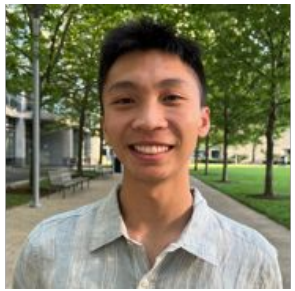
CS294-158 Deep Unsupervised Learning

Lecture 2 Likelihood Models: Autoregressive Models






Pieter Abbeel, Wilson Yan, Kevin Frans, Philipp Wu

Autoregressive Models – Lecture Outline

- Motivation
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- Deeper Dive into Causal Masked Neural Models
  - Convolutional
  - Attention
  - Tokenization
  - Caching
- Other things to be aware of
  - Decoder-only vs. Encoder-Decoder models
  - New incarnations of Recurrent Models
  - Alternative / Complementary ideas to tokenization

Autoregressive Models – Lecture Outline

- **Motivation**
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- Deeper Dive into Causal Masked Neural Models
  - Convolutional
  - Attention
  - Tokenization
  - Caching
- Other things to be aware of
  - Decoder-only vs. Encoder-Decoder models
  - New incarnations of Recurrent Models
  - Alternative / Complementary ideas to tokenization

Likelihood-based models

Problems we'd like to solve:

- Generate data: synthesize images, videos, speech, text
- Compress data: construct efficient codes
- Detect anomalies: i.e. data that is out of distribution

Likelihood-based models: estimate p_{data} from samples $x^{(1)}, \ldots, x^{(n)} \sim p_{data}(x)$

Learns a distribution p that allows:

- Computing p(x) for arbitrary x
- Sampling x ~ p(x)

Today: **discrete** data

Desiderata

We want to estimate distributions of **complex, high-dimensional data**

- A 128x128x3 image lies in a ~50,000-dimensional space

We also want computational and statistical efficiency

- Efficient training and model representation
- Expressiveness and generalization
- Sampling quality and speed
- Compression rate and speed

Autoregressive Models – Lecture Outline

- Motivation
- **1-Dimensional Distributions**
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- Deeper Dive into Causal Masked Neural Models
  - Convolutional
  - Attention
  - Tokenization
  - Caching
- Other things to be aware of
  - Decoder-only vs. Encoder-Decoder models
  - New incarnations of Recurrent Models
  - Alternative / Complementary ideas to tokenization

Learning: Estimate frequencies by counting

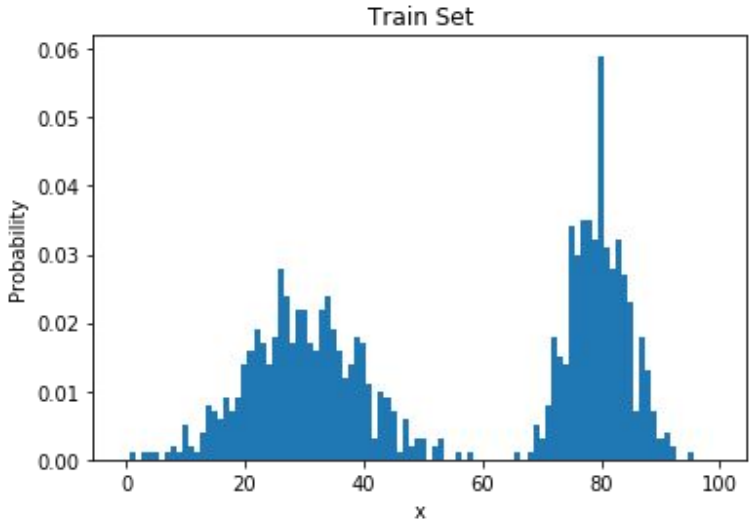
Recall: the goal is to estimate $\mathbf{p_{data}}$ **from samples** $\mathbf{x^{(1)}, \ldots, x^{(n)} \sim p_{data}(x)}$

Suppose the samples take on values in a finite set {1, …, k}

The model: a **histogram**

- (Redundantly) described by k nonnegative numbers: $p_1, \ldots, p_k$
- To train this model: count frequencies

  p_i = (# times i appears in the dataset)
  / (# points in the dataset)



Inference and Sampling

Inference (querying p_i for arbitrary i): simply a lookup into the array $p_1, \ldots, p_k$

Sampling (lookup into the inverse cumulative distribution function)

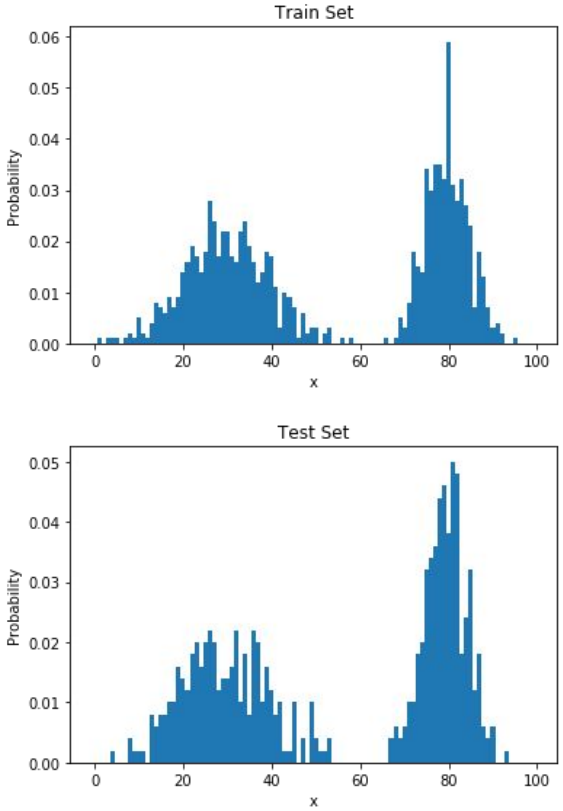
1. From the model probabilities $p_1, \ldots, p_k$, compute the cumulative distribution

$$F_i = p_1 + \cdots + p_i \quad \text{for all } i \in \{1, \ldots, k\}$$

2. Draw a uniform random number $u \sim [0, 1]$
3. Return the smallest i such that $u \leq F_i$

Are we done?

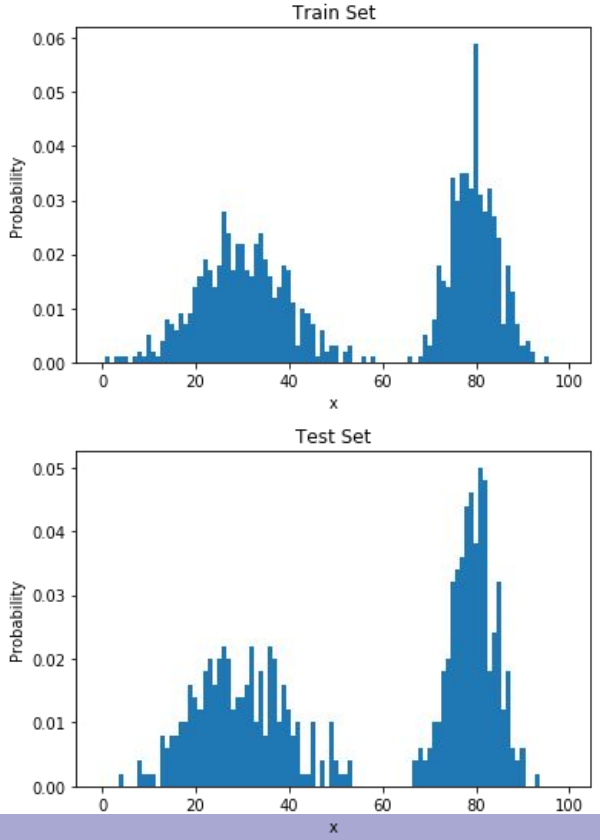
Problem: Lack of Generalization

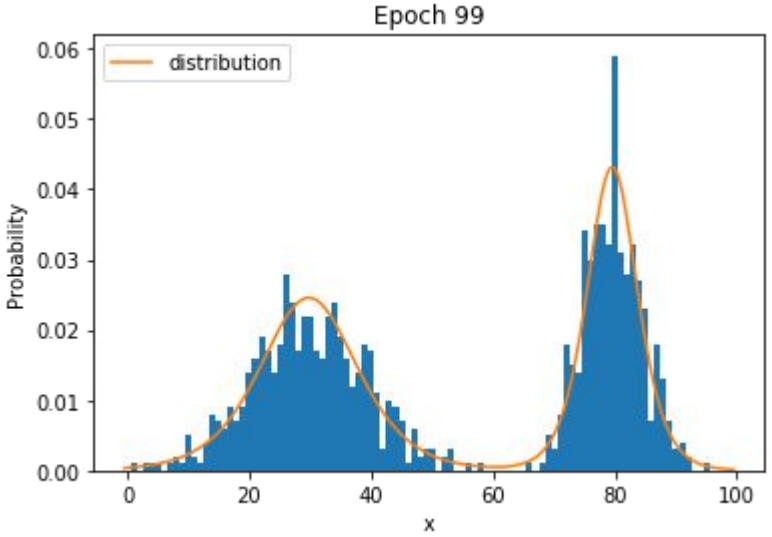


learned histogram = training data distribution

→ often poor generalization

Solution: Parameterized Distributions





Fitting a parameterized distribution often generalizes better

Learning a Parameterized Generative Model

- Recall: the goal is to **estimate p_{data} from $x^{(1)}, \ldots, x^{(n)} \sim p_{data}(x)$**
- We introduce a parameterized model $p_\theta(x)$ with the goal to learn θ so that $p_\theta(x) \approx p_{data}(x)$
- To learn θ, we pose a search problem over parameters

$$\arg\min_{\theta} \ \text{loss}(\theta, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)})$$

- Want the loss function + search procedure to:
  - Work with large datasets (n is large, say millions of training examples)
  - Yield θ such that p_θ matches p_{data} — i.e. the training algorithm *works.* Think of the loss as a distance between distributions.
  - Note that the training procedure can only see the empirical data distribution, not the true data distribution: we want the model to generalize.

Maximum Likelihood

- Maximum likelihood: given a dataset $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, find θ by solving the optimization problem

$$\arg\min_{\theta} \ \text{loss}(\theta, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) = \frac{1}{n} \sum_{i=1}^{n} -\log p_\theta(\mathbf{x}^{(i)})$$

- Statistics tells us that if the model family is expressive enough and if enough data is given, then solving the maximum likelihood problem will yield parameters that generate the data
- Equivalent to minimizing KL divergence between the empirical data distribution and the model

$$\hat{p}_{\text{data}}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[\mathbf{x} = \mathbf{x}^{(i)}]$$

$$\text{KL}(\hat{p}_{\text{data}} \,\|\, p_\theta) = \mathbb{E}_{\mathbf{x} \sim \hat{p}_{\text{data}}}[-\log p_\theta(\mathbf{x})] - H(\hat{p}_{\text{data}})$$

Stochastic Gradient Descent

Maximum likelihood is an optimization problem. How do we solve it?

Stochastic gradient descent (SGD).

- SGD minimizes expectations: for f a differentiable function of θ, it solves

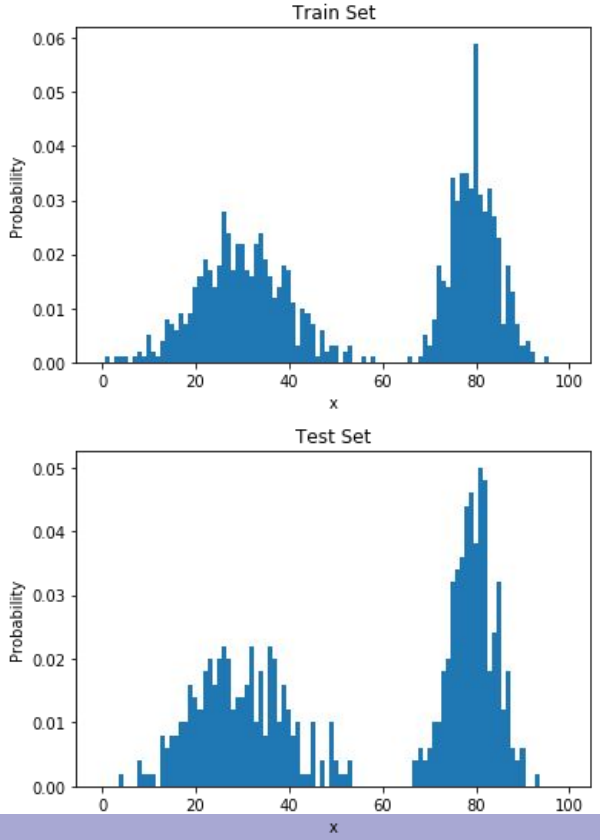
$$\arg\min_{\theta} \mathbb{E}[f(\theta)]$$

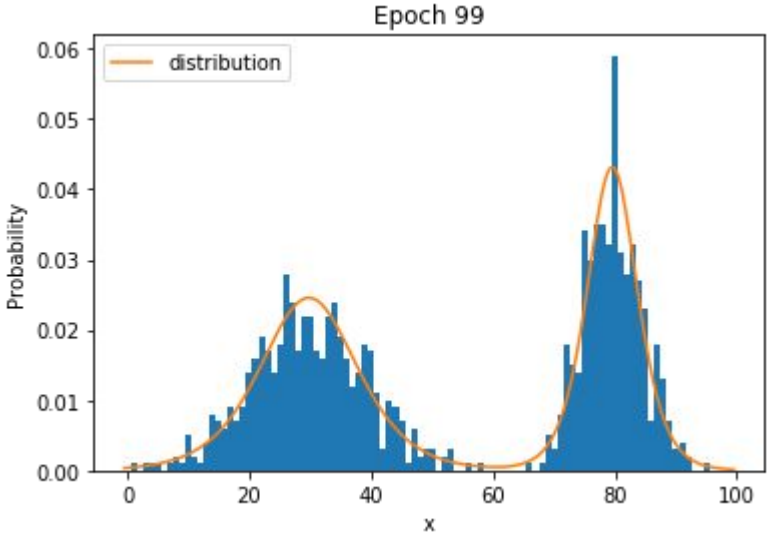
- With maximum likelihood, the optimization problem is

$$\arg\min_{\theta} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\text{data}}}[-\log p_{\theta}(\mathbf{x})]$$

- **Why maximum likelihood + SGD?** It works with large datasets and is compatible with neural networks.

Our Example Earlier was the Result of Fitting Parameterized Distribution with Maximum Likelihood





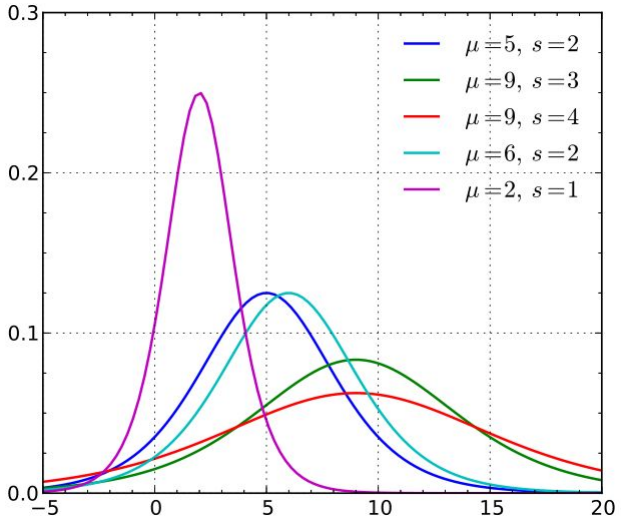
Fitting a parameterized distribution often generalizes better

How to represent the distribution?

