Autoregressive Models

Presented By Mingyu Yang and Bahaa Aldeeb

Autoregressive Model - Background

Textbook definition: Autoregressive models defines a random process where the current state is linearly dependent on prior state and on a stochastic term.

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

We will think of them as: Models that generate data based on prior data

What Problems Could Autoregressive Models Solve?

What are some uses of autoregressive models?

- Statistics, econometrics and financial modelling (1950s)
- Language Modeling: XLNet
- Text-to-speech: WavNet
- Language translation
- Conditional Image Generation - encoding image distributions

Conditional Image Generation with PixelCNN Decoders

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

From Google DeepMind

NIPS 2016

Presented by Bahaa Aldeeb

Modeling Image Distributions

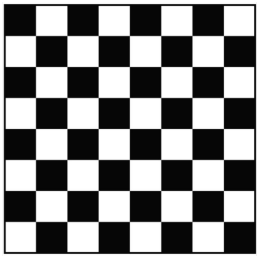
Motivation - Some uses of learning image distributions:

- Filtering: removing occlusions, denoising.
- Prediction: future frames, image from word
- Generation: generate new content
- Others

Formulating the problem

We are trying to learn or model this **x** in a network

$\mathbf{x}$

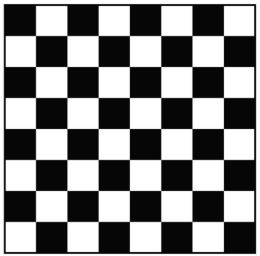


$x_1, ..., x_{n^2}$

Formulating the problem

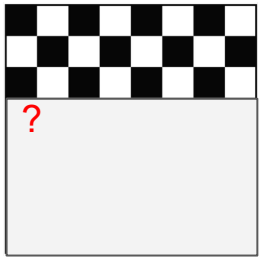
We are trying to learn or model this **x** in a network

$p(\mathbf{x})$



$x_1, ..., x_{n^2}$

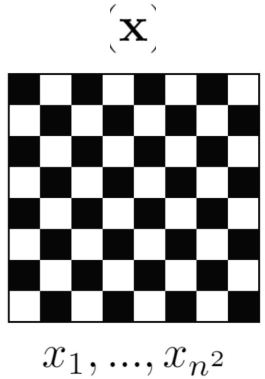
$p(x_i|x_1, ..., x_{i-1})$

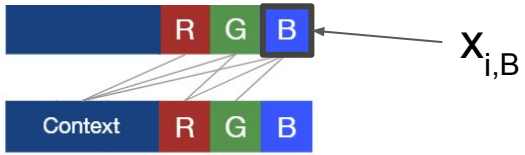


Formulating the problem - probabilistically

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

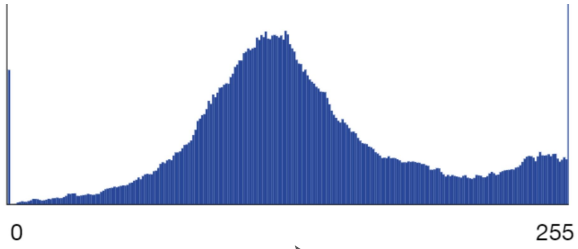
$$= \prod_{i=1}^{n^2} p(x_{i,R} | \mathbf{x}_{<i}) p(x_{i,G} | \mathbf{x}_{<i}, x_{i,R}) p(x_{i,B} | \mathbf{x}_{<i}, x_{i,R}, x_{i,G})$$





