to unstable training dynamics.	- relatively inefficient during sampling

(cf. https://openai.com/blog/generative-models/)

Autoregressive Image Modeling

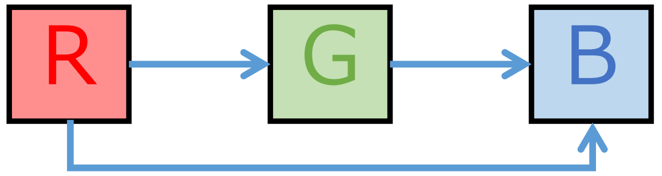
- Autoregressive models train a network that models the conditional distribution of every individual pixel given previous pixels (raster scan order dependencies).

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



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

- 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.

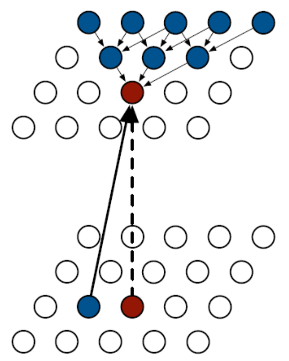


Previous work: Pixel Recurrent Neural Networks.

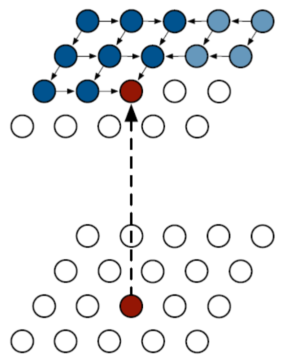
- "Pixel Recurrent Neural Networks" got best paper award at ICML2016.
- They proposed two types of models, **PixelRNN** and **PixelCNN**
  (two types of LSTM layers are proposed for PixelRNN.)

PixelRNN

Row LSTM

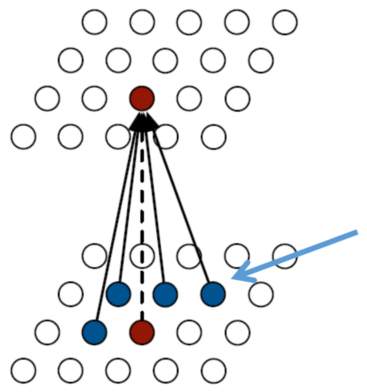


Diagonal BiLSTM



- LSTM based models are natural choice for dealing with the autoregressive dependencies.

PixelCNN



masked convolution

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \circ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- CNN based model uses masked convolution, to ensure the model is causal.

	PixelRNN	PixelCNN
Pros.	• effectively handles long-range dependencies ⇒ **good performance**	Convolutions are easier to parallelize ⇒ **much faster** to train
Cons.	• Each state needs to be computed sequentially. ⇒ **computationally expensive**	Bounded receptive field ⇒ **inferior performance** **Blind spot problem** (due to the masked convolution) needs to be eliminated.

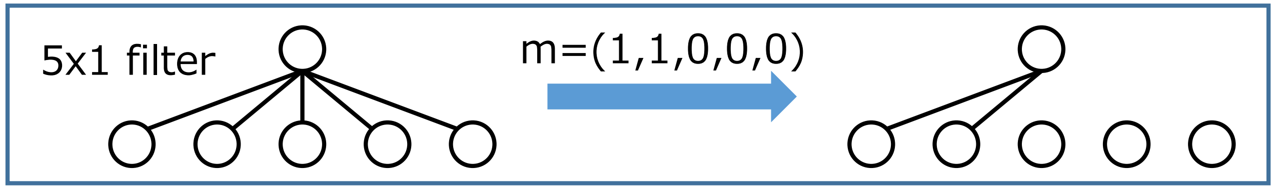
Details of “Masked Convolution” & “Blind Spot”

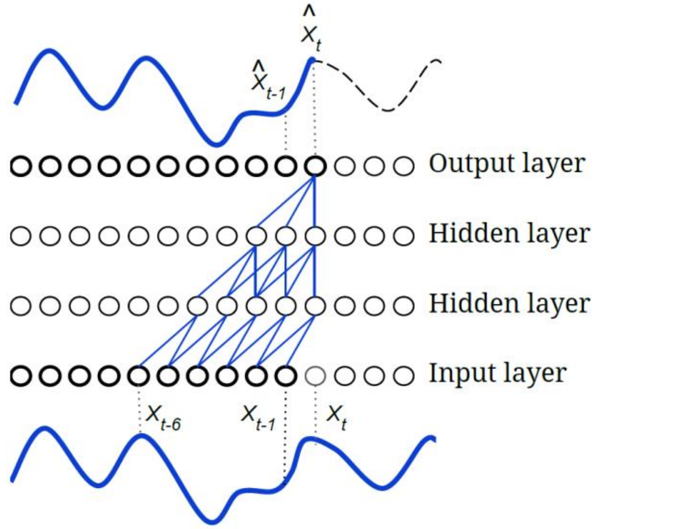
- To generate next pixel, the model can only condition on the previously generated pixels.
- Then, to make sure CNN can only use information about pixels above and to the left of current pixel, the filters of the convolution need to be masked.

(cf. Generating Interpretable Images with Contollable Structure, S.Reed et.al., 2016)

Case 1D

(ex. text, audio, etc)





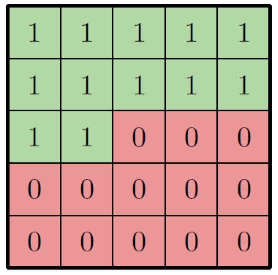
- Right figure shows 5x1 convolutional filters after multiplying them by mask.
- The filters connecting the input layer to the first hidden layer are in this case multiplied by m=(1,1,0,0,0), to ensure the model is causal.

Case 2D

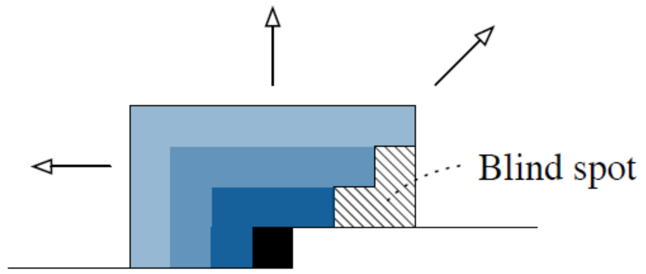
(ex. image)

- 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



3x3 filter, 3 layered

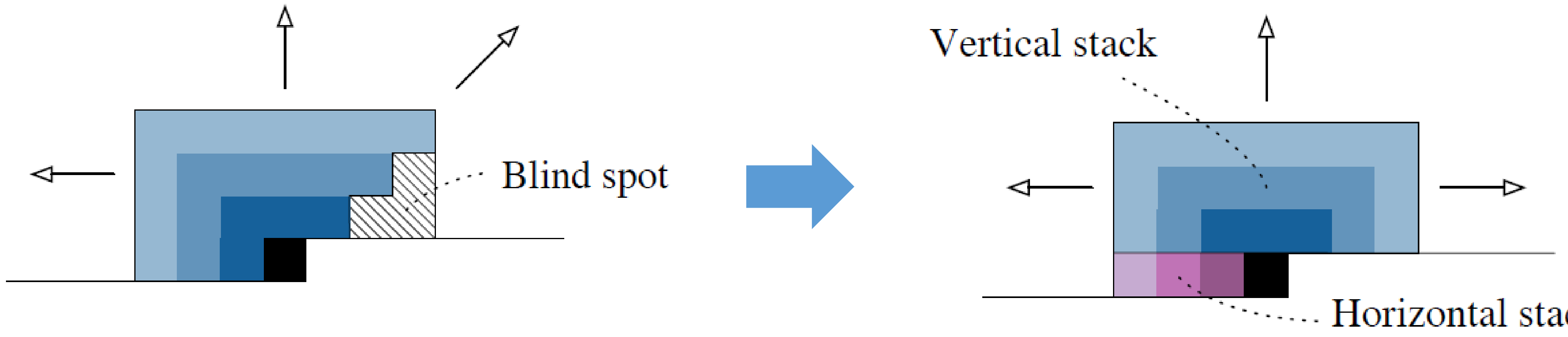
Proposed approach: Gated PixelCNN

- In this paper, they proposed the improved version of PixelCNN.
- Major improvements are as follows.