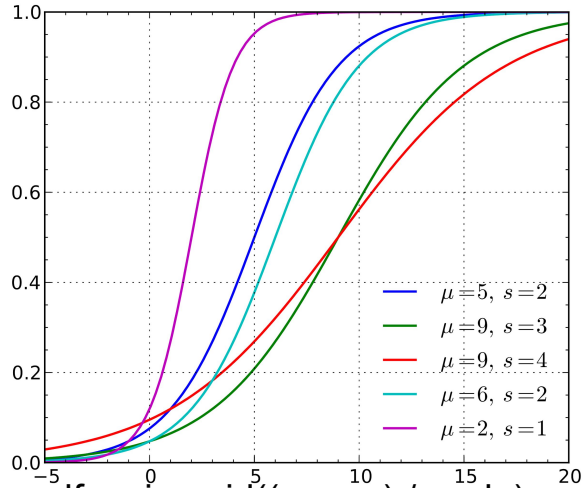
Data discrete, (many) smooth parameterized distributions continuous

→ Want easy access to cumulative distribution to allow for exact computation of probability mass in each interval + exact backprop → a good choice: logistic distribution

PS: alternatively take peace with just using the probability density at center of interval times width, but inexactness can often make debugging harder and could lead to learning (poorly performing) densities with extreme peaks



pdf



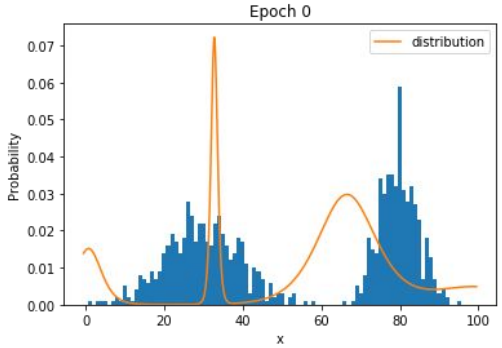
cdf = sigmoid((x - mu) / scale)

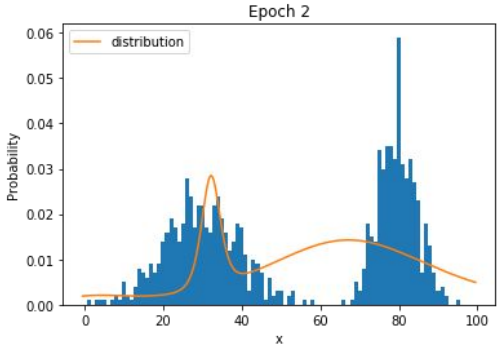
Mixture of Logistics for Multi-modal

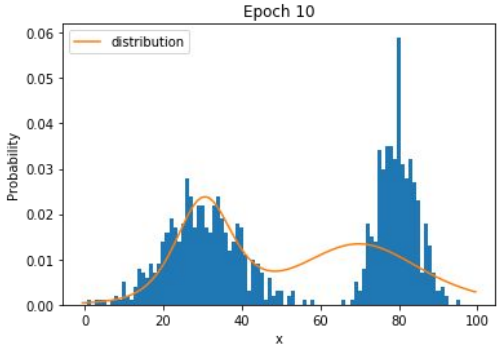
$$\nu \sim \sum_{i=1}^{K} \pi_i \text{logistic}(\mu_i, s_i)$$

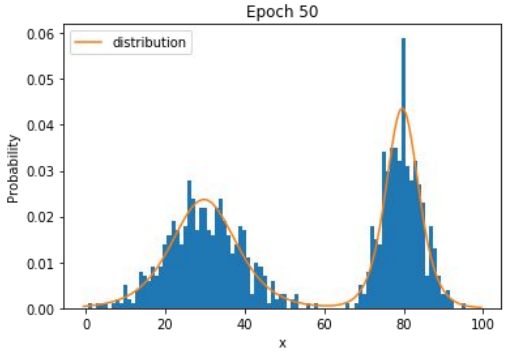
$$P(x|\pi, \mu, s) = \sum_{i=1}^{K} \pi_i \left[\sigma((x + 0.5 - \mu_i)/s_i) - \sigma((x - 0.5 - \mu_i)/s_i)\right],$$

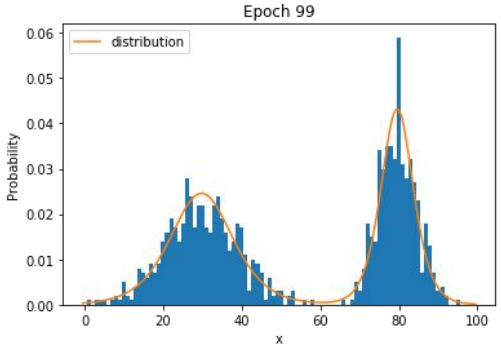
Ex. Training Mixture of Logistics

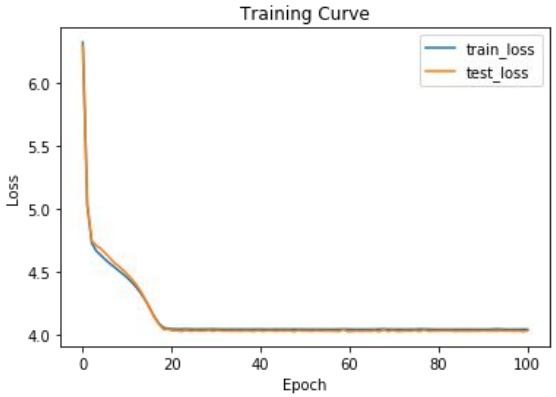












Autoregressive Models – Lecture Outline

- Motivation
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- **High-Dimensional Distributions**
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- Deeper Dive into Causal Masked Neural Models
  - Convolutional
  - Attention
  - Tokenization
  - Caching
- Other things to be aware of
  - Decoder-only vs. Encoder-Decoder models
  - New incarnations of Recurrent Models
  - Alternative / Complementary ideas to tokenization

Challenge of High Dimensional Data

Simply flattening into 1-D distribution doesn't work there are too many bins, even for simple cases:

- (Binary) MNIST: 28x28 images, each pixel in {0, 1}
- There are $2^{784} \approx 10^{236}$ possible images

  i.e. 10^{236} probabilities to estimate
- Any reasonable training set covers only a tiny fraction of this
- Each image influences only one parameter. No generalization whatsoever!

Chain Rule

- Any multi-variable distribution can be written as a product of conditionals

$$p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{1:i-1})$$

- This is called an **autoregressive model**.

- Are we done? No!
  - What we need to resolve still: how to efficiently represent and learn each conditional

AutoRegressive Models

- Recall, AutoRegressive Model $p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{1:i-1})$
- How to achieve efficiently representable and learnable parameterizations of the conditionals?
  - ***Solution 1: Bayes' Nets*** – sparsify conditioning

$$p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid Parents(x_i))$$

Bayes Nets

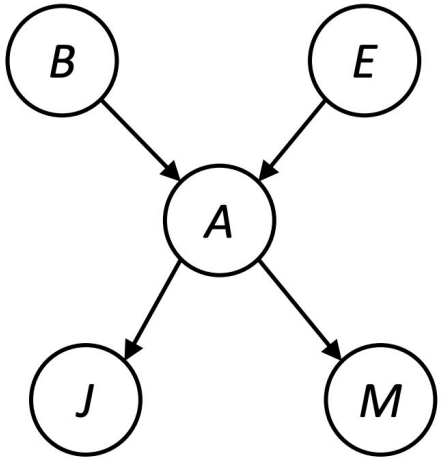
$$p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid Parents(x_i))$$

$$p_\theta(B, E, A, J, M) = p_\theta(B)p_\theta(E)p_\theta(A \mid B, E)p_\theta(J \mid A)p_\theta(M \mid A)$$

+ If the data has such underlying (causal?) structure, can be a great inductive bias
- But generally, sparsification introduces strong assumptions, limits expressivity
- In standard form has limited parameter sharing between conditionals (though in principle could introduce some sharing)

B	P(B)
+b	0.001
-b	0.999

E	P(E)
+e	0.002
-e	0.998



A	J	P(J|A)
+a	+j	0.9
+a	-j	0.1
-a	+j	0.05
-a	-j	0.95

A	M	P(M|A)
+a	+m	0.7
+a	-m	0.3
-a	+m	0.01
-a	-m	0.99

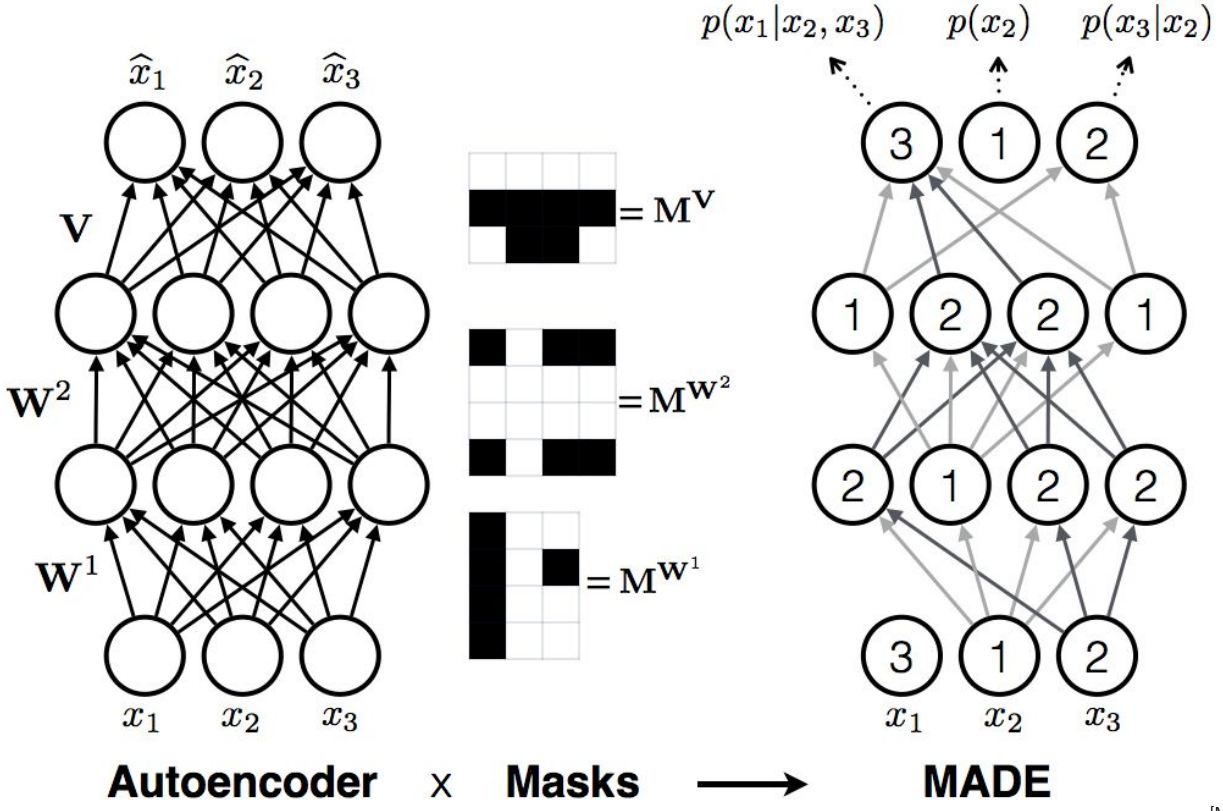
AutoRegressive Models

- Recall, AutoRegressive Model $p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{1:i-1})$
- How to achieve efficiently representable and learnable parameterizations of the conditionals?
  - ***Solution 1: Bayes' Nets*** – sparsify conditioning
    - Efficient, but: too strong an assumption in most cases, leads to poor fits

AutoRegressive Models

- Recall, AutoRegressive Model $p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{1:i-1})$
- How to achieve efficiently representable and learnable parameterizations of the conditionals?
  - ***Solution 1: Bayes' Nets*** – sparsify conditioning
    - Efficient, but: too strong an assumption in most cases, leads to poor fits
  - ***Solution 2: MADE*** – parameterize conditionals with neural net

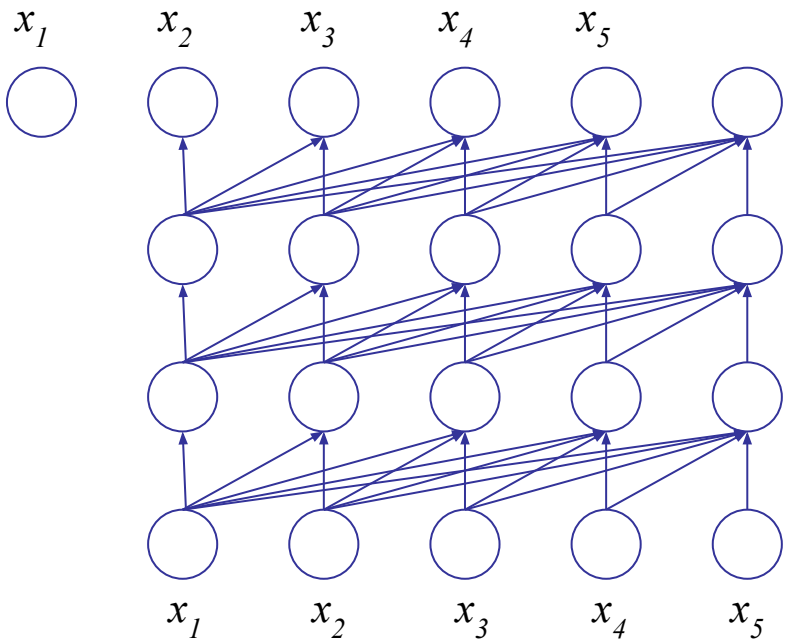
Masked Autoencoder for Distribution Estimation (MADE)



[MADE – Germain, Gregor, Murray, Larochelle, 2015]

Masked Autoencoder for Distribution Estimation (MADE)

In more modern notation/diagram



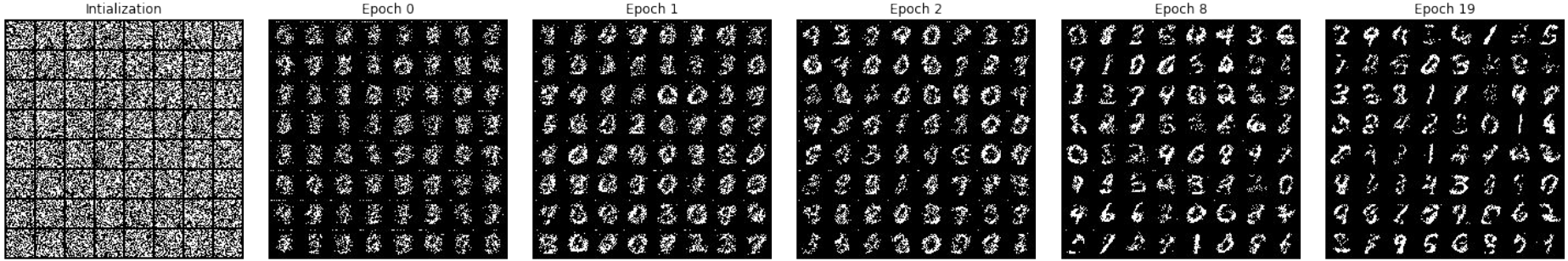
Every hidden layer node can be vector valued
Every edge can be its own MLP connection

MNIST

- Handwritten digits
- 28x28
- 60,000 train
- 10,000 test

- Original: greyscale
- “Binarized MNIST” -- 0/1 (black/white)

MADE on MNIST



Masked Autoencoder for Distribution Estimation (MADE)

```
# param: normal FC weight
# x: layer input
# y: autoregressive activations
mask = get_linear_ar_mask(in_size, out_size)
# create mask of pattern
# array([[0., 1., 1.],
#        [0., 0., 1.],
#        [0., 0., 0.]], dtype=float32)
y = tf.matmul(x, param * mask)
```

MADE results

Table 6. Negative log-likelihood test results of different models on the binarized MNIST dataset.