Autoregressive Image Generation

$$p(x_{i,R}|\mathbf{x}_{<i})$$

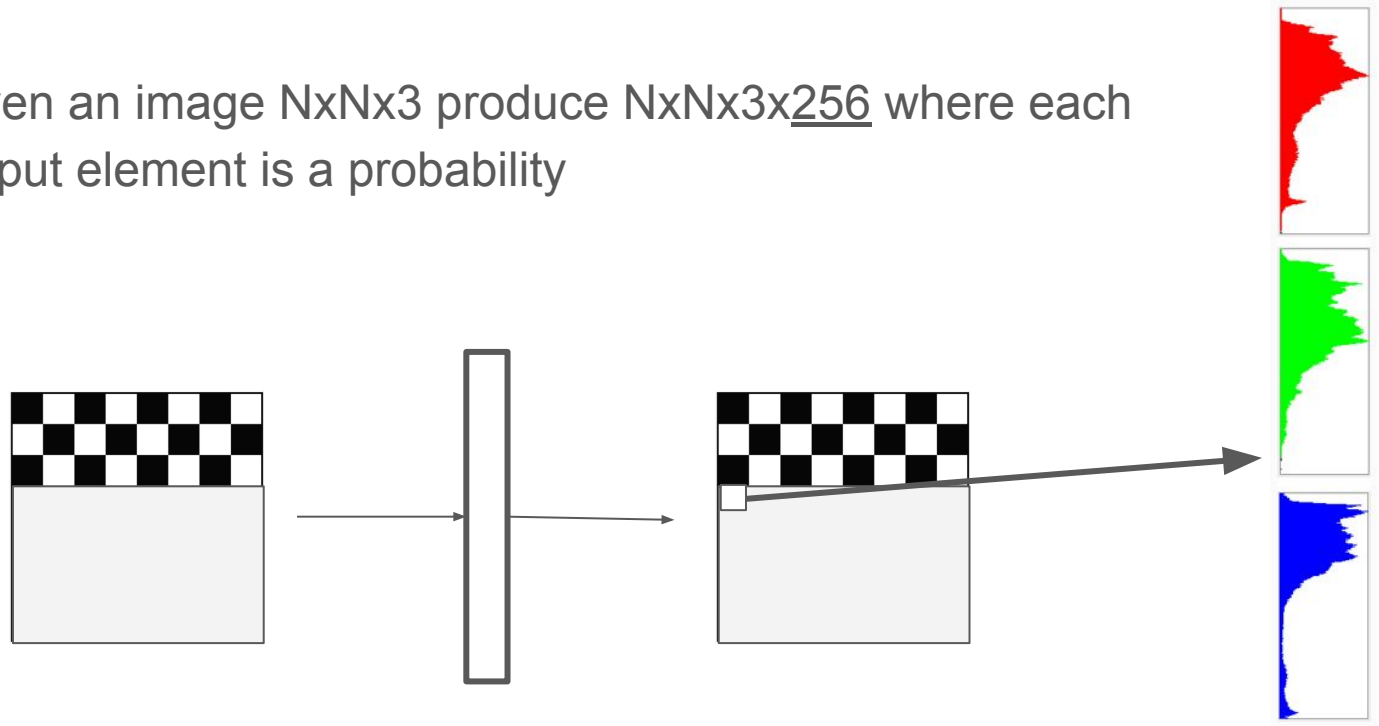


Learning this is in a sense learning the image distribution

Discrete probability distribution
Softmax output

Formulating the problem

Given an image NxNx3 produce NxNx3x256 where each output element is a probability

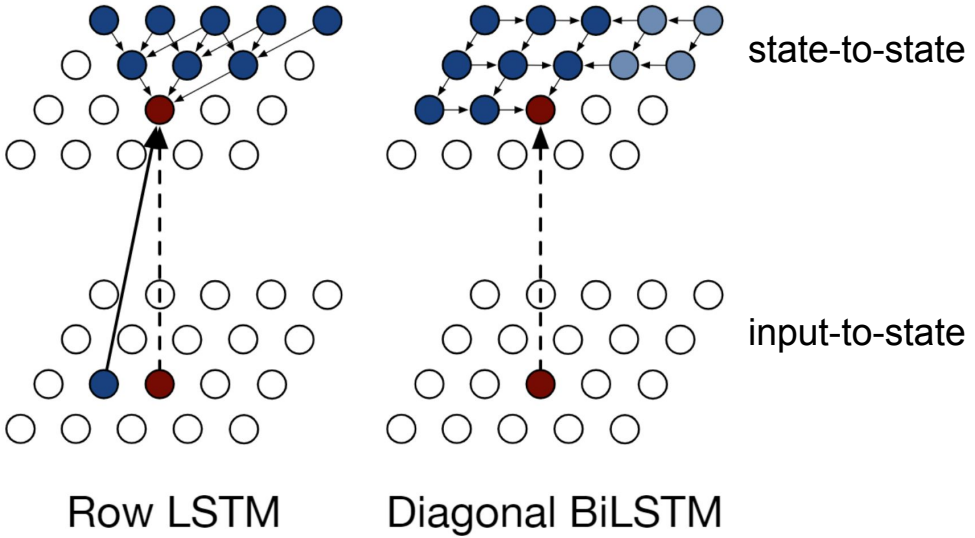


PixelRNN

Connected LSTMs which take in one pixel at a time.