1. Removal of blind spots in the receptive field by combining **the horizontal stack** and **the vertical stack**.
2. Replacement of the ReLu activations between the masked convolutions in the original PixelCNN with the **gated activation unit**.
3. Given a latent vector, they modeled the conditional distribution of images, **conditional PixelCNN**.
   a. conditioning on class-label
   b. conditioning on embedding from trained model
4. 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.

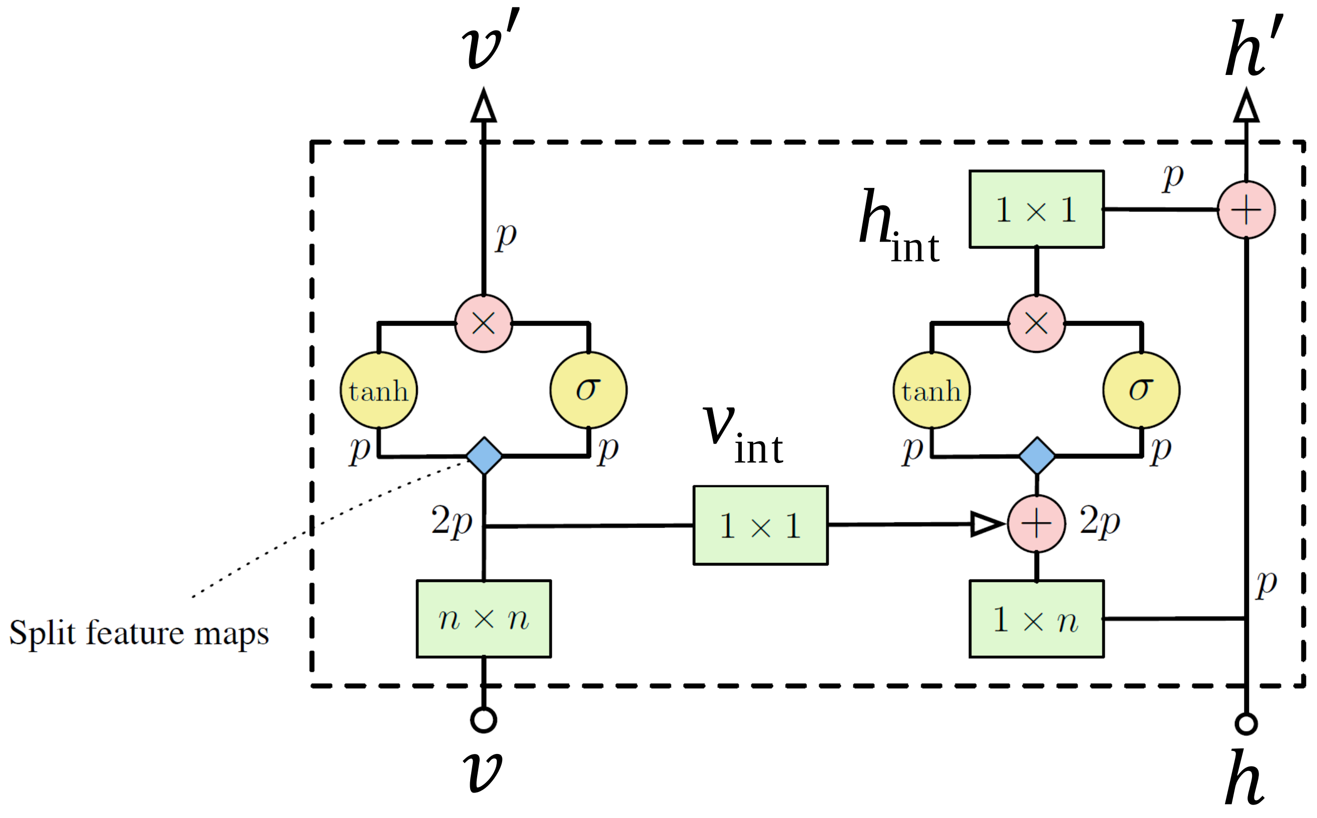
  - Details about implementation techniques are described below.

Second improvement: Gated activation and architecture

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

(σ: sigmoid, k: number of layer, ⊙: element-wise product, *: convolutional operator)

■ Single layer block of a Gated PixelCNN



- 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

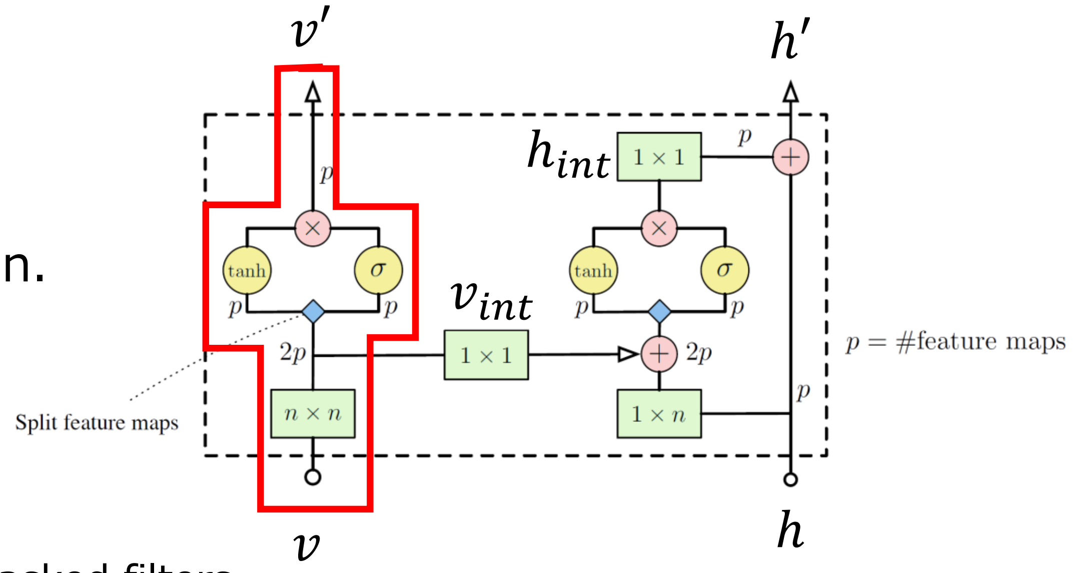
- Break down operations into four steps.