Model	$-\log p$	
RBM (500 h, 25 CD steps)	≈ 86.34	Intractable
DBM 2hl	≈ 84.62	
DBN 2hl	≈ 84.55	
DARN n_h=500	≈ 84.71	
DARN n_h=500, adaNoise	≈ 84.13	
MoBernoullis K=10	168.95	Tractable
MoBernoullis K=500	137.64	
NADE 1hl (fixed order)	88.33	
EoNADE 1hl (128 orderings)	87.71	
EoNADE 2hl (128 orderings)	85.10	
MADE 1hl (1 mask)	88.40	
MADE 2hl (1 mask)	89.59	
MADE 1hl (32 masks)	88.04	
MADE 2hl (32 masks)	86.64	

MADE results

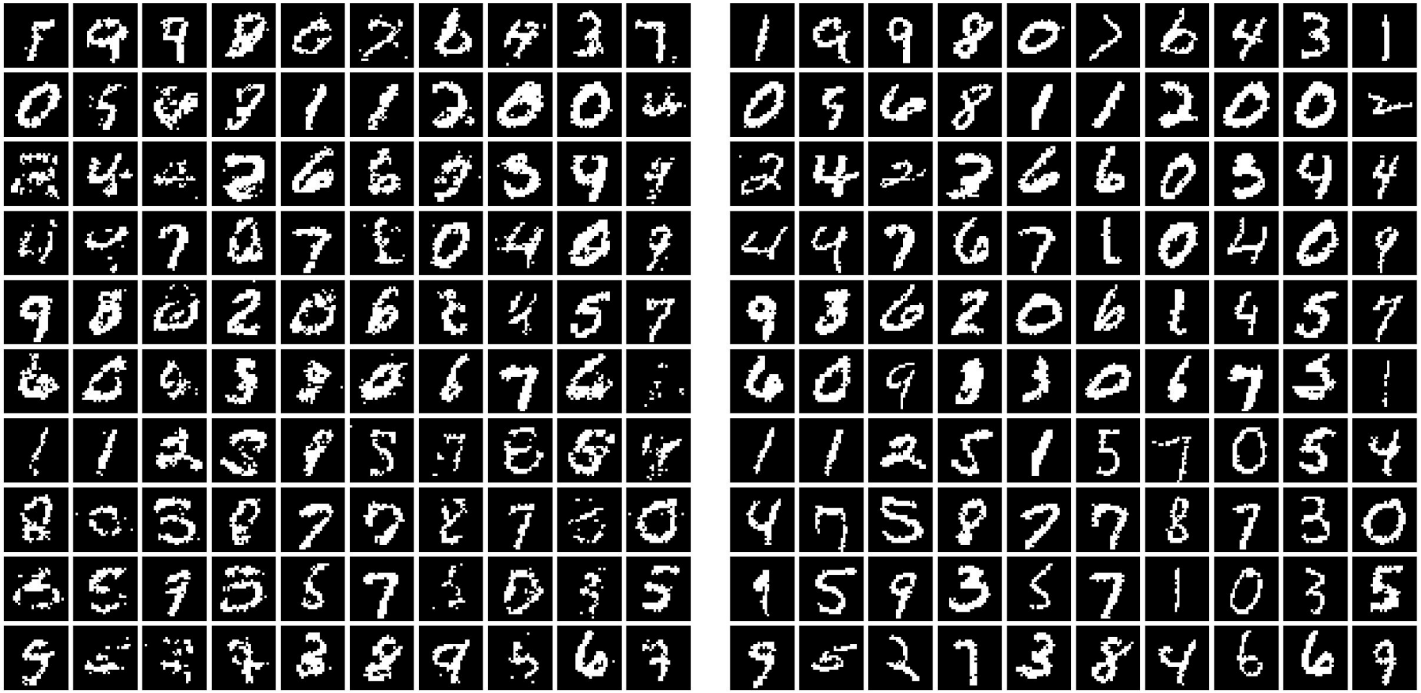
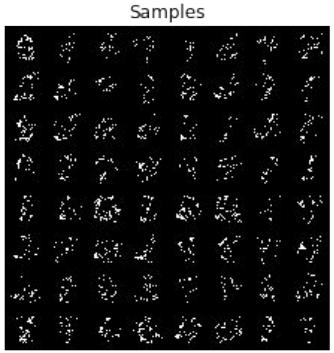


Figure 3. Left: Samples from a 2 hidden layer MADE. Right: Nearest neighbour in binarized MNIST.

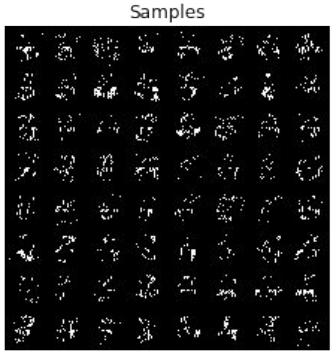
MADE -- Different Orderings

All orderings achieve roughly the same negative log likelihood loss, but samples are different

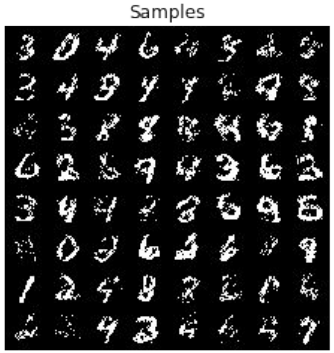
Random Permutation



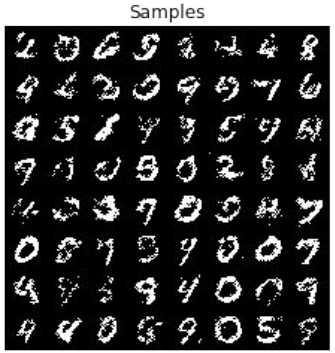
Even then Odd Indices



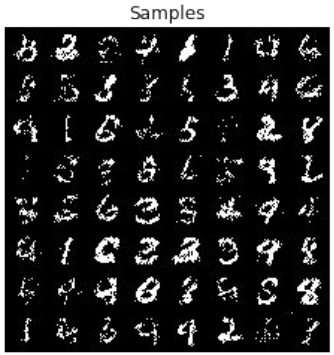
Rows (Raster Scan)



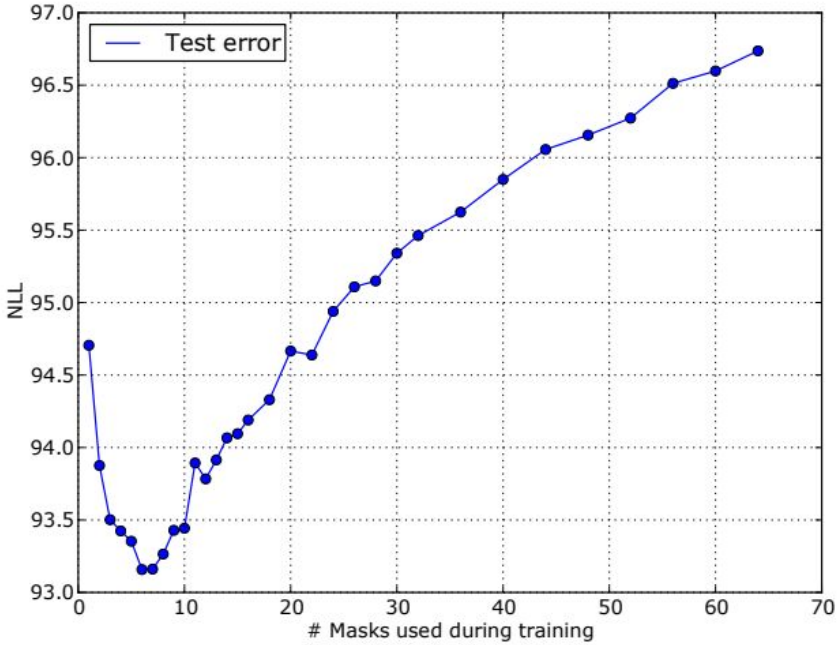
Columns



Top to Middle,
Bottom to Middle



MADE: Multiple Orderings



AutoRegressive Models

- Recall, AutoRegressive Model $p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{1:i-1})$
- How to achieve efficiently representable and learnable parameterizations of the conditionals?
  - ***Solution 1: Bayes' Nets*** – sparsify conditioning
    - Efficient, but: too strong an assumption in most cases, leads to poor fits
  - ***Solution 2: MADE*** – parameterize conditionals with neural net
    - Expressive, but: not enough parameter sharing for efficient learning

AutoRegressive Models

- Recall, AutoRegressive Model $p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{1:i-1})$
- How to achieve efficiently representable and learnable parameterizations of the conditionals?
  - ***Solution 1: Bayes' Nets*** – sparsify conditioning
    - Efficient, but: too strong an assumption in most cases, leads to poor fits
  - ***Solution 2: MADE*** – parameterize conditionals with neural net
    - Expressive, but: not enough parameter sharing for efficient learning
  - ***Solution 3: Causal Masked Neural Models***
    - parameterize conditionals with neural net (aka MADE)
    - + parameter sharing across conditionals
    - + add coordinate coding to still be able to individualize conditionals

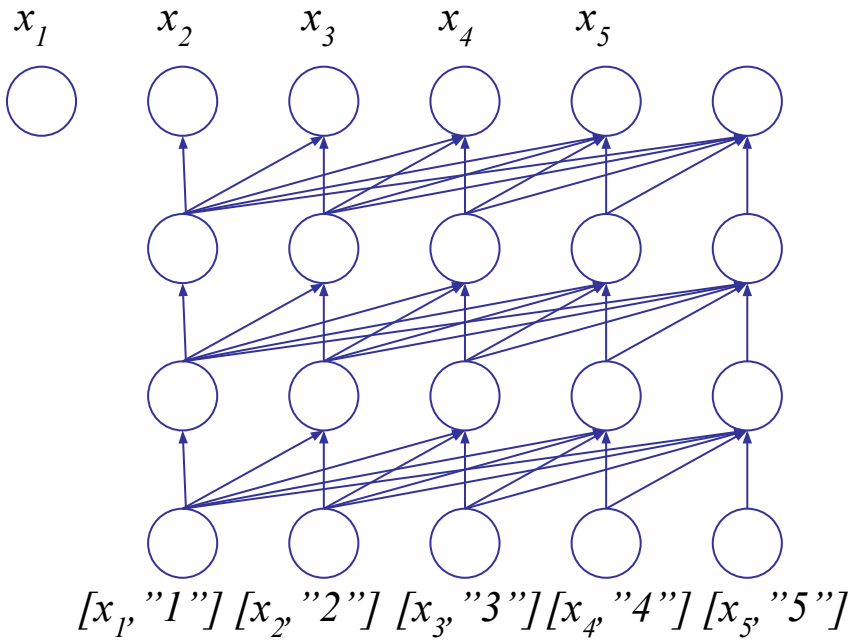
Causal Masked Neural Models

MADE:

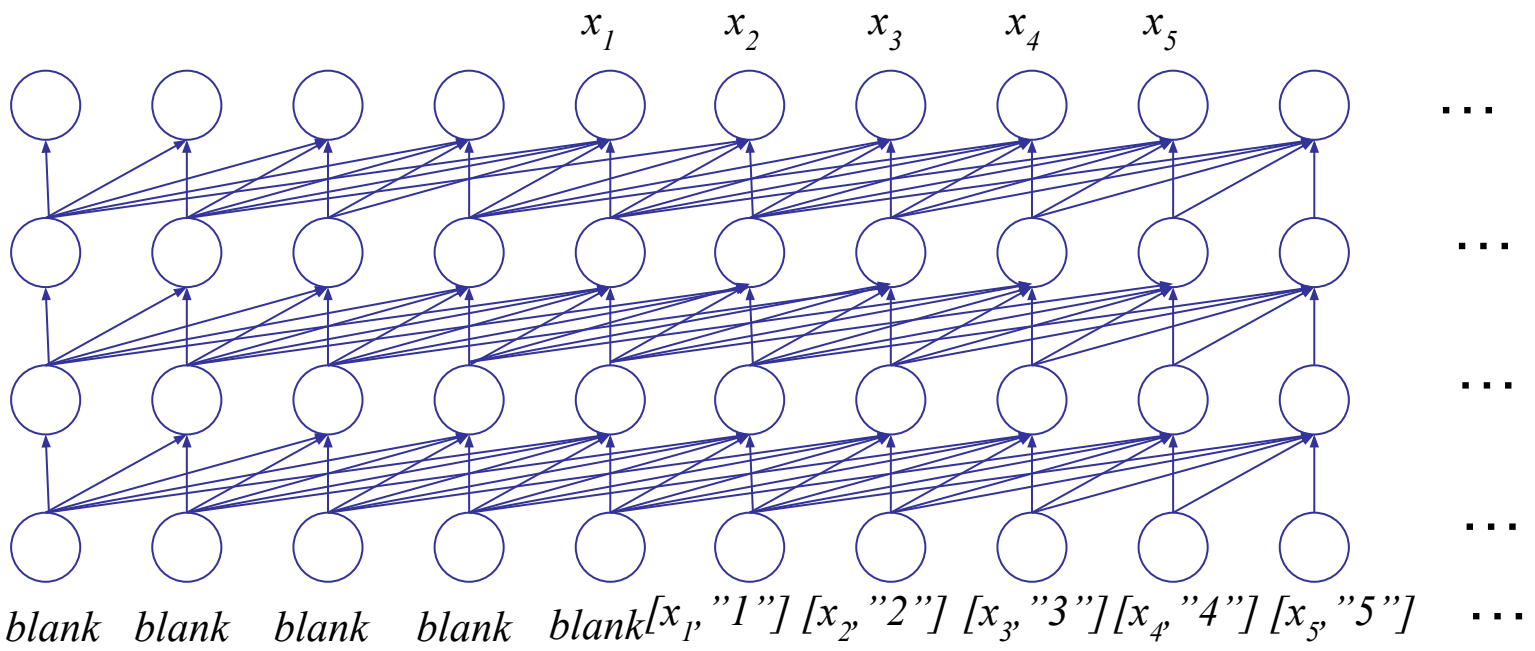
- Every hidden layer node can be vector valued
- Every edge can be its own MLP connection

Modern Causal Masked Neural Models:

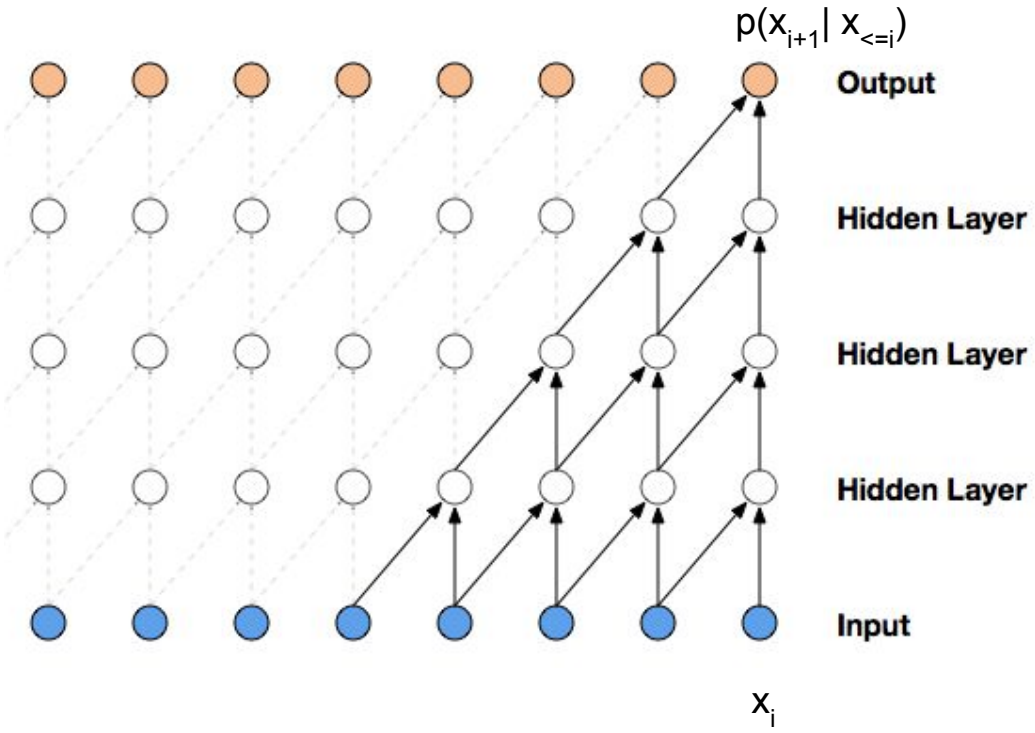
- Every hidden layer node can be vector valued
- Same parameters when shifting from column i to i+1
- To retain ability to know where we are in sequence: add coordinate coding
- Uniformize fully with padding