Pros: Large receptive field, sufficiently complex model

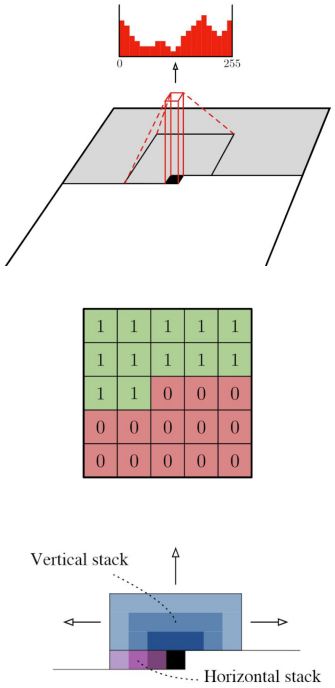
Issues: Very slow to train (hard to parallelize)



PixelCNN

Pros: Faster, Easier to train

Cons: Smaller receptive field, not complex enough



PixelCNN architecture in an image

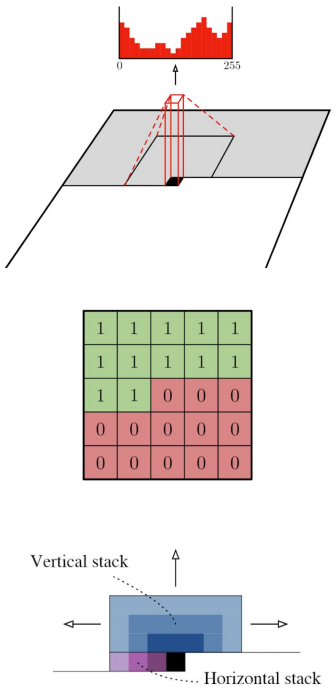
Gated PixelCNN

Pros: Faster, Data as input (instead of using memory)

Cons: Smaller receptive field, not complex enough

Solutions:

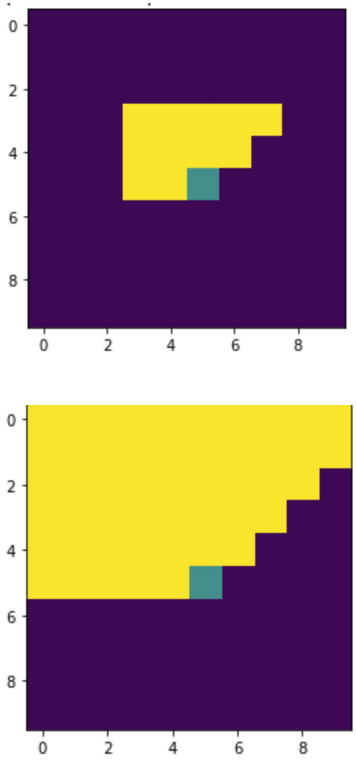
- More masked layers and updated mask shapes to increase receptive field
- Gated convolutions to increase model complexity and ability to learn