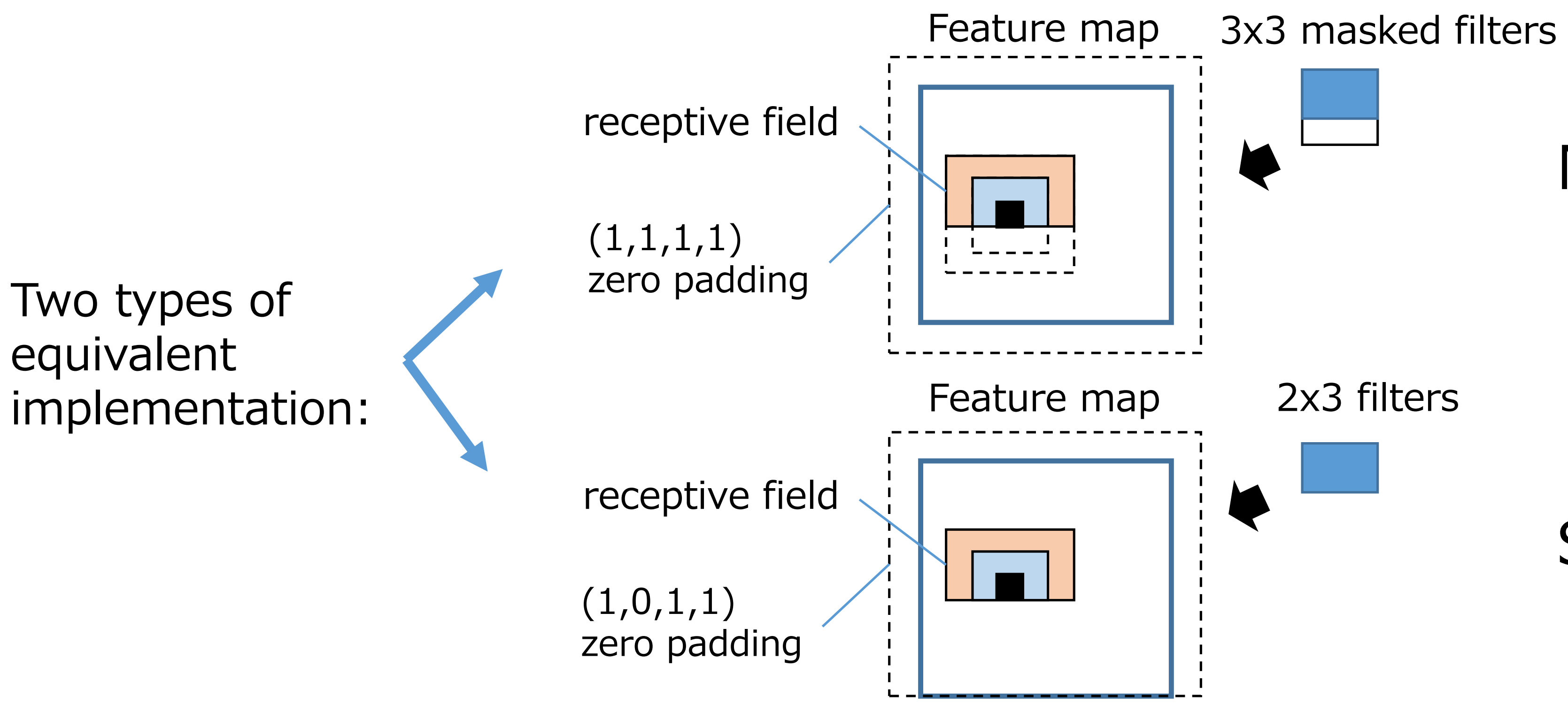
① Calculate vertical feature maps

… n×n convolutions are calculated with gated activation.

Input: v (= input image if 1st layer)

Output: v' (ex. n=3)





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.

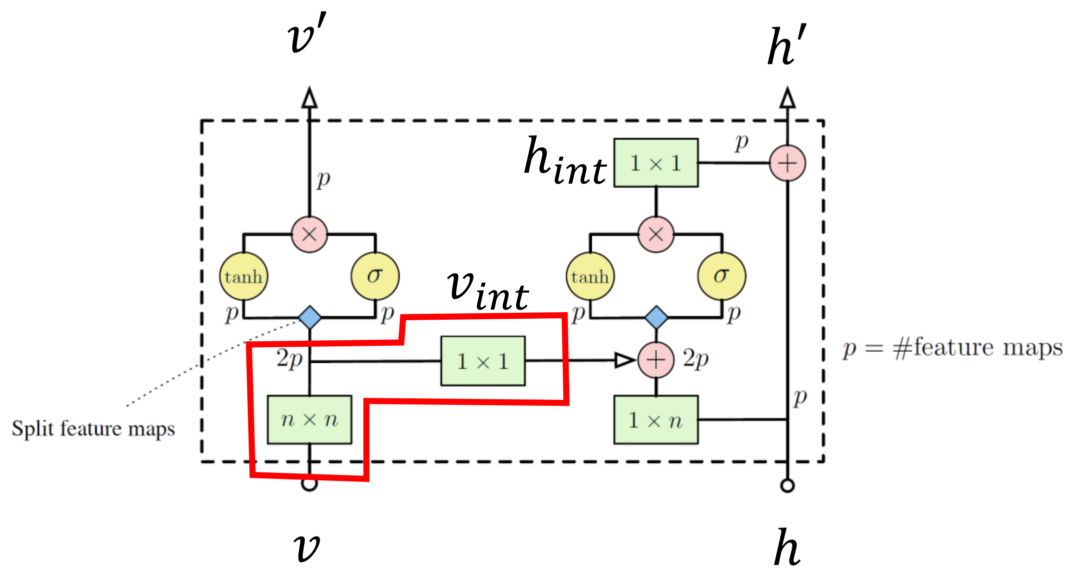
Details of Gated PixelCNN architecture

② Feed vertical maps into horizontal stack

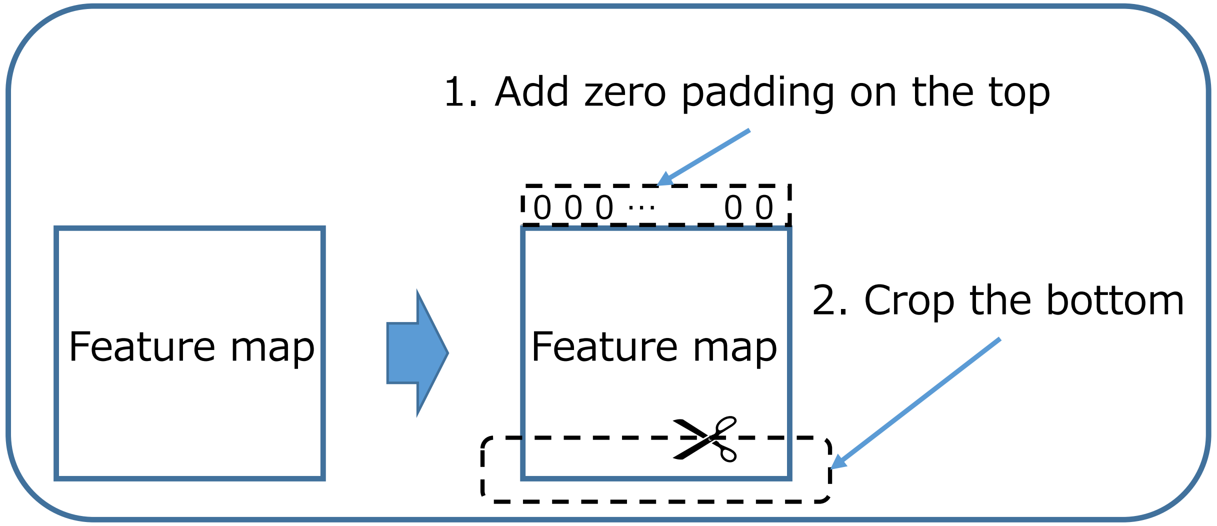
1. n x n masked convolution
2. shifting down operation (as below)
3. 1 x 1 convolution