Causal Masked Neural Models



Masked Temporal (1D) Convolution



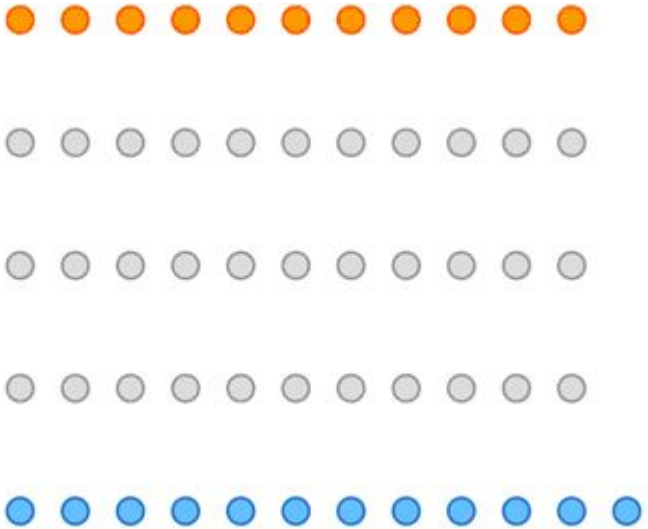
- Easy to implement, masking part of the conv kernel
- Constant parameter count for variable-length distribution!
- Efficient to compute, convolution has hyper-optimized implementations on all hardware

However

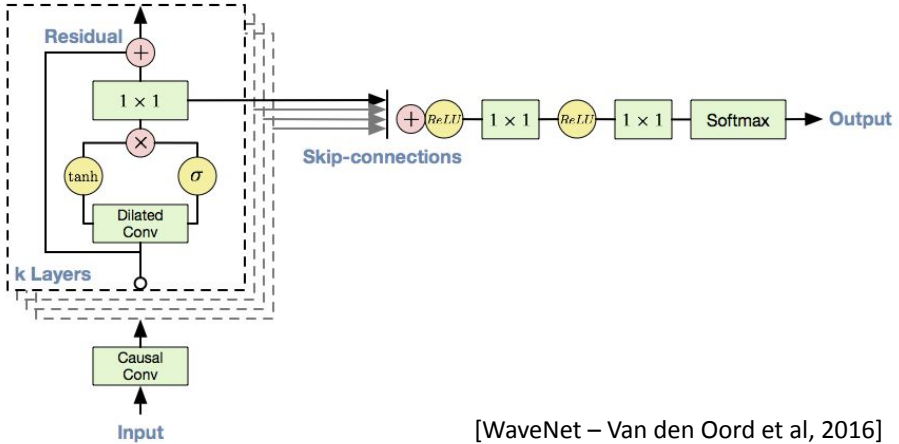
- Limited receptive field, linear in number of layers

[WaveNet – Van den Oord et al, 2016]

WaveNet

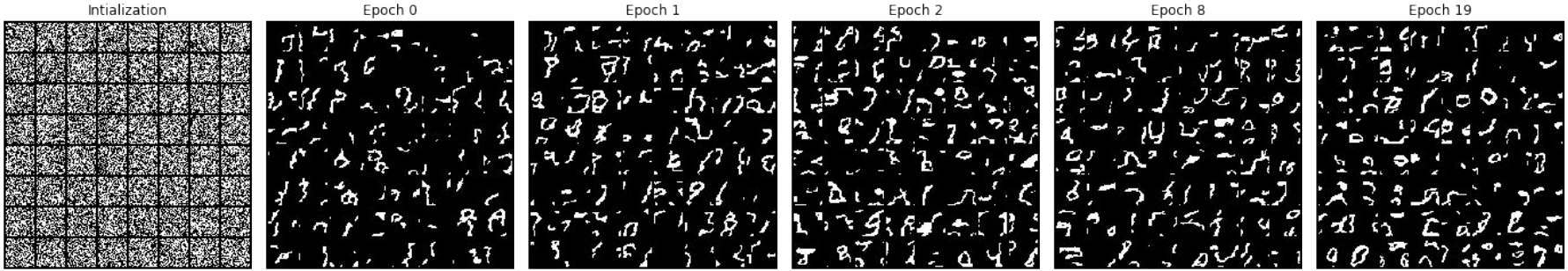


- Improved receptive field: dilated convolution, with exponential dilation
- Better expressivity: Gated Residual blocks, Skip connections



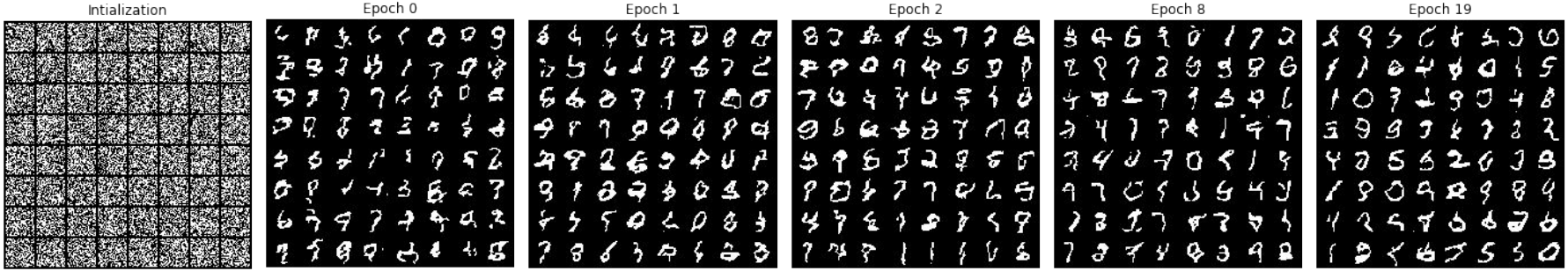
[WaveNet – Van den Oord et al, 2016]

WaveNet on MNIST



WaveNet with Pixel Location Appended on MNIST

- Append (x,y) coordinates of pixel in the image as input to WaveNet



Masked Temporal (1D) Convolution

```
# More efficient implementation possible by
#  padding instead of masking kernels
# k: size of kernel
# kernel: convolution weights
padded_x = tf.pad(x, [
        (0, 0), (k - 1, 0),
        (0, 0), (0, 0)
])
y = tf.nn.conv2d(padded_x, kernel, padding='VALID')
```

AutoRegressive Models

- Recall, AutoRegressive Model $p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{1:i-1})$
- How to achieve efficiently representable and learnable parameterizations of the conditionals?
  - ***Solution 1: Bayes' Nets*** – sparsify conditioning
    - Efficient, but: too strong an assumption in most cases, leads to poor fits
  - ***Solution 2: MADE*** – parameterize conditionals with neural net
    - Expressive, but: not enough parameter sharing for efficient learning
  - ***Solution 3: Causal Masked Neural Models***
    - parameterize conditionals with neural net (aka MADE)
      + parameter sharing across conditionals
      + add coordinate coding to still be able to individualize conditionals
    - Expressive and efficient!
      Any buts? possible concern is finite context window, but in practice quite large + retrieval can help

AutoRegressive Models

- Recall, AutoRegressive Model $p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{1:i-1})$
- How to achieve efficiently representable and learnable parameterizations of the conditionals?
  - ***Solution 1: Bayes' Nets*** – sparsify conditioning
    - Efficient, but: too strong an assumption in most cases, leads to poor fits
  - ***Solution 2: MADE*** – parameterize conditionals with neural net
    - Expressive, but: not enough parameter sharing for efficient learning
  - ***Solution 3: Causal Masked Neural Models***
    - parameterize conditionals with neural net (aka MADE)
    - + parameter sharing across conditionals
    - + add coordinate coding to still be able to individualize conditionals
    - Expressive and efficient!

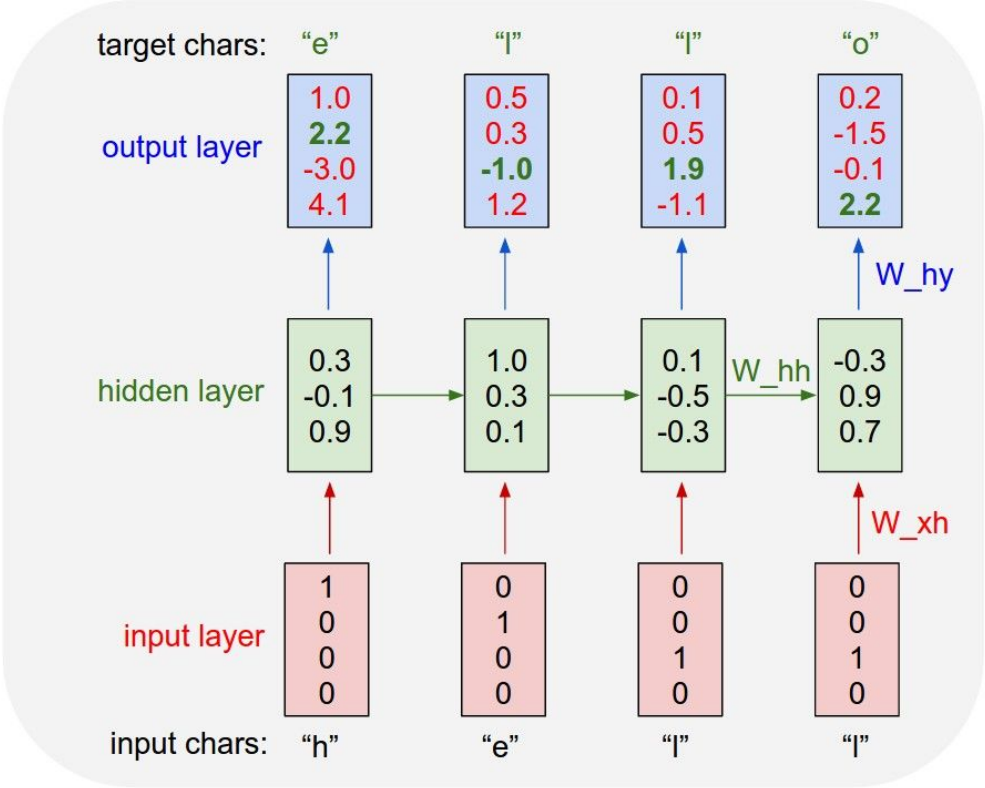
      Any buts? possible concern is finite context window, but in practice quite large + retrieval can help
  - ***Solution 4: Recurrent Neural Net*** – parameter sharing + "infinite look-back"

RNN autoregressive models - char-rnn

$$\log p(\mathbf{x}) = \sum_{i=1}^{d} \log p(x_i \mid \mathbf{x}_{1:i-1})$$

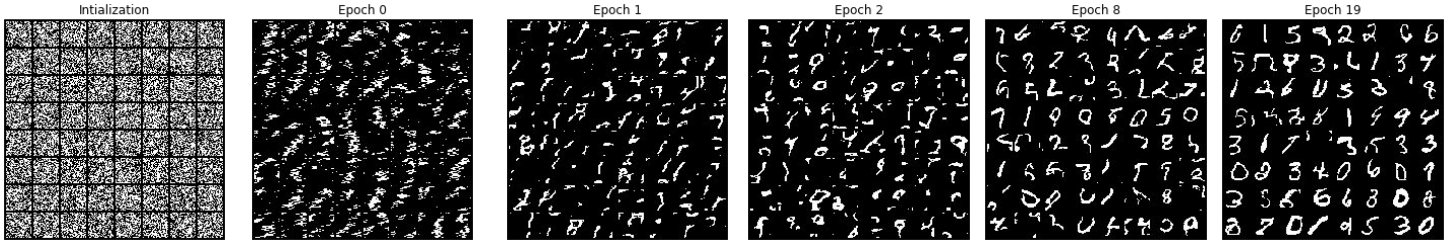
Sequence of characters

Character at i^{th} position



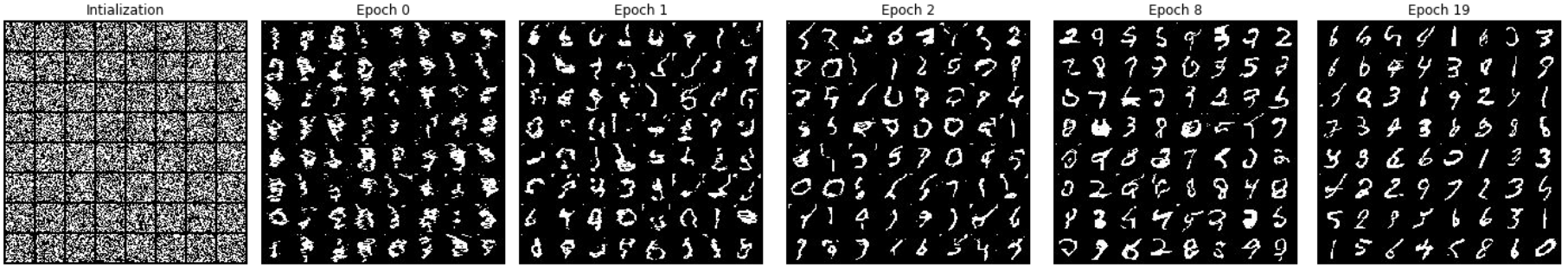
[Diagram: Karpathy, 2015; earlier papers: A Neural Probabilistic Model – Bengio et al, 2003; Sequence to Sequence Learning with NNs – Sutskever, Vinyals, Le, 2014]

RNN on MNIST



RNN with Pixel Location Appended on MNIST

- Append (x,y) coordinates of pixel in the image as input to RNN

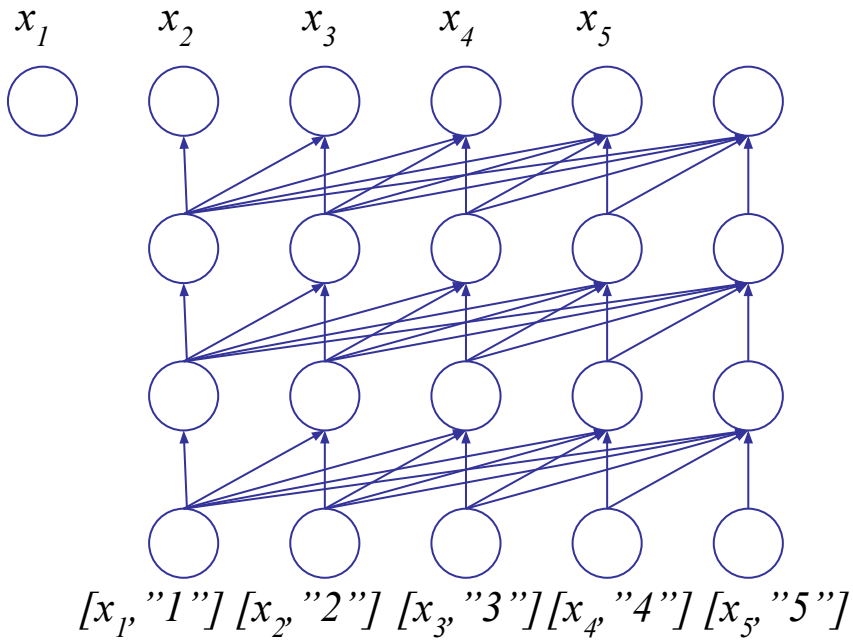


AutoRegressive Models

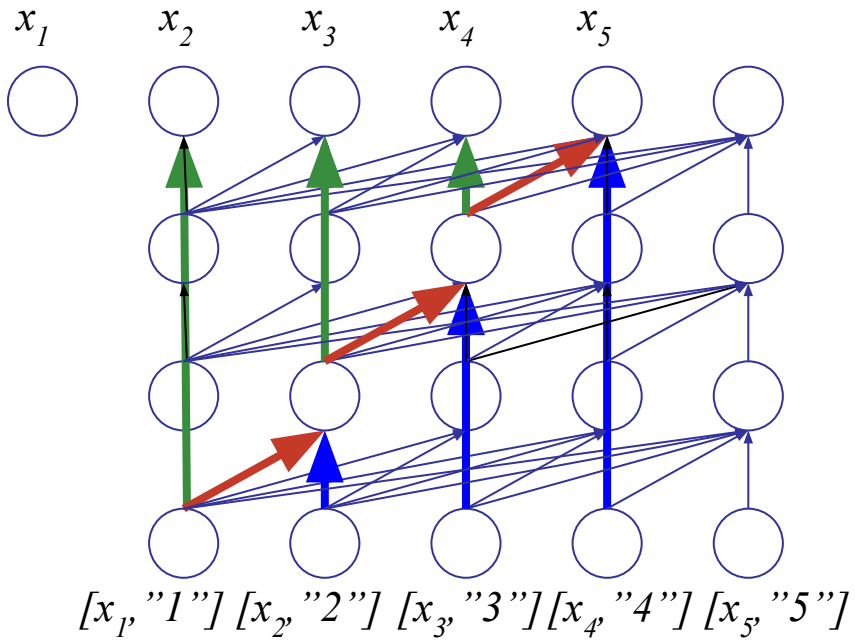
- Recall, AutoRegressive Model $p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{1:i-1})$
- How to achieve efficiently representable and learnable parameterizations of the conditionals?
  - ***Solution 1: Bayes' Nets*** – sparsify conditioning
    - Efficient, but: too strong an assumption in most cases, leads to poor fits
  - ***Solution 2: MADE*** – parameterize conditionals with neural net
    - Expressive, but: not enough parameter sharing for efficient learning
  - ***Solution 3: Causal Masked Neural Models***
    - parameterize conditionals with neural net (aka MADE)
    - + parameter sharing across conditionals
    - + add coordinate coding to still be able to individualize conditionals
    - Expressive and efficient!