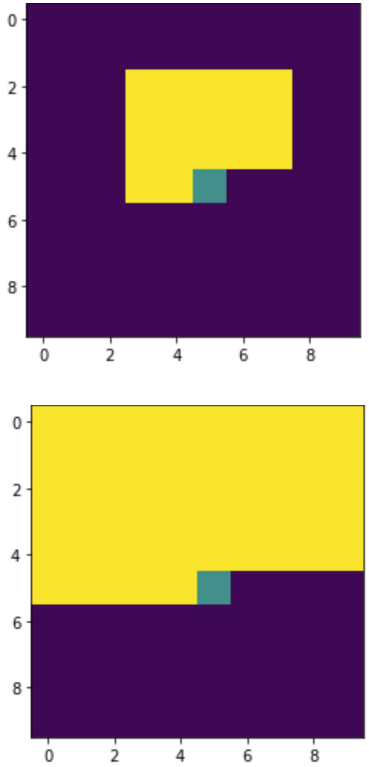
PixelCNN architecture in an image

Gated pixelCNN - Updated Mask Shapes

PixelCNN masks



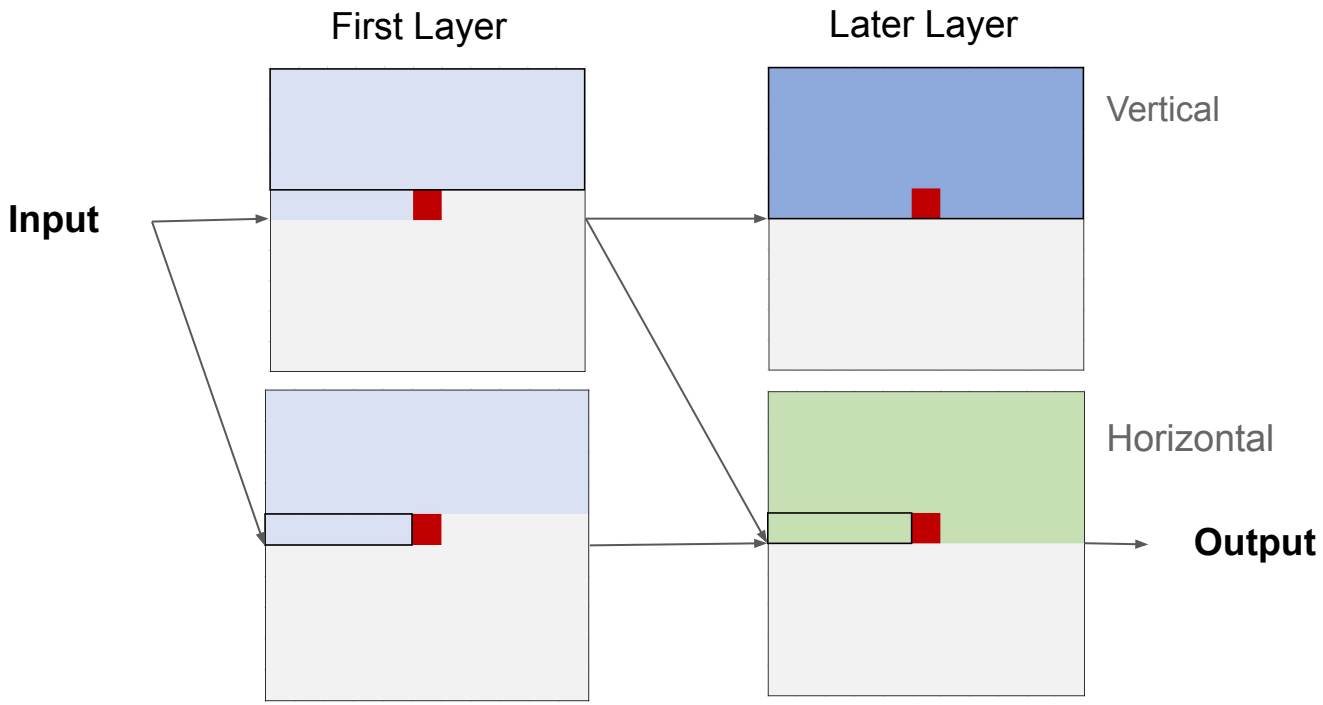
Gated PixelCNN masks



2nd Conv Layer

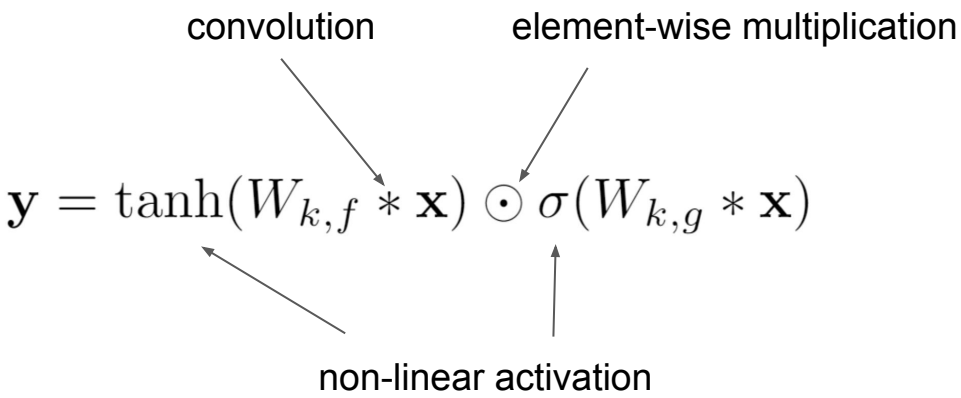