Input: v (= input image if 1st layer)

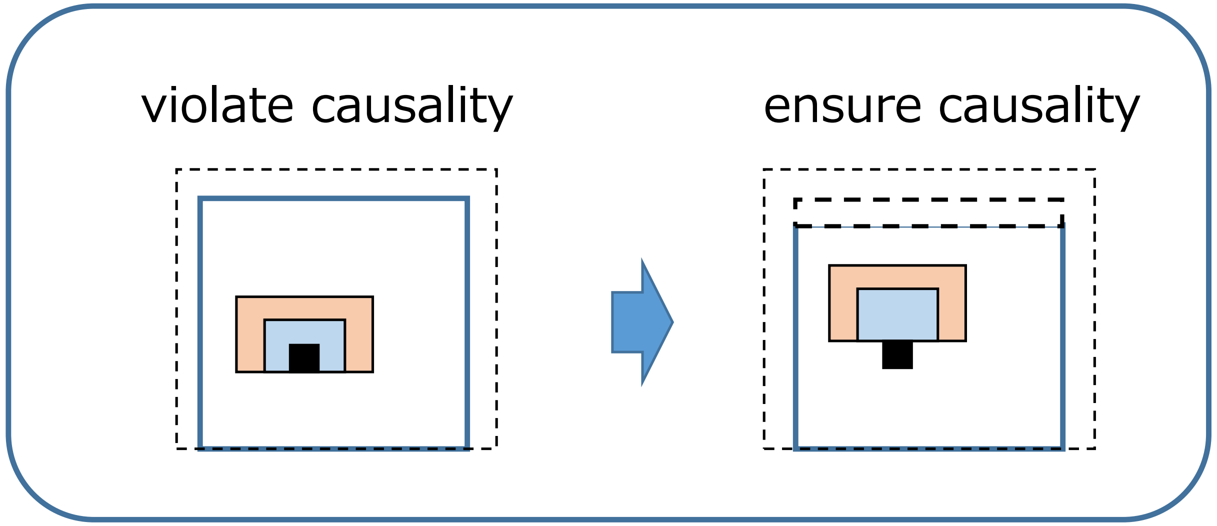
Output: v_{int}



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



Left operations can be interpreted as below.



Details of Gated PixelCNN architecture

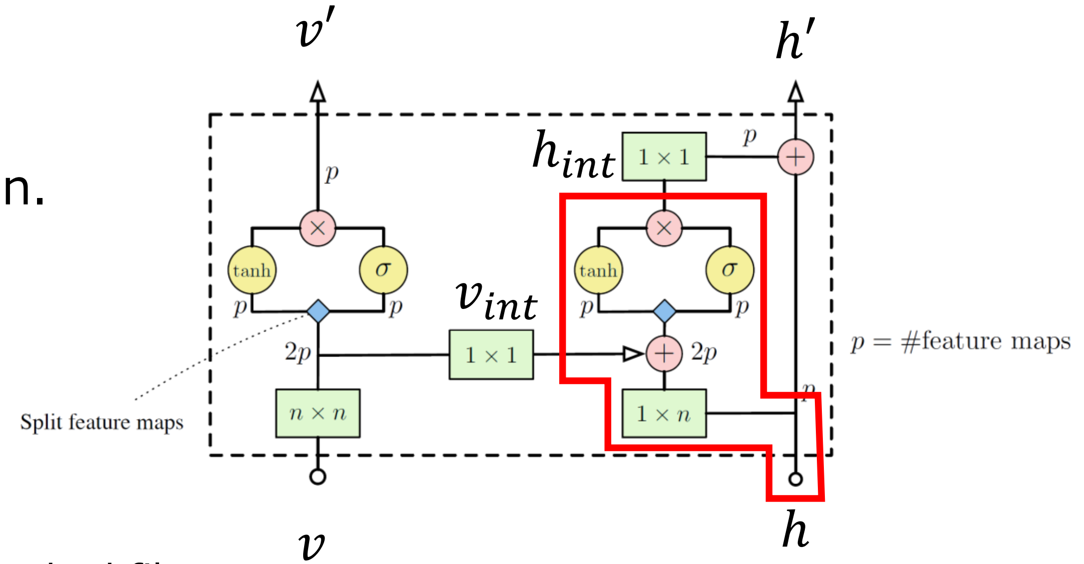
③ Calculate horizontal feature maps

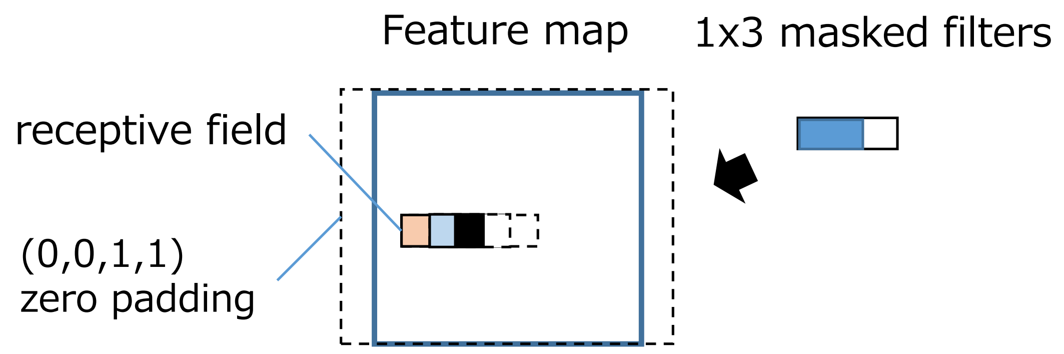
⋯ 1×n convolutions are calculated with gated activation.
(vertical maps are added before activation.)

Input: v_{int} , h (input image if 1st layer)

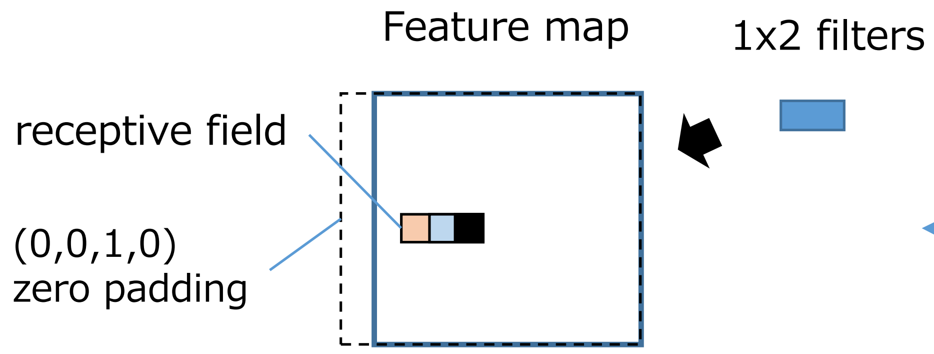
Output: h_{int}

(ex. n=3)





Two types of equivalent implementation:



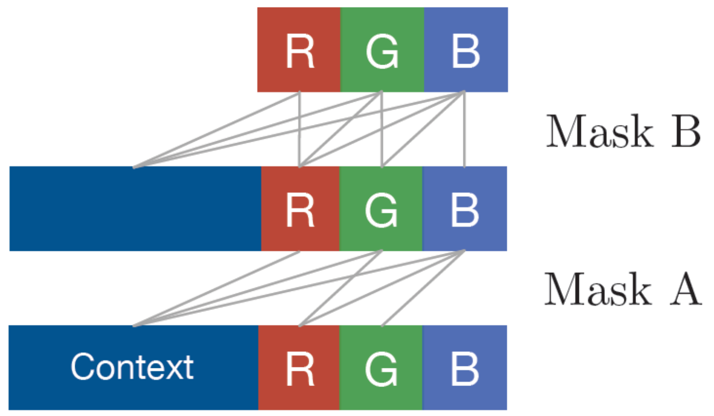
Next problem:

➢ Mask 'A' vs Mask 'B'

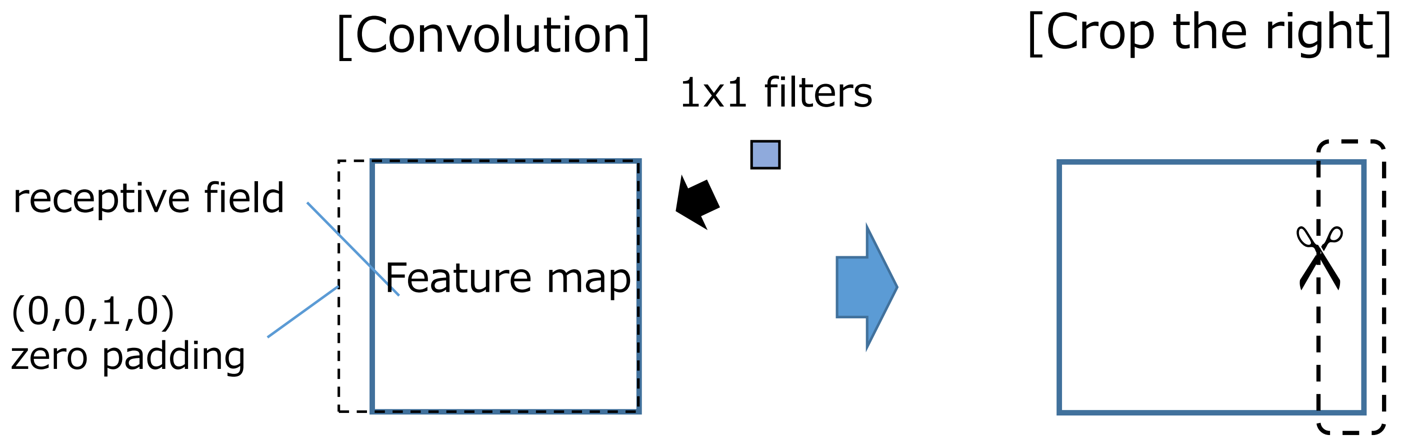
- Mask 'A' (restrict connection from itself) is applied to only to the first convolution.
- Mask 'B' (allow connection from itself) is applied to all the subsequent convolution.

Details of Gated PixelCNN architecture

③ Calculate horizontal feature maps



- Mask 'A' can be implemented as below. (ex. n=3)

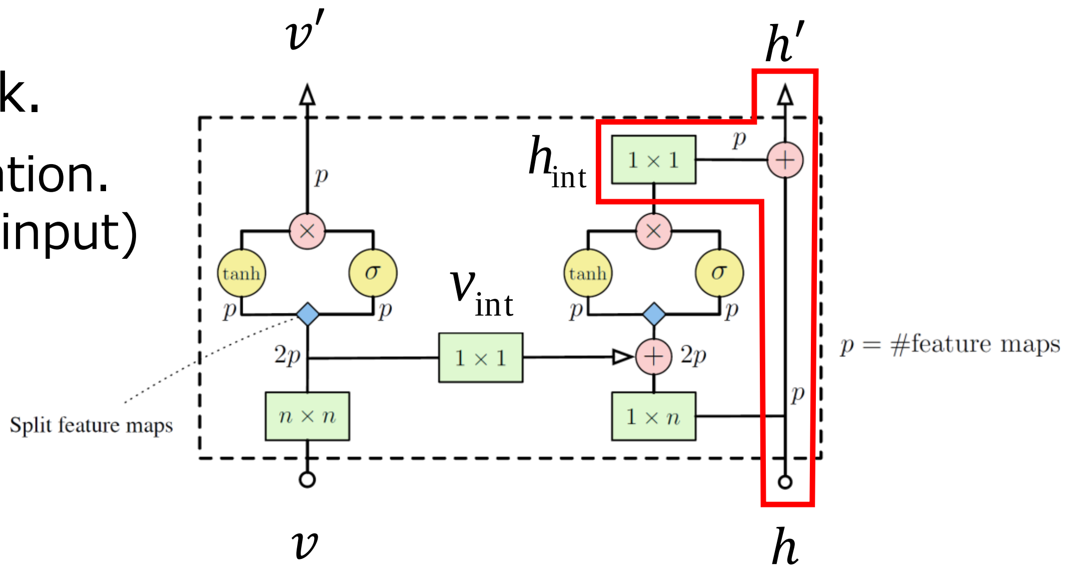


④ Calculate residual connection in horizontal stack.

⋯ 1×1 convolutions are calculated without gated activation. then, maps are added to horizontal maps (of layer's input)

Input: h_{int} , h (input image if 1st layer)

Output: h'

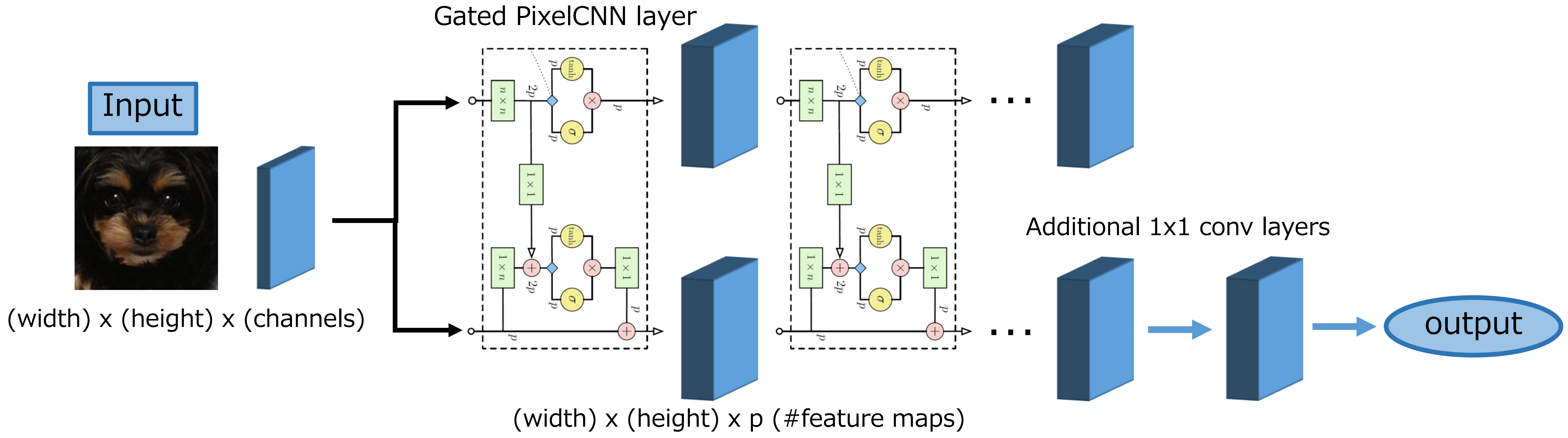


Output layer and whole architecture

■ 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


Gated PixelCNN layer
Input
(width) x (height) x (channels)
$n \times n$
$2p$
p
tanh
σ
1×1
$1 \times n$
Additional 1x1 conv layers
output
(width) x (height) x p (#feature maps)