      Any buts? possible concern is finite context window, but in practice quite large + retrieval can help
  - ***Solution 4: Recurrent Neural Net*** – parameter sharing + "infinite look-back"
    - Expressive, but: in practice doesn't tend to work as well as next proposal
      - not as amenable to parallelization
      - backprop through time can have exploding / vanishing gradients (there are tricks)
      - hard to truly have signal propagate from long history (i.e. benefit less than "advertised")
      - expressive but maybe not sufficiently expressive / not the right inductive biases

Causal Masked Neural Models



Sidenote: RNN = Special, Highly Restrictive Causal Masked Model?



RNN ~ very deep causal masked model with very sparse connectivity:
- Encoder (skip)
- Main diagonal
- Decoder (skip)
- Parameter sharing for each of these

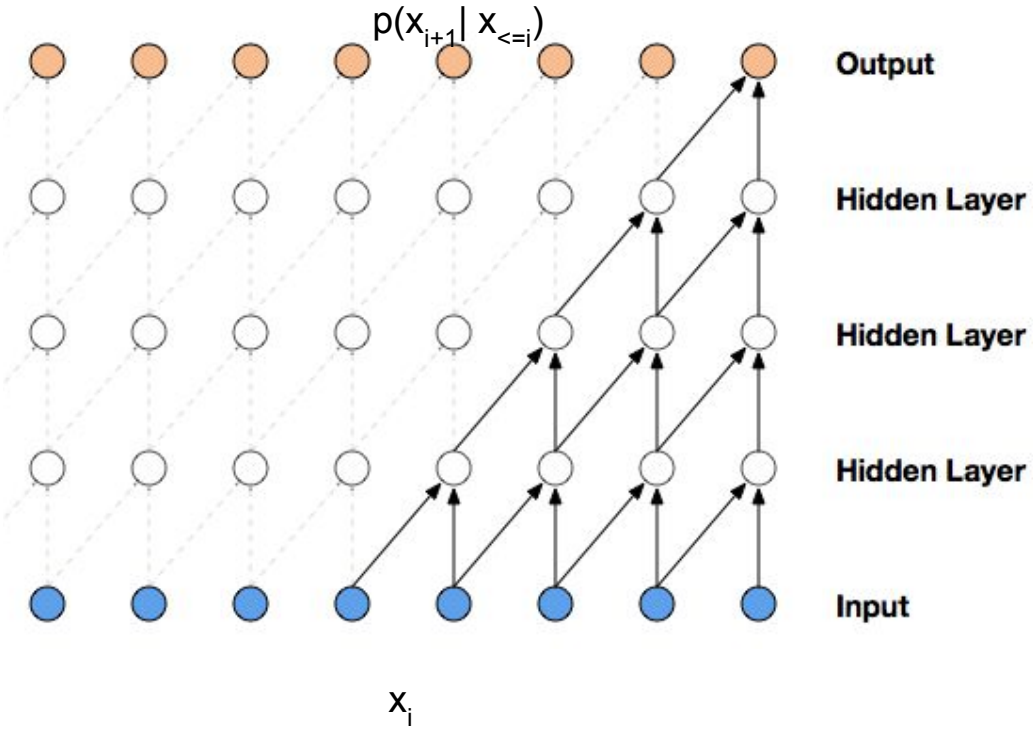
Autoregressive Models – Lecture Outline

- Motivation
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- **Deeper Dive into Causal Masked Neural Models**
  - Convolutional
  - Attention
  - Tokenization
  - Caching
- Other things to be aware of
  - Decoder-only vs. Encoder-Decoder models
  - New incarnations of Recurrent Models
  - Alternative / Complementary ideas to tokenization

Autoregressive Models – Lecture Outline

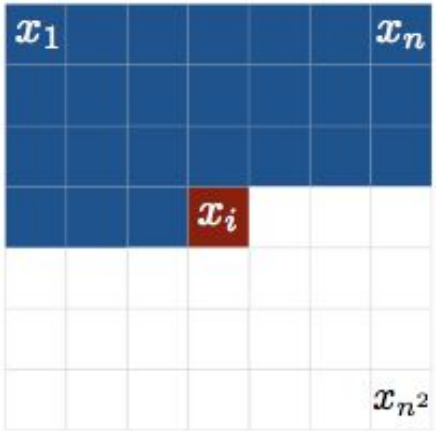
- Motivation
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- **Deeper Dive into Causal Masked Neural Models**
  - **Convolutional**
  - Attention
  - Tokenization
  - Caching
- Other things to be aware of
  - Decoder-only vs. Encoder-Decoder models
  - New incarnations of Recurrent Models
  - Alternative / Complementary ideas to tokenization

Earlier: Masked Temporal (1D) Convolution



Masked Spatial (2D) Convolution - PixelCNN

- Images can be flatten into 1D vectors, but they are fundamentally 2D
- We can use a masked variant of ConvNet to exploit this knowledge
- First, we impose an autoregressive ordering on 2D images:

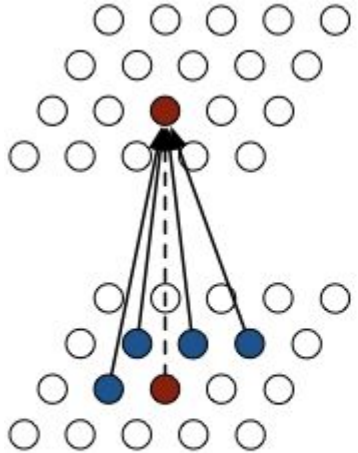


This is called raster scan ordering. (Different orderings are possible, more on this later)

PixelCNN

- Design question: how to design a masking method to obey that ordering?
- One possibility: PixelCNN (2016)

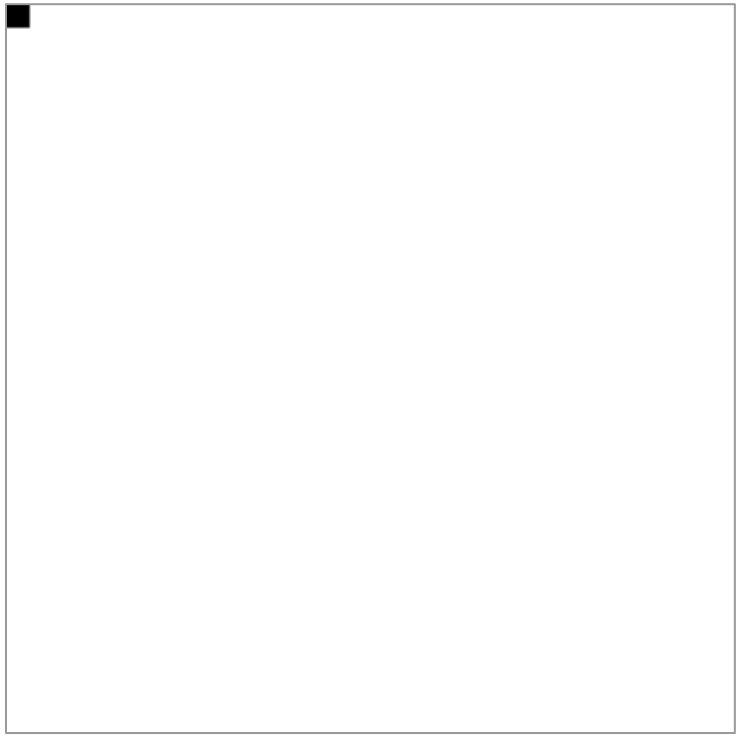
1	1	1
1	0	0
0	0	0



PixelCNN

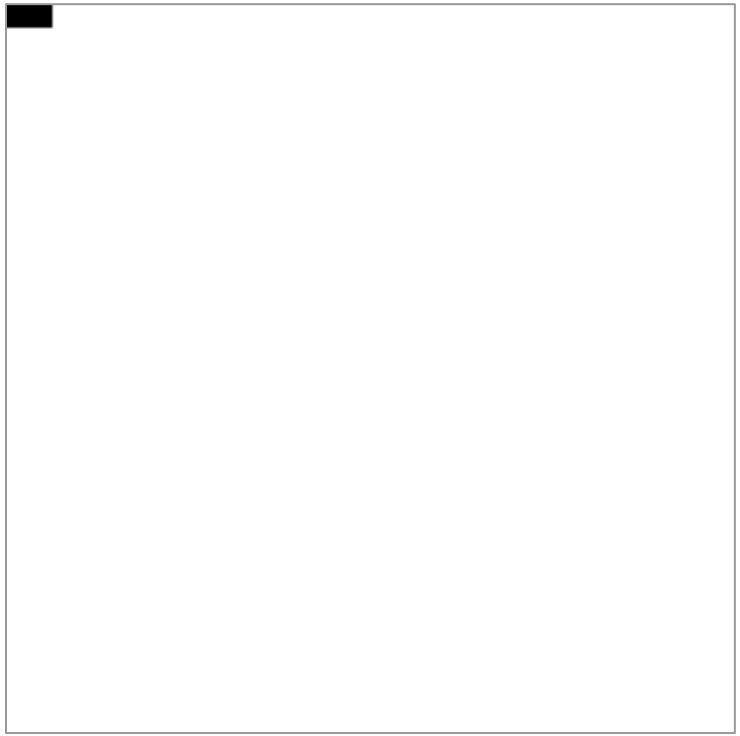
[PixelCNN – van den Oord et al, 2016]