5nd Conv Layer

Gated PixelCNN - Convolution Masks

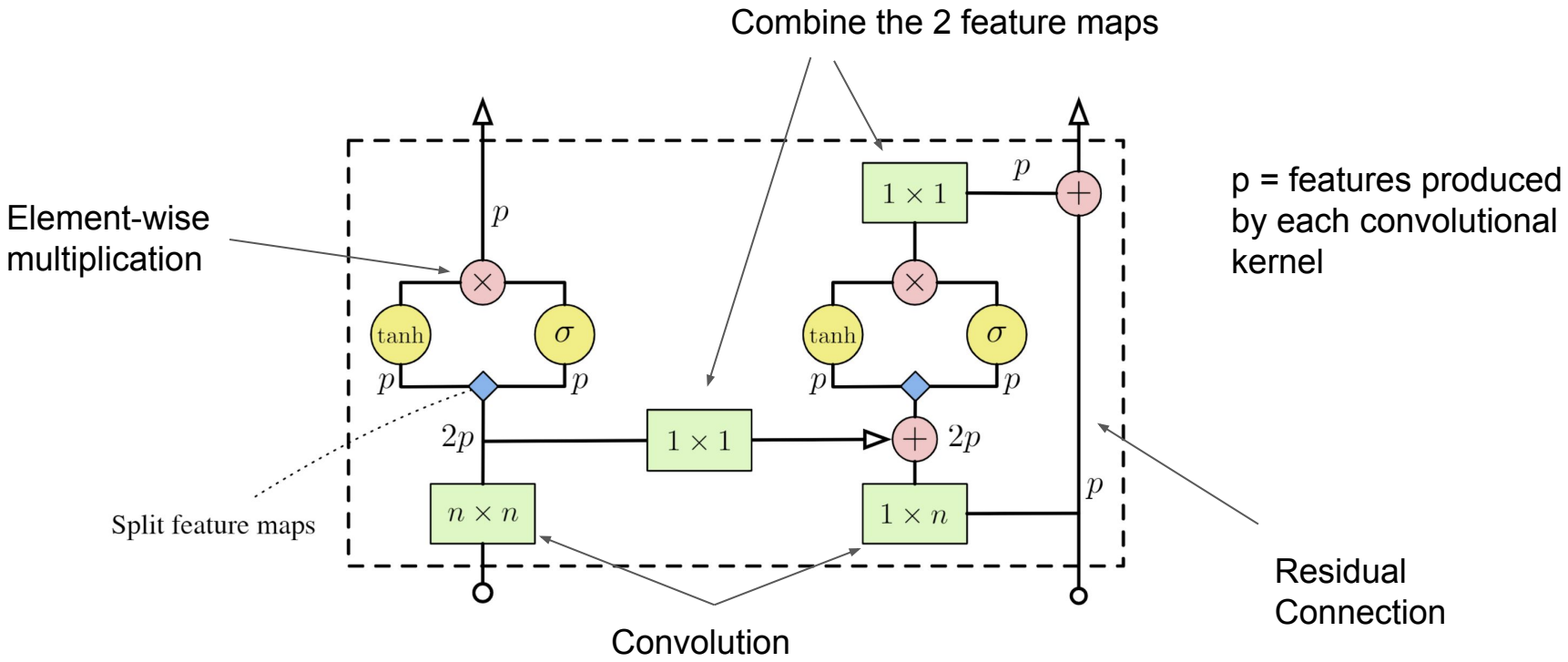


Gated Convolution

Convolutional layer



Gated PixelCNN - Gated Convolutions



Conditional Generation

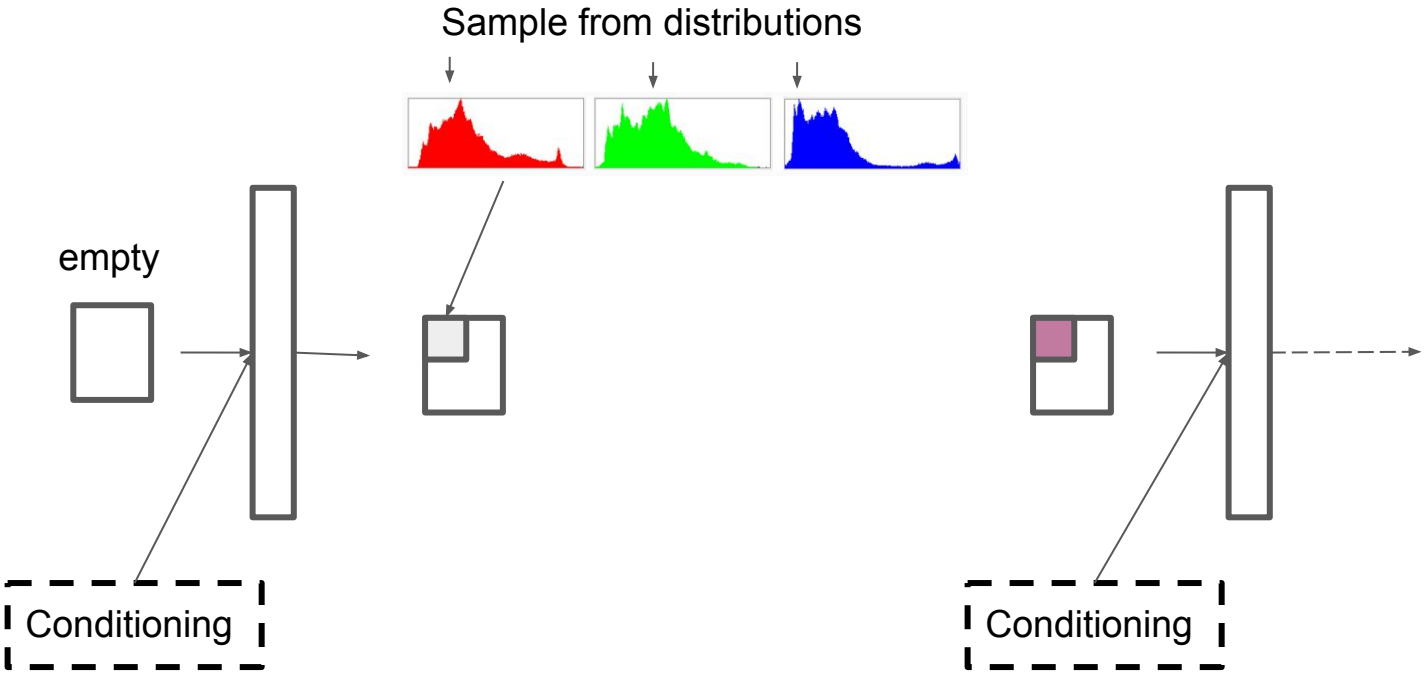