Third improvements: conditional PixelCNN & PixelCNN AutoEncoder

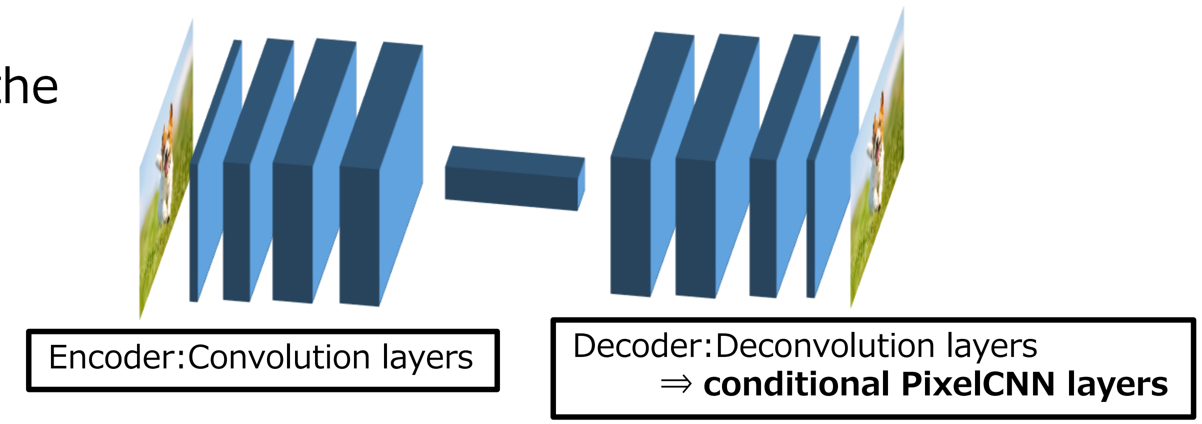
■ coniditional PixelCNN

	original	**conditional**
Model	$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i \mid x_1, ..., x_{i-1}).$	$p(\mathbf{x} \mid \mathbf{h}) = \prod_{i=1}^{n^2} p(x_i \mid x_1, ..., x_{i-1}\underline{, \mathbf{h}})$
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 h to the activations before the nolinearities

■ PixelCNN AutoEncoder

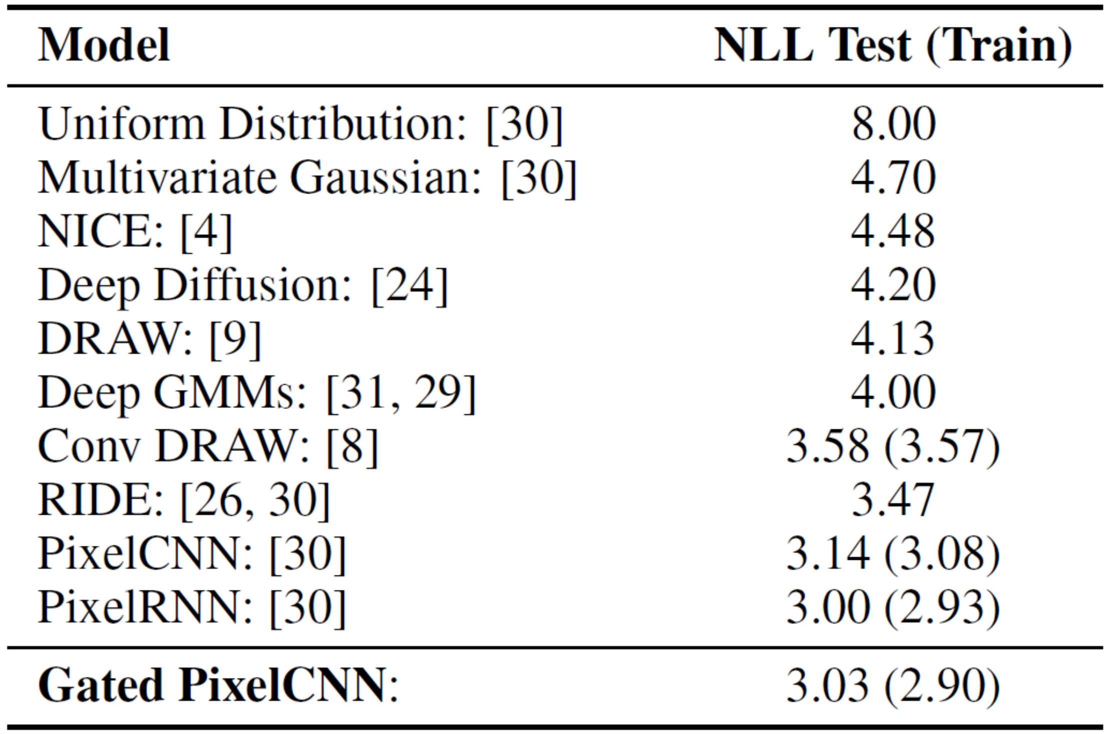
- From a convolutional auto-encoder, they replaced the deconvolutional decoder with conditional PixelCNN


Encoder:Convolution layers
Decoder:Deconvolution layers
⇒ **conditional PixelCNN layers**

Experimental Results (Unconditional)

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

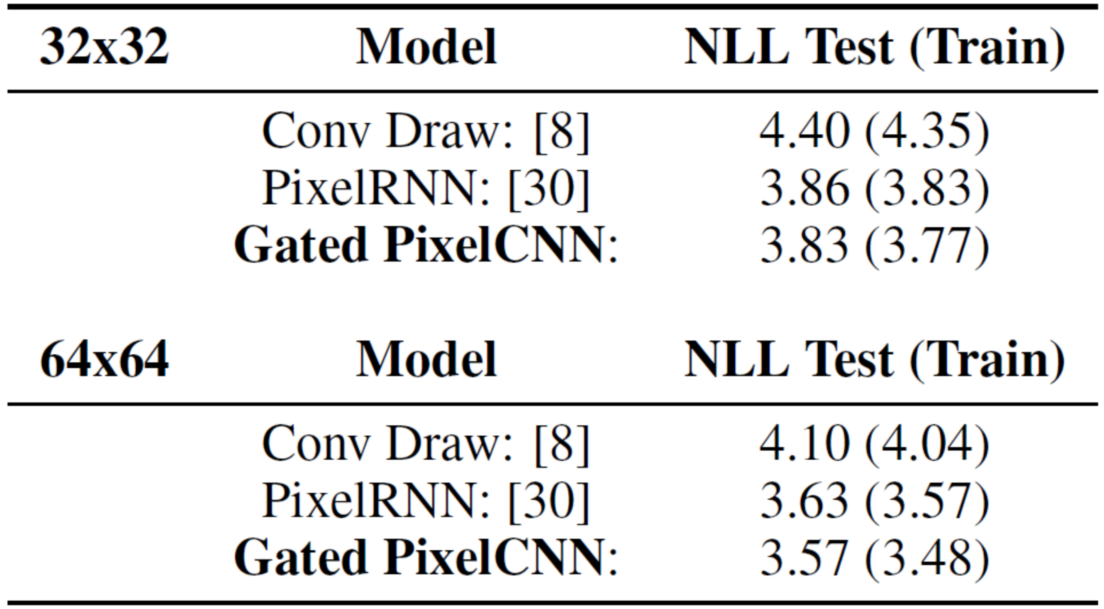
- Data: CIFAR-10 dataset



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)
Gated PixelCNN:	3.03 (2.90)