Softmax Sampling



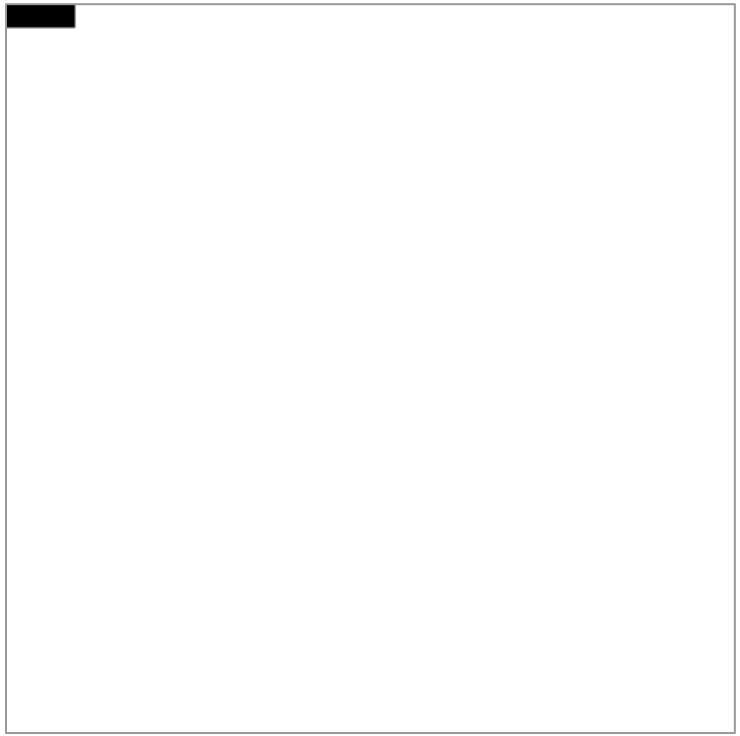
Slide by Aaron van den Oord

Softmax Sampling



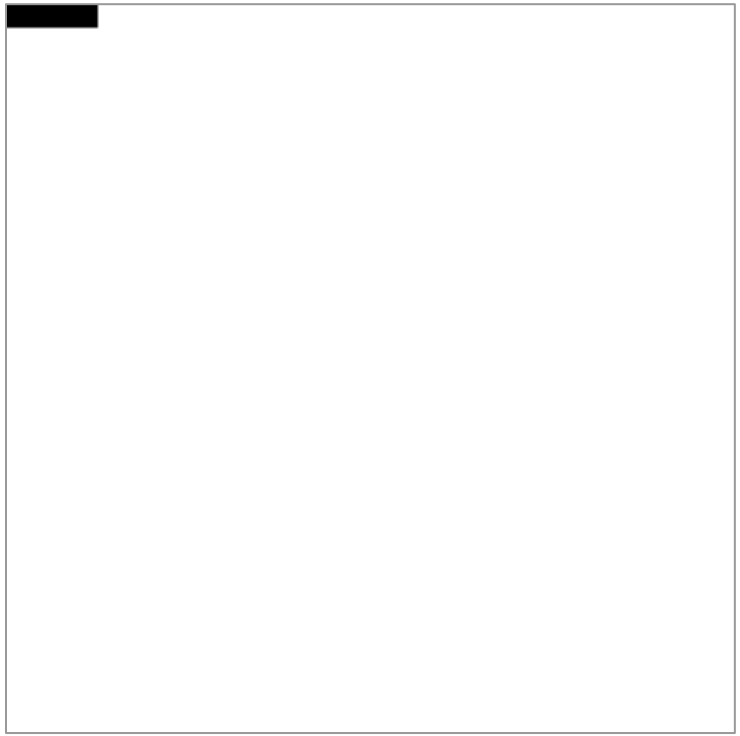
Slide by Aaron van den Oord

Softmax Sampling



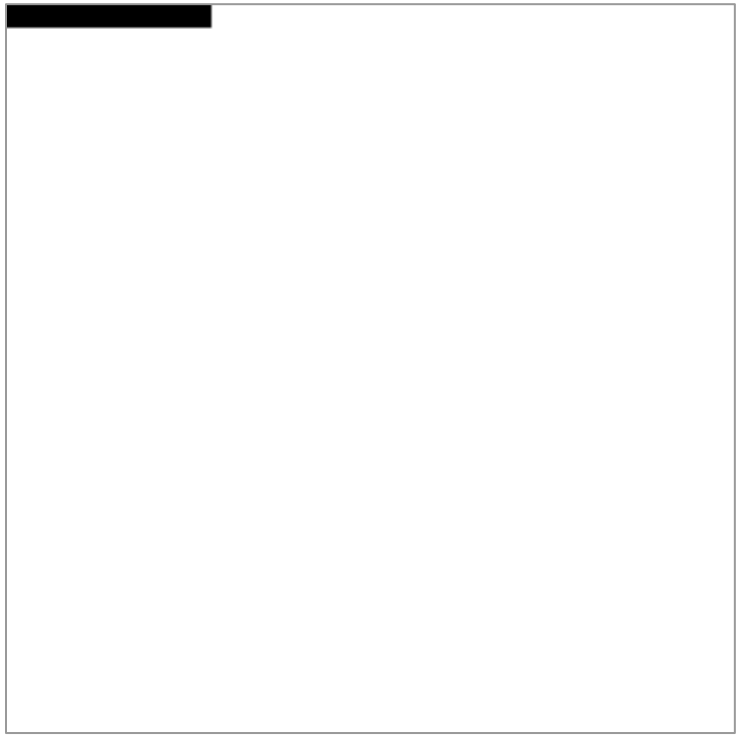
Slide by Aaron van den Oord

Softmax Sampling



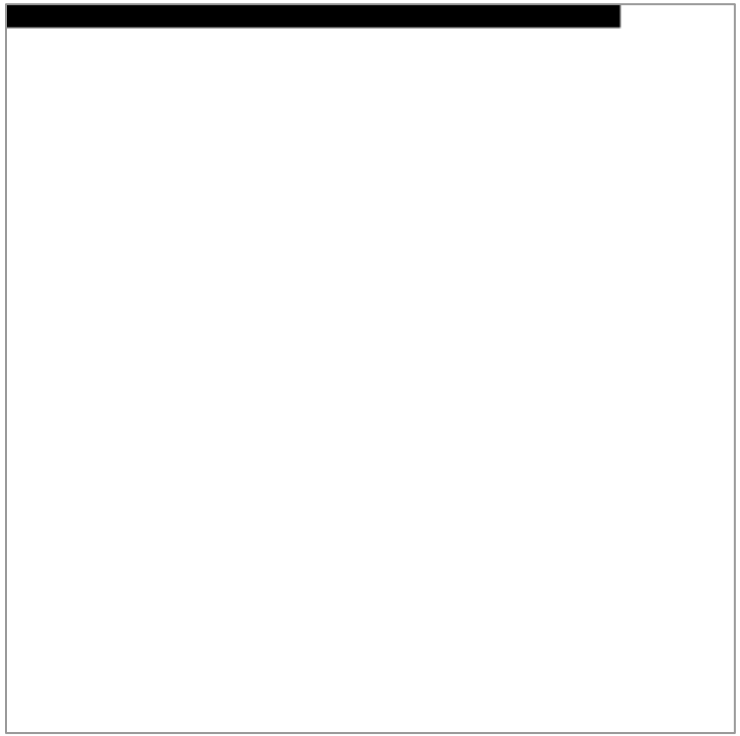
Slide by Aaron van den Oord

Softmax Sampling



Slide by Aaron van den Oord

Softmax Sampling



Slide by Aaron van den Oord

Softmax Sampling

Slide by Aaron van den Oord

Softmax Sampling

Slide by Aaron van den Oord

Softmax Sampling

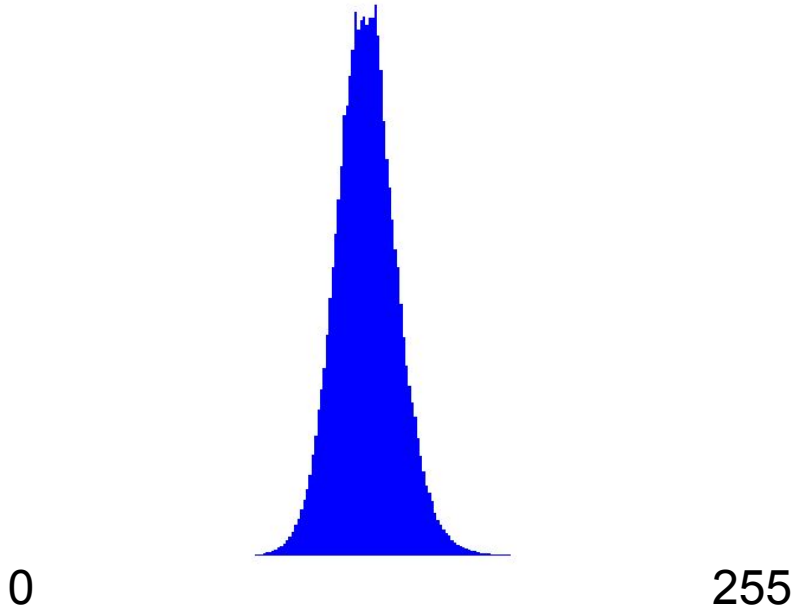
Slide by Aaron van den Oord

Softmax Sampling

Slide by Aaron van den Oord

Softmax Sampling

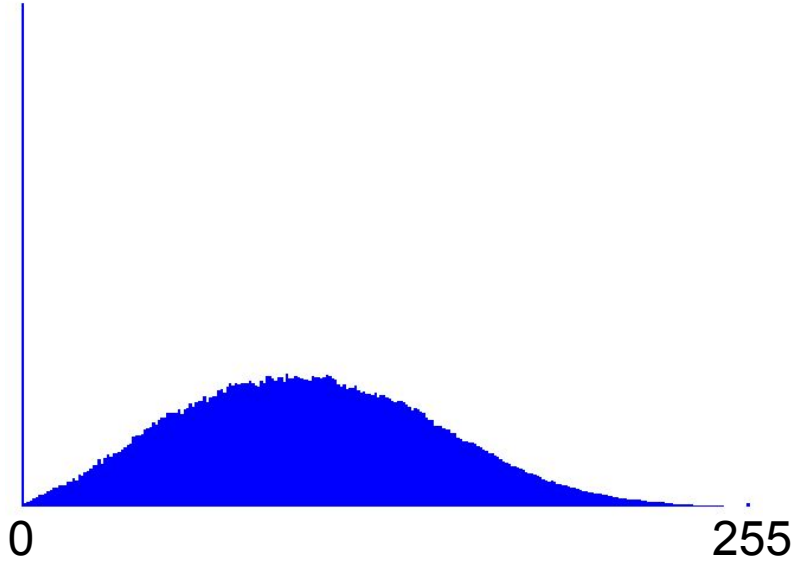
Softmax Sampling



Google DeepMind

Slide by Aaron van den Oord

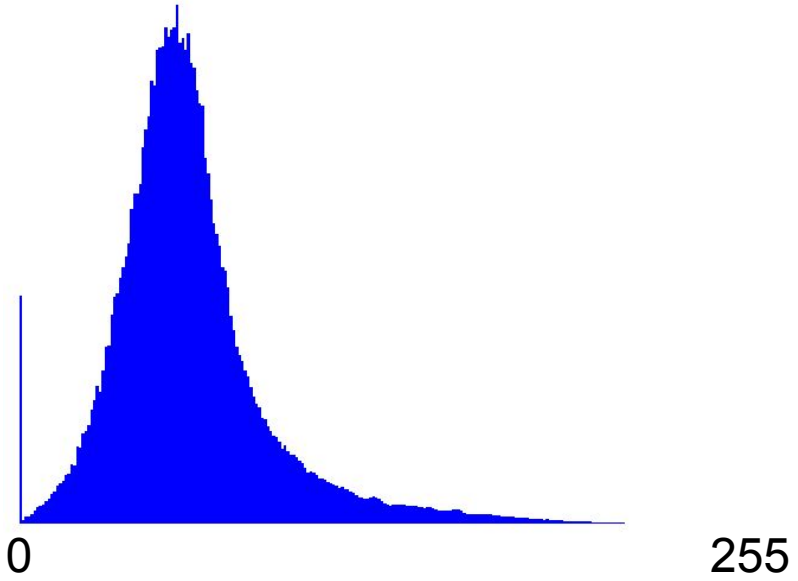
Softmax Sampling



Google DeepMind

Slide by Aaron van den Oord

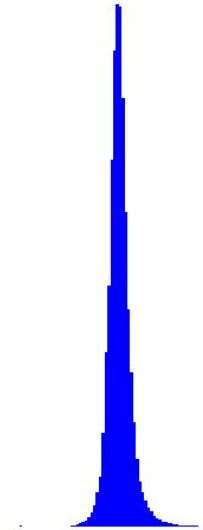
Softmax Sampling



Google DeepMind

Slide by Aaron van den Oord

Softmax Sampling



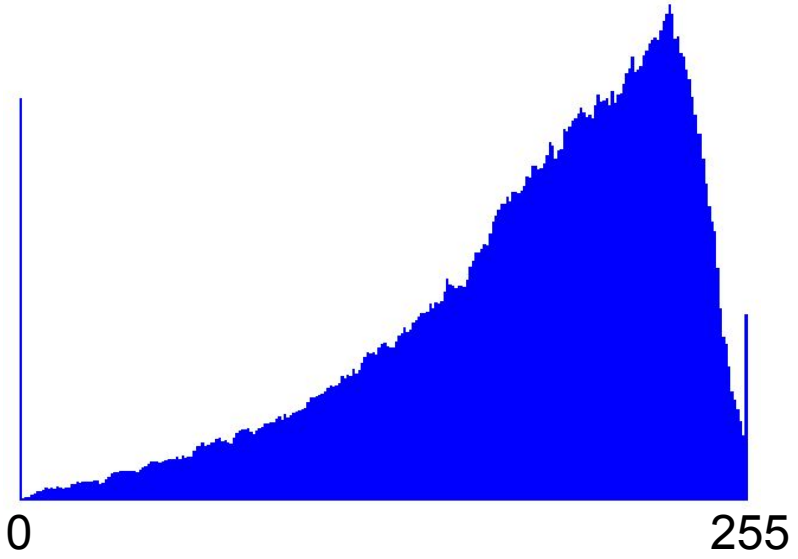
0

255

Google DeepMind

Slide by Aaron van den Oord

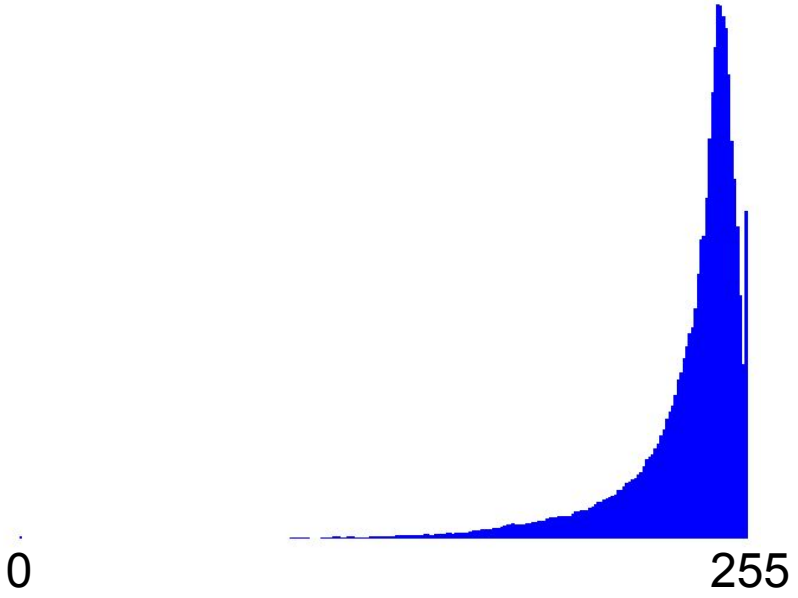
Softmax Sampling



Google DeepMind

Slide by Aaron van den Oord

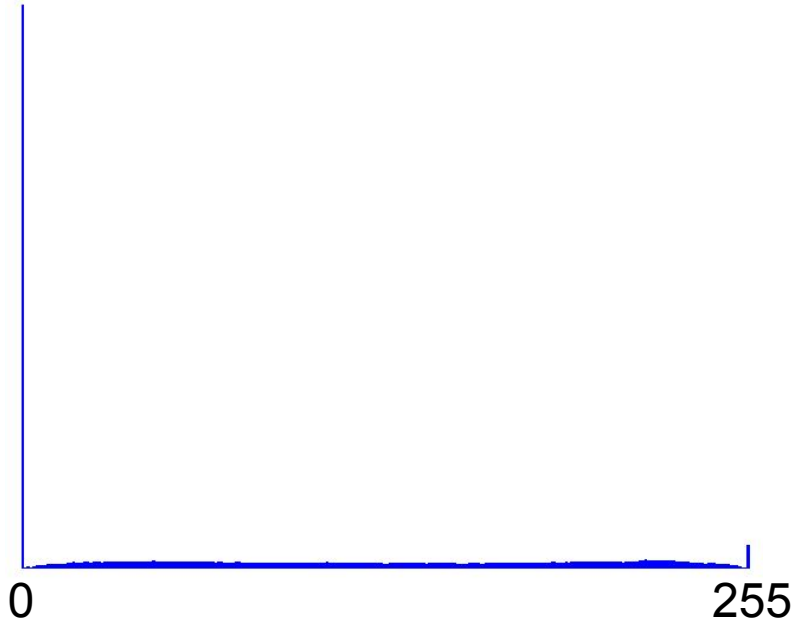
Softmax Sampling



Google DeepMind

Slide by Aaron van den Oord

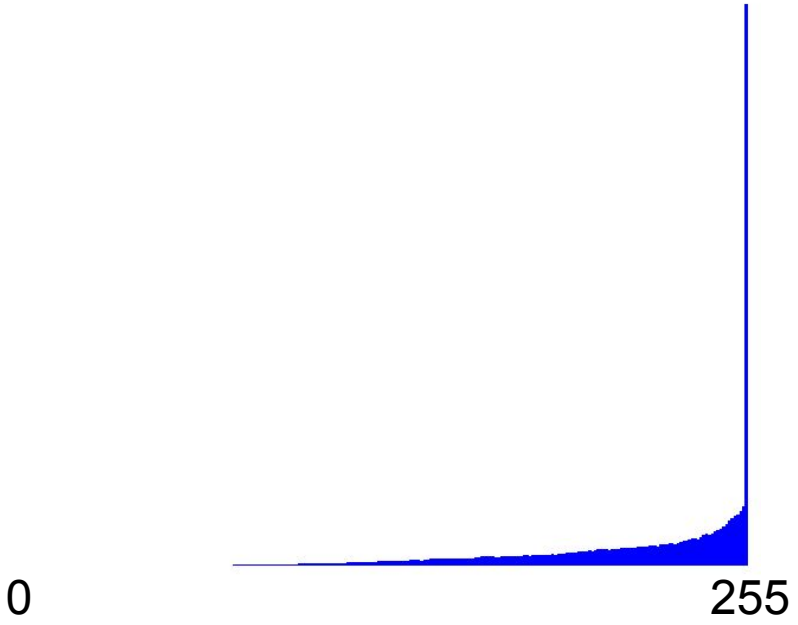
Softmax Sampling



Google DeepMind

Slide by Aaron van den Oord

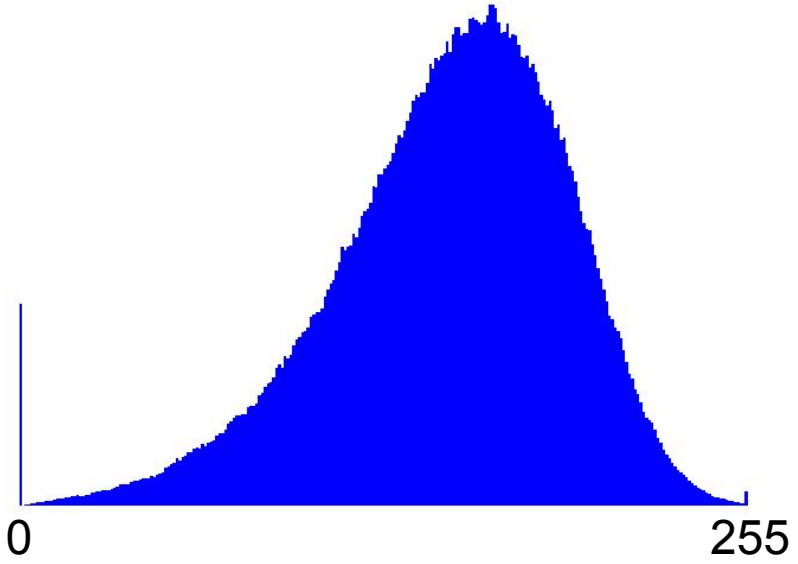
Softmax Sampling



Google DeepMind

Slide by Aaron van den Oord

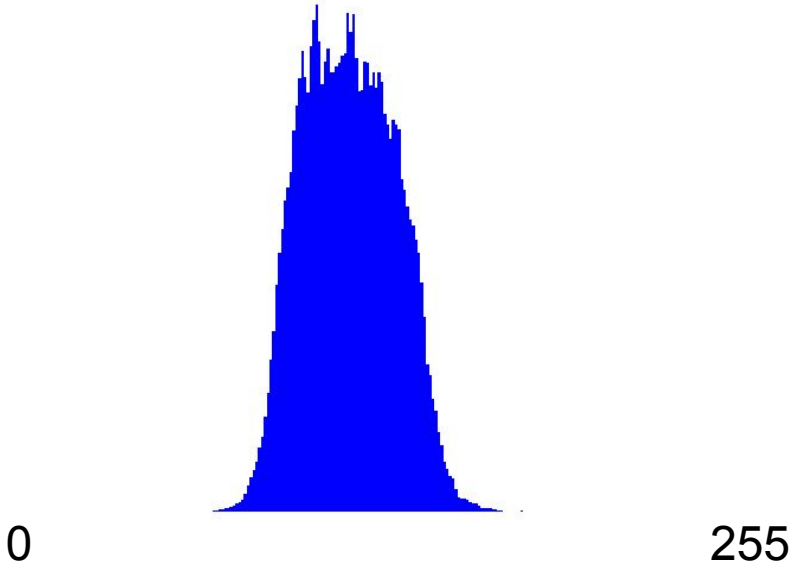
Softmax Sampling



Google DeepMind

Slide by Aaron van den Oord

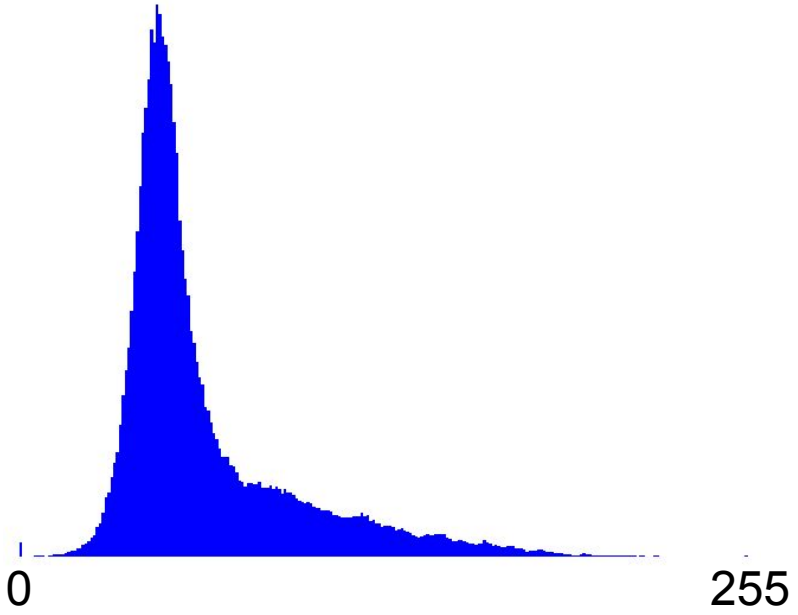
Softmax Sampling



Google DeepMind

Slide by Aaron van den Oord

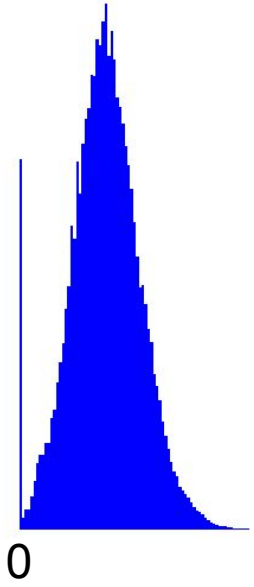
Softmax Sampling



Google DeepMind

Slide by Aaron van den Oord

Softmax Sampling

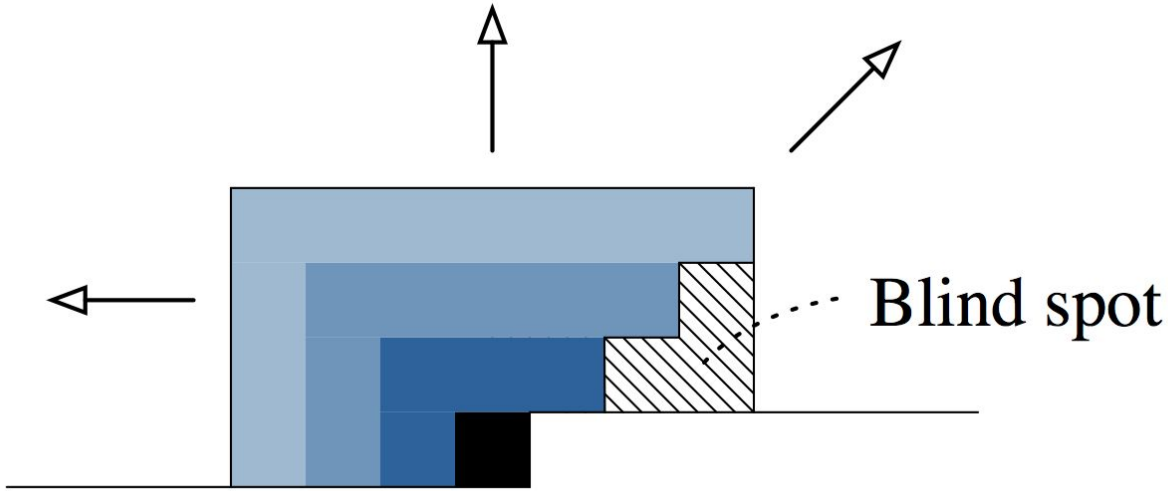