Re-formulated problem: $p(\mathbf{x}|\mathbf{h}) = \prod_{i=1}^{n^2} p(x_i|x_1, ..., x_{i-1}, \mathbf{h}).$

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

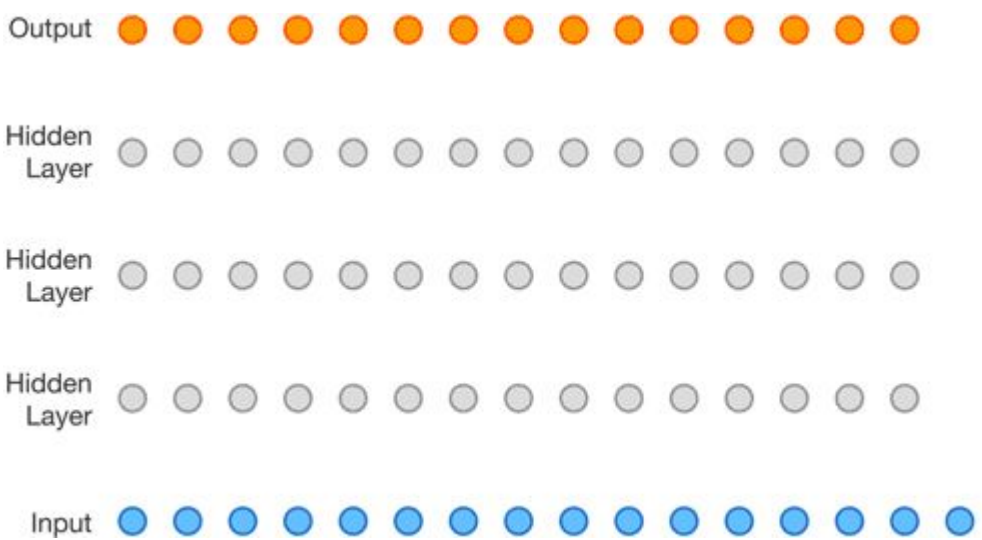
Possible “h” values:

- Constant one-hot encoded category
- Position dependent encoding
- Output of an encoder (VQ-VAE2)

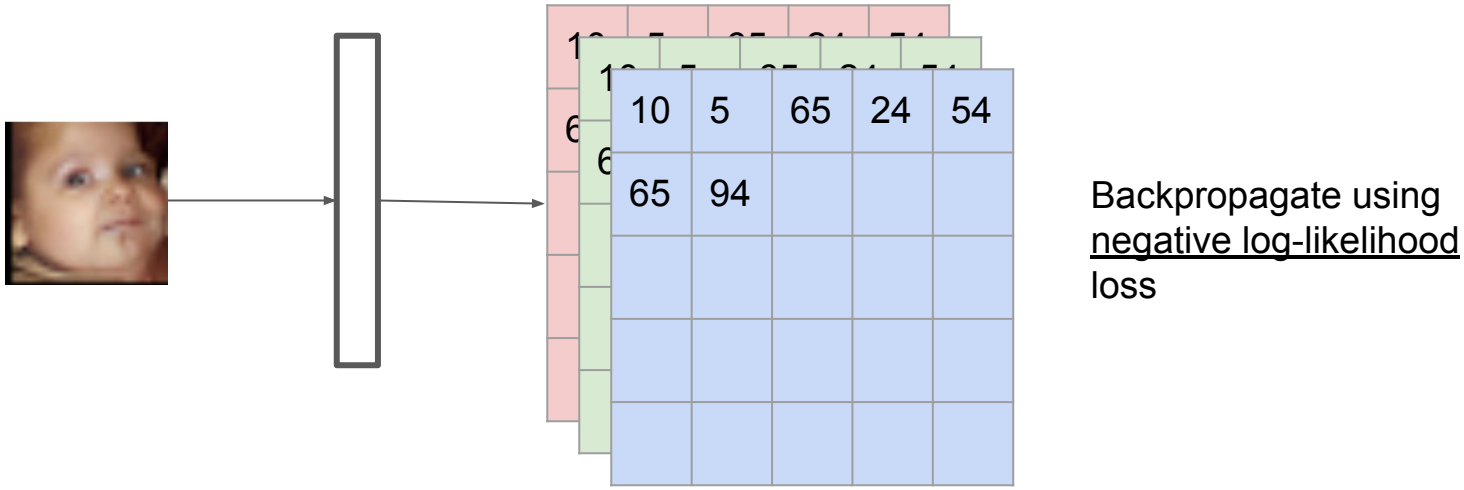
Sampling From PixelCNN



Data progression



Training


Select the probability of true intensity
10 5 65 24 54
65 94
Backpropagate using negative log-likelihood loss

Results

CIFAR10, 32x32	NLL (bpd)
Uniform Distribution	8.00
Multivariate Gaussian (van den Oord et al., 2016b)	4.70
Attention-based	
Image Transformer (Parmar et al., 2018)	2.90
PixelSNAIL (Chen et al., 2018)	2.85
Sparse Transformer (Child et al., 2019)	**2.80**
Convolutional	
PixelCNN (1 stream) (van den Oord et al., 2016b)	3.14
Gated PixelCNN (2 stream) (van den Oord et al., 2016a)	3.03
PixelCNN++ (1 stream)	2.99
PixelCNN++ (2 stream) (Salimans et al., 2017)	2.92
Ours, S-curve (1 stream, 1 order)	2.91
Ours, S-curve (1 stream, 8 orders)	**2.89**

Results from LMConv paper

Results

Running on ImageNet Data

- PixelCNN trains for half the time
- Used 20 layers of 5x5 filters
- Possible success of PixelCNN:
  - More data - reduces underfitting
  - More easily trained

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 - Conditional Generation

Constant one-hot encoded category

Gated PixelCNN - Conditional Generation

Sampling from embedding

Latent space interpolation

Pros and Cons

Pros:

- Plays well as a decoder (as shown by VQ-VAE2)
- Faster than it’s autoregressive predecessors

Cons:

- Uses relatively small images
- Still complex in the way it processes data
- Has to sequentially generate data (slower)

Discussion

- Will GANs replace Autoregressive models for image generation?

- Could attention mechanisms be used to improve Autoregressive model architectures?

- Questions? Interesting observations?