- 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



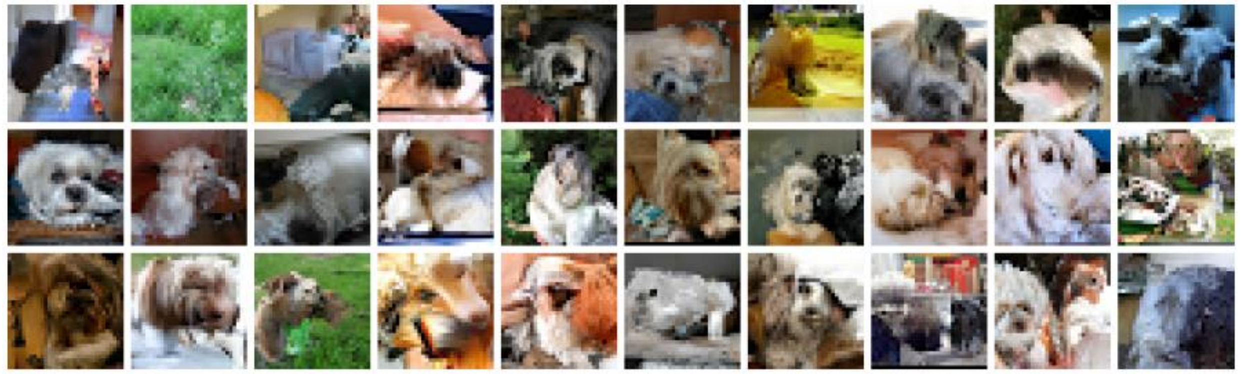
32x32	**Model**	**NLL Test (Train)**
	Conv Draw: [8]	4.40 (4.35)
	PixelRNN: [30]	3.86 (3.83)
	Gated PixelCNN:	3.83 (3.77)
64x64	**Model**	**NLL Test (Train)**
	Conv Draw: [8]	4.10 (4.04)
	PixelRNN: [30]	3.63 (3.57)
	Gated PixelCNN:	3.57 (3.48)

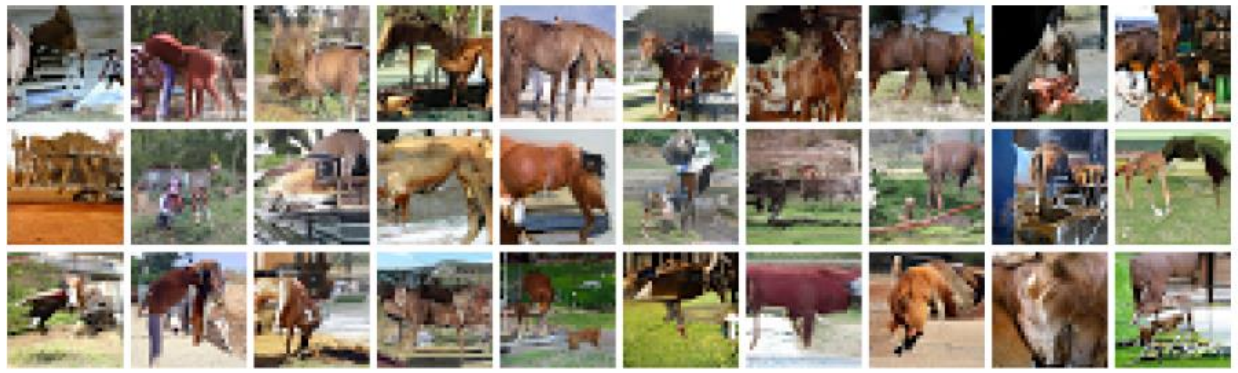
- Gated PixelCNN outperforms PixelRNN.
- Achieve similar performance to the PixelRNN in less than half the training time.

Experimental Results (Conditional)

■ Coniditioning on ImageNet classes

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

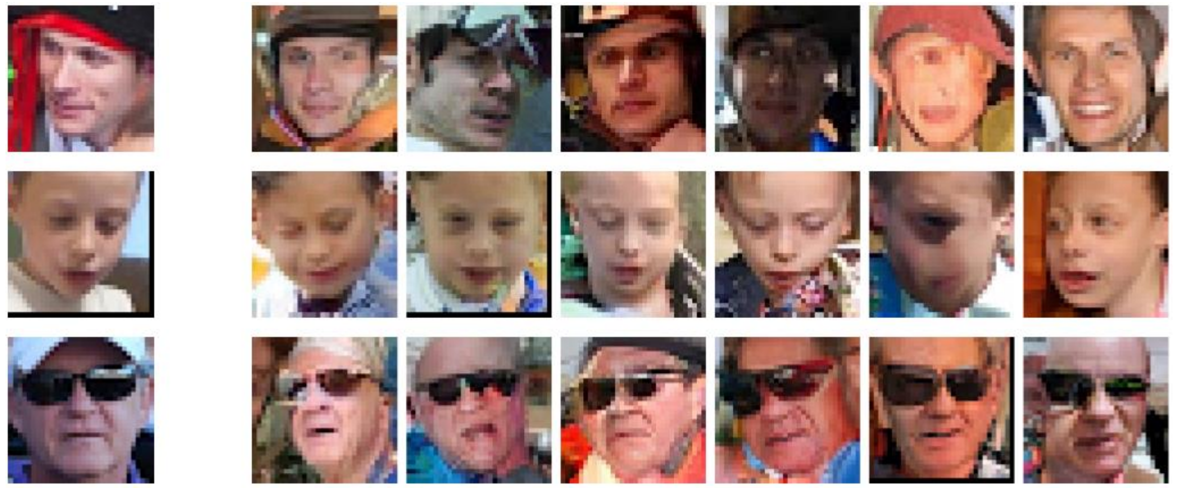

Lhasa Apso (dog)


Sorrel horse

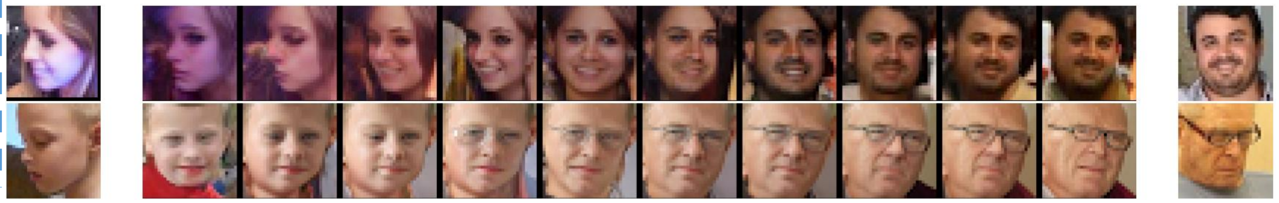
(part of results.)

■ Coniditioning on Portrait Embeddings

- Embeddings are took from top layer of a conv network trained on a large database of portraits from Flickr images.
- After the supervised net was trained, {x:image, h: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.