Google DeepMind

Slide by Aaron van den Oord

PixelCNN

- PixelCNN-style masking has one problem: blind spot in receptive field



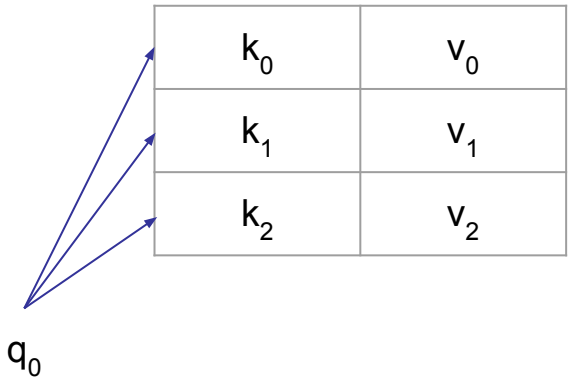
Autoregressive Models – Lecture Outline

- Motivation
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- **Deeper Dive into Causal Masked Neural Models**
  - Convolutional
  - **Attention**
  - Tokenization
  - Caching
- Other things to be aware of
  - Decoder-only vs. Encoder-Decoder models
  - New incarnations of Recurrent Models
  - Alternative / Complementary ideas to tokenization

Masked Attention

- A recurring problem for convolution: limited receptive field -> hard to capture long-range dependencies
- (Self-)Attention: an alternative that has
  - unlimited receptive field!!
  - also O(1) parameter scaling w.r.t. data dimension
  - parallelized computation (versus RNN)

Scaled Dot-Product Attention



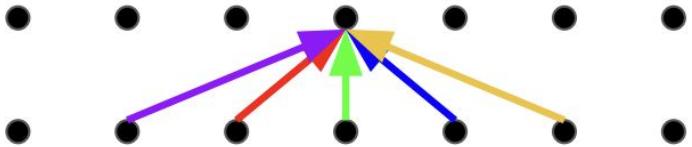
$$A(q, K, V) = \sum_i \frac{e^{s(q,k_i)}}{\sum_j e^{s(q,k_j)}} v_i$$

$$s(q, k) = \frac{q \cdot k}{\sqrt{d}}$$

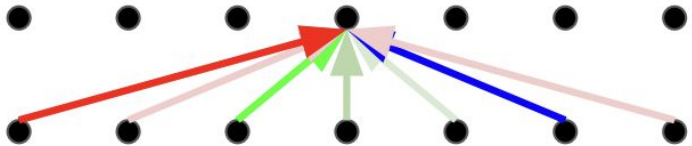
Matrix Form: QKV are L x D

$$A(Q, K, V) = \text{softmax}(QK^T)V$$

Self-Attention

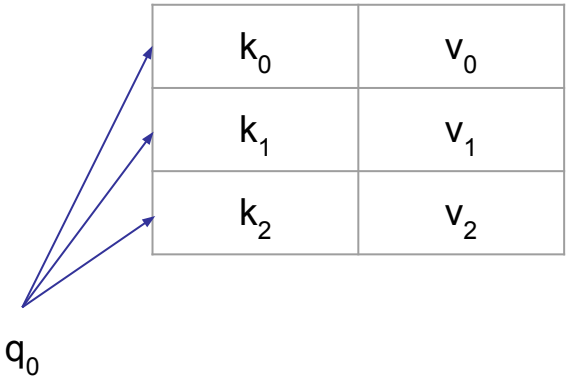


Convolution



Self-attention

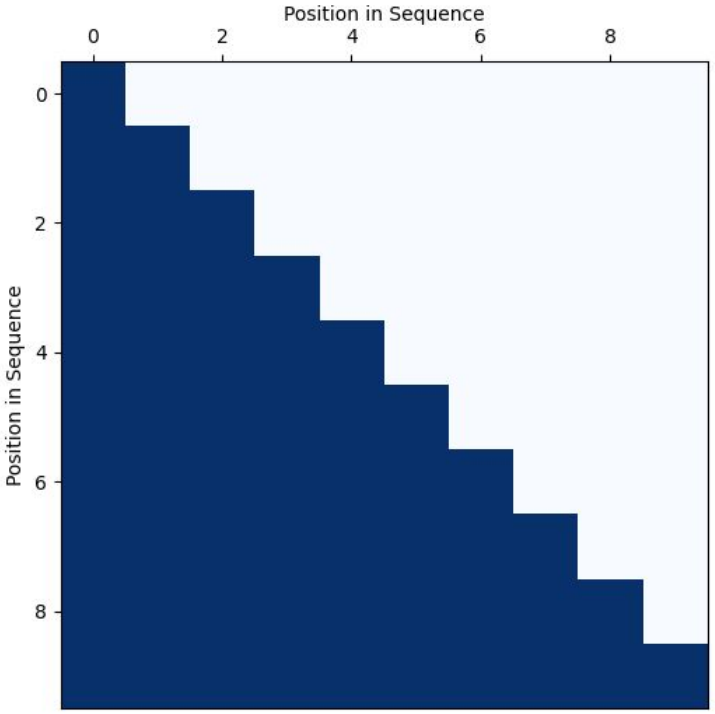
Self-Attention



$$A(Q, K, V) = \text{softmax}(QK^T)V$$

$$Q = XW_Q, K = XW_K, V = XW_V$$

Masked Attention



$$A(q, K, V) = \sum_i \frac{e^{s(q,k_i)}}{\sum_j e^{s(q,k_j)}} v_i$$

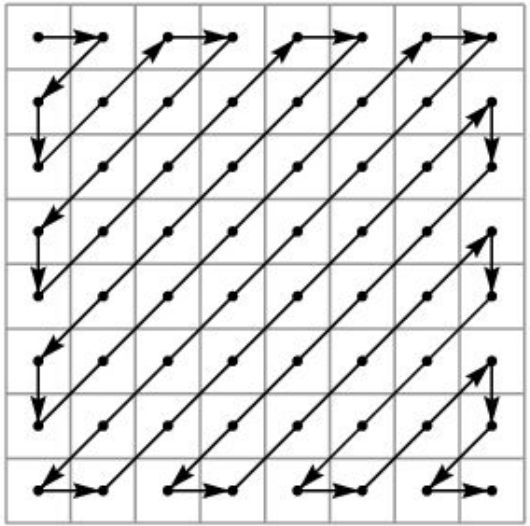
$$s(q, k) = \frac{q \cdot k}{\sqrt{d}} - (1 - \text{mask}(q, k)) * 10^{10}$$

Matrix Form

$$A(Q, K, V) = \text{softmax}(QK^T - (1 - M) * 10^{10})$$

Masked Attention

- Much more flexible than masked convolution. We can design any autoregressive ordering we want
- An example:



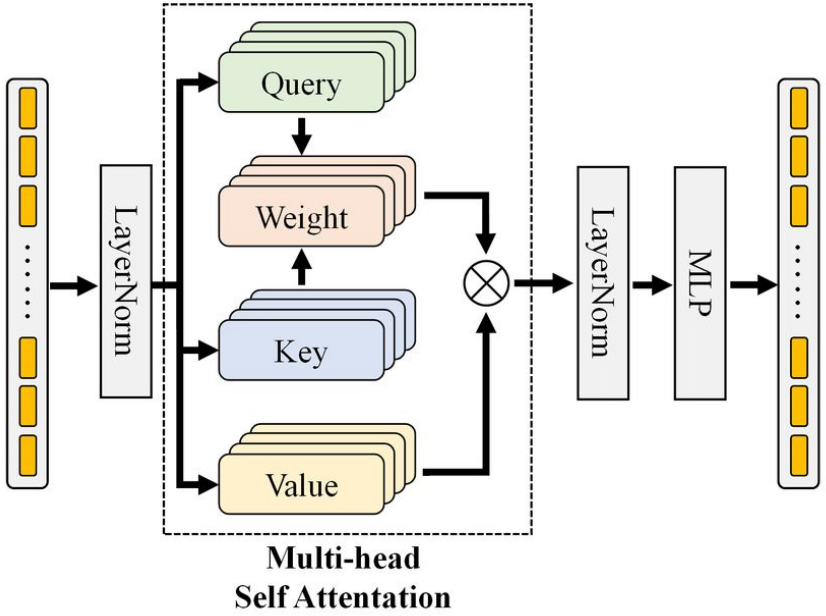
Zigzag ordering

- How to implement with masked conv?
- Trivial to do with masked attention!