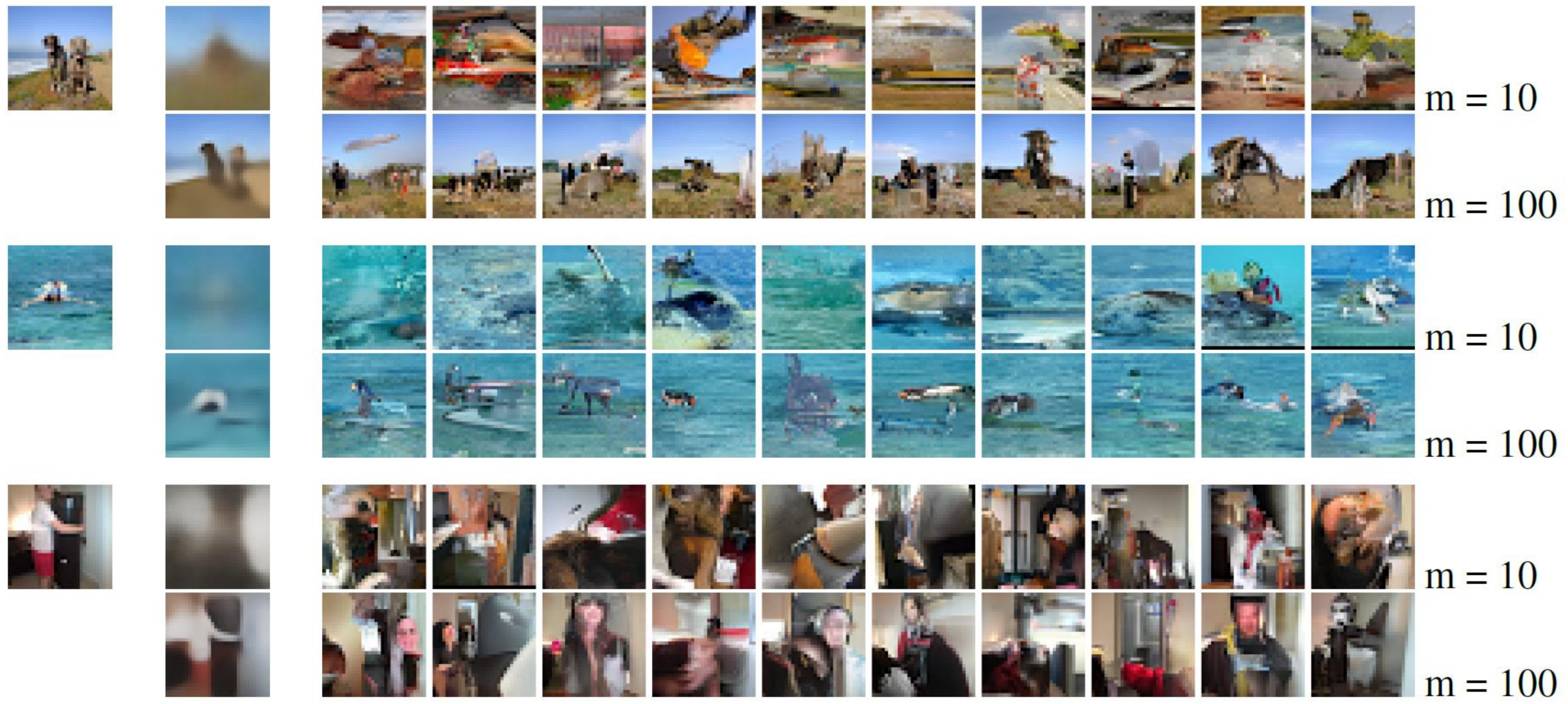
- And experimented with reconstructions conditioned on linear interpolations between embeddings of pairs of images.



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)

Summary & Reference

◆ Summary

■ Improved PixelCNN:

- Same performance as PixelRNN, but faster (easier to parallelize)
- Fixed "*blind spots*" problem
- Gated activation units

■ Conditional Generation:

- Conditioned on class-label
- Conditioned on portrait embedding
- PixelCNN AutoEncoders

◆ References

[1] Aäron van den Oord et al., "Conditional Image Generation with PixelCNN Decoders", NIPS 2016

[2] Aäron van den Oord et al., "Pixel Recurrent Neural Networks", ICML 2016 (Best Paper Award)

[3] S. Reed, A. van den Oord et al., "Generating Interpretable Images with Controllable Structure", Under review as a conference paper at ICLR 2017

Appendix – Progress of research related to this paper

■ Applied to other domains

- **"WaveNet: A Generative Model for Raw Audio"**, A. van den Oord et al. (DeepMind)
  - The conditional probability distribution is modelled by a stack of dilated causal convolutional layers with gated activation units.

- **"Video Pixel Networks"**, Nal Kalchbrenner, A. van den Oord et al. (DeepMind)
  - The architecture of the generative video model consists of two parts:
    1. Resolution preserving CNN encoders
    2. PixelCNN decoders

- **"Generating Interpretable Images with Controllable Structure"**, S.Reed, A. van den Oord et al. (Google DeepMind), Under review as a conference paper at ICLR 2017
  - Text-to-image synthesis (generating images from captions and other strcuture) using gated conditional PixelCNN model.

- **"Language Modeling with Gated Convolutional Networks"**, Yann N.Dauphin et al. (Facebook AI Research)
  - A new language model that replace recurrent connections typically used in RNN with gated temporal convolutions.

Appendix – Progress of research related to this paper

■ Modifications of Gated PixelCNN model

➢ **"PixelCNN++: A PixelCNN Inplementation with Discretized Logistic Mixture Likelihood and Other Modifications"**, Tim Salimans, Andrej Karpathy, et al. (OpenAI), Under review as a conference paper at ICLR 2017

- A number of modifications to the original gated PixelCNN model.
  1. Use a discretized logistic mixture likelihood, rather than a 256-way sofmax.
  2. Condition on whole pixels, rather than R/G/B sub-pixels.
  3. Use downsampling.
  4. Introduce additional short-cut connections. (like as U-net)
  5. Regularize the model using dropout.

➢ **"PixelVAE: A Latent Variable Model for Natural Images"**, Ishaan Gulrajani, Kundan Kumar et al., Under review as a conference paper at ICLR 2017

- A VAE model with an autoregressive decoder based on PixelCNN.

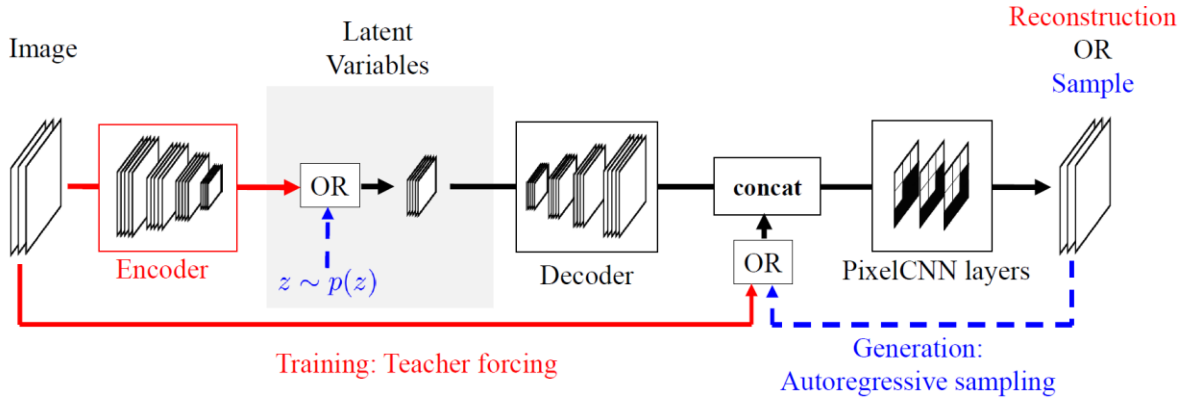


Figure 2: Our proposed model, PixelVAE, makes use of PixelCNN to model an autoregressive decoder for a VAE. VAEs, which assume (conditional) independence among pixels, are known to suffer from blurry samples, while PixelCNN, modeling the joint distribution, produces sharp samples, but lack a latent representation that might be more useful for downstream tasks. PixelVAE combines the best of both worlds, providing a meaningful latent representation, while producing sharp samples.