Transformers

- Transformer Block
  - Multi-head Self-Attention (MHSA)
  - MLP

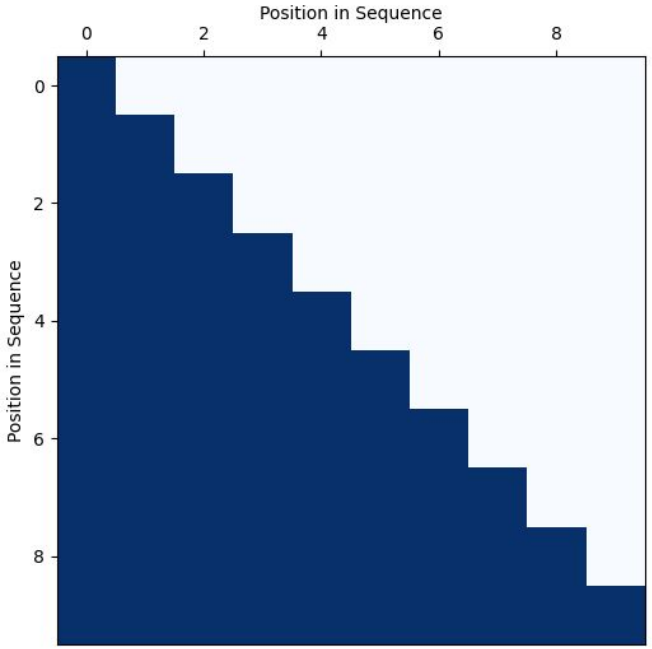
$$h = h + \mathrm{MHSA}(\mathrm{LayerNorm}(\mathrm{h}))$$
$$h = h + \mathrm{MLP}(\mathrm{LayerNorm}(\mathrm{h}))$$

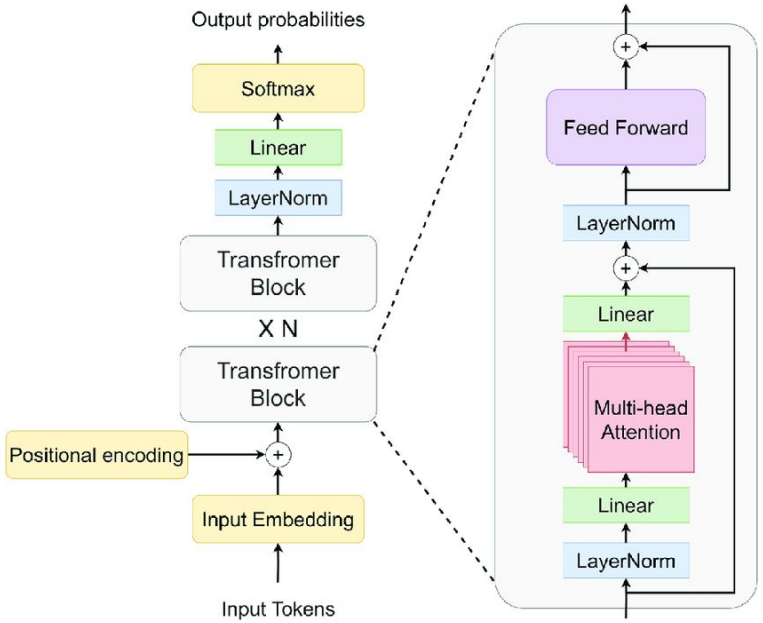


Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Autoregressive Transformers

- Standard transformer with a causal mask applied





Computational Scaling

Assuming sequence length L tokens and dimension D, FLOPs are:

- Multi-Head Attention: $8D^2L + 4DL^2 = O(D^2L + DL^2)$
  - **Quadratic in both D and L!**
- MLP: $16D^2L = O(D^2L)$

Autoregressive Models – Lecture Outline

- Motivation
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- **Deeper Dive into Causal Masked Neural Models**
  - Convolutional
  - Attention
  - **Tokenization**
  - Caching
- Other things to be aware of
  - Decoder-only vs. Encoder-Decoder models
  - New incarnations of Recurrent Models
  - Alternative / Complementary ideas to tokenization

Tokens

- Autoregressive transformers are general-purpose, modality-agnostic models
- Can model any complex distribution / modality as long as you can tokenize (discretize) the data

Tokens for Text

- **Characters**: a=0, b=1, c=2, …
  - Small vocabulary
  - Large number of tokens (recall quadratic scaling for attention…)
- **Words:** car=0, apple=1, dog=2, …
  - Large vocabulary
  - Small number of tokens
  - What happens if we introduce a new word?

Byte-Pair Encoding (BPE)

Can we consider something in-between?

- Tokenize based on *groupings* of characters, prioritizing by frequency

function BYTE-PAIR ENCODING(strings C, number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters
for $i = 1$ **to** k **do** # merge tokens til k times
  $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C
  $t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating
  $V \leftarrow V + t_{NEW}$ # update the vocabulary
  Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus
return V

Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." *arXiv preprint arXiv:1508.07909* (2015).

Byte-Pair Encoding (BPE)

Sub-word tokenizers: (a=0, umbr=1, ella=2, …)

- Middle-ground between number of tokens, and codebook size
- 1-2 tokens per word
- Generalizes to novel combinations of characters

GPT-1

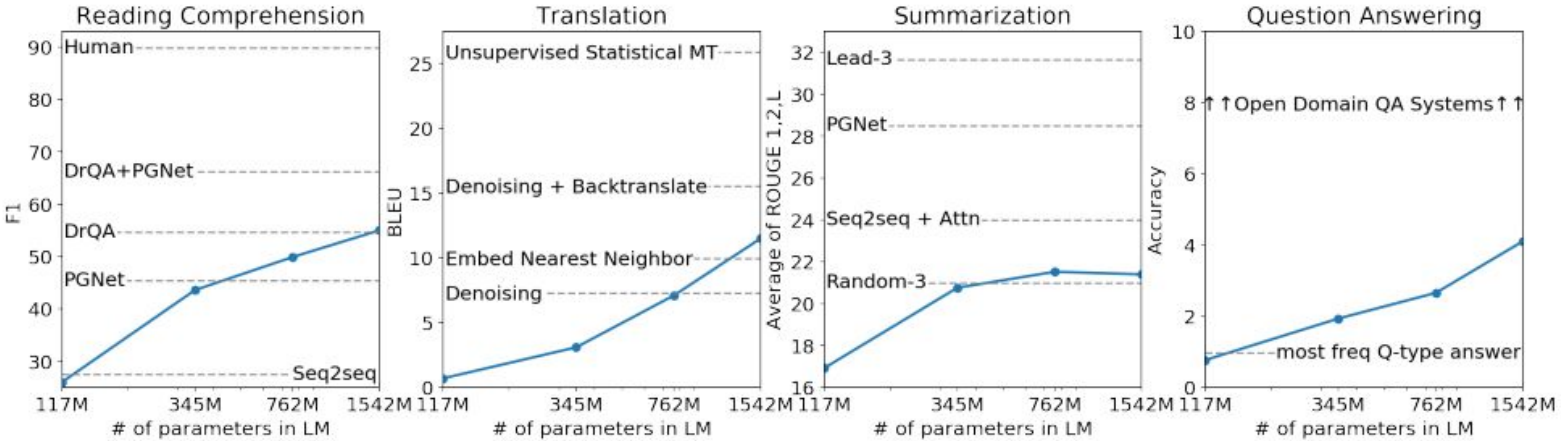
- Unsupervised pre-training followed by supervised fine-tuning
- 100M parameter transformer model
- Finetuning a model pretrained on general language outperforms models designed for each task

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).

GPT-2

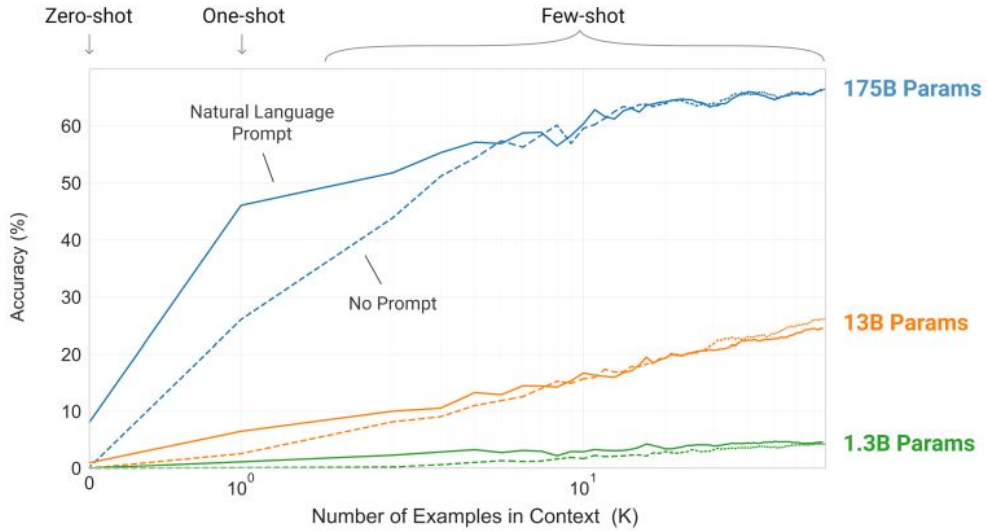
- Scaled data and model size to 1.5B parameters
- Improvements with scale even without supervised finetuning (zero-shot evaluation on downstream tasks)



Radford, Alec, et al. "Language models are unsupervised multitask learners." *OpenAI blog* 1.8 (2019): 9.

GPT-3

In-context learning abilities emerge as number of parameters grow

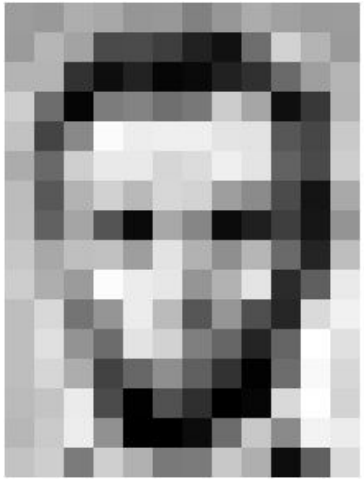


Brown, Tom, et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901.

Tokens for Images

Consider encoding RGB (H x W x 3) images as tokens

- Raw Pixels: each pixel for each color channel is generally stored as a single byte (0-255)



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

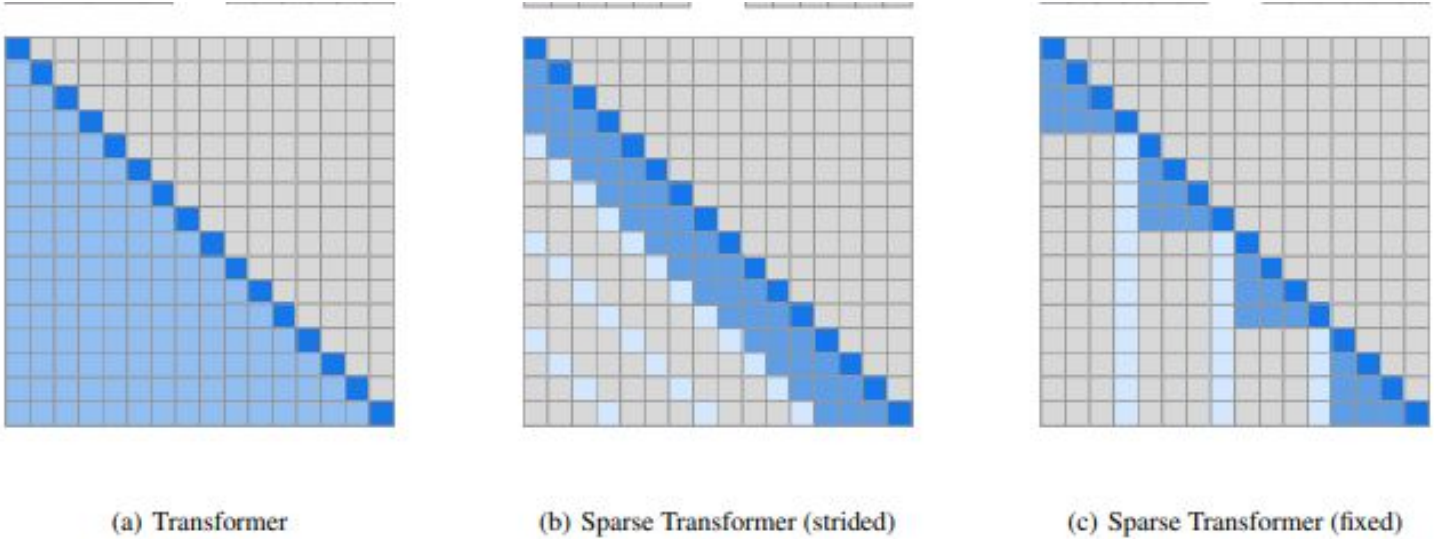
Raw Pixels as Tokens

- 64 x 64 x 3 = 12k tokens
- 256 x 256 x 3 = 196k tokens!

Quadratic scaling of attention makes this prohibitively expensive

Sparse Transformer

- Train a causal transformer, but with special hard-coded masking

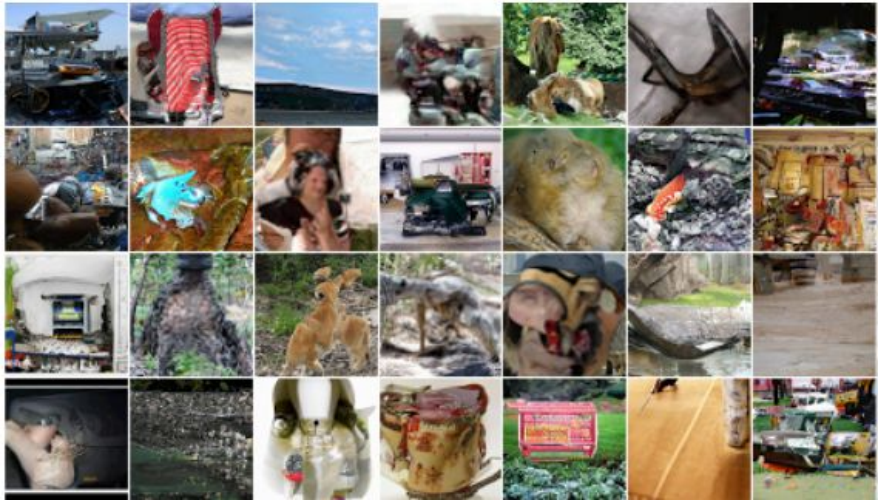


(a) Transformer (b) Sparse Transformer (strided) (c) Sparse Transformer (fixed)

Child, Rewon, et al. "Generating long sequences with sparse transformers." *arXiv preprint arXiv:1904.10509* (2019).
Parmar, Niki, et al. "Image transformer." *International conference on machine learning*. PMLR, 2018.

Sparse Transformer

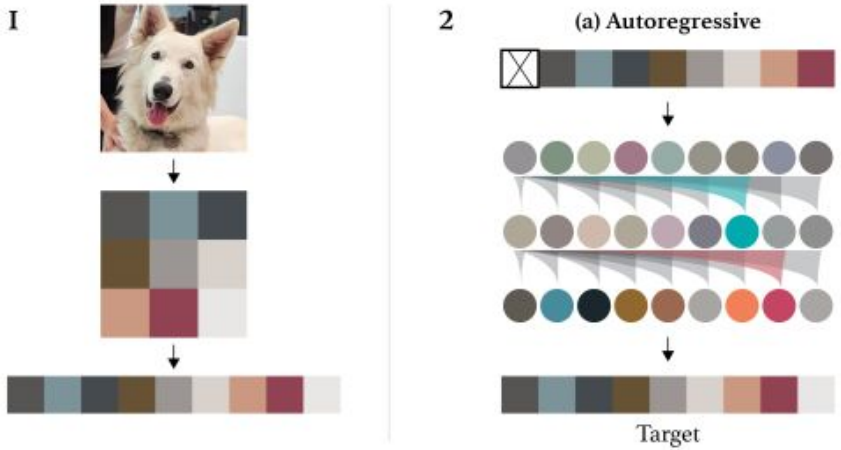
- SOTA results on perplexity, but still lacking in sample quality



Child, Rewon, et al. "Generating long sequences with sparse transformers." *arXiv preprint arXiv:1904.10509* (2019).

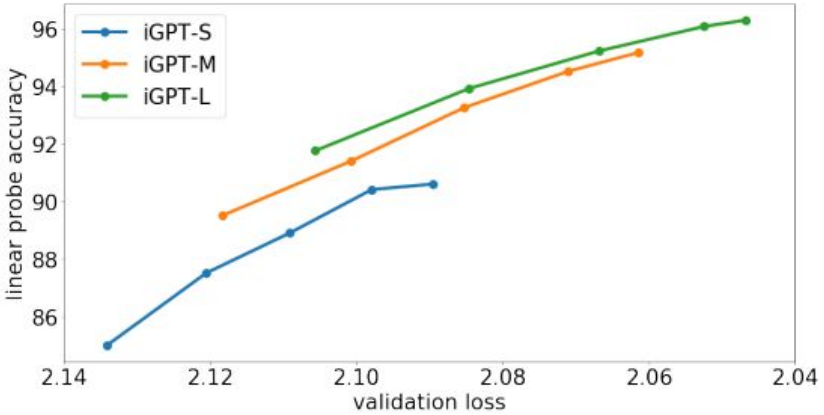
iGPT

- Created a custom 9-bit color palette by clustering RGB pixel values with k-means clustering
  - ~3x reduction in sequence length from the original 24-bit color palette



iGPT

Showed strong scaling results on representation learning with generation models compared to contemporary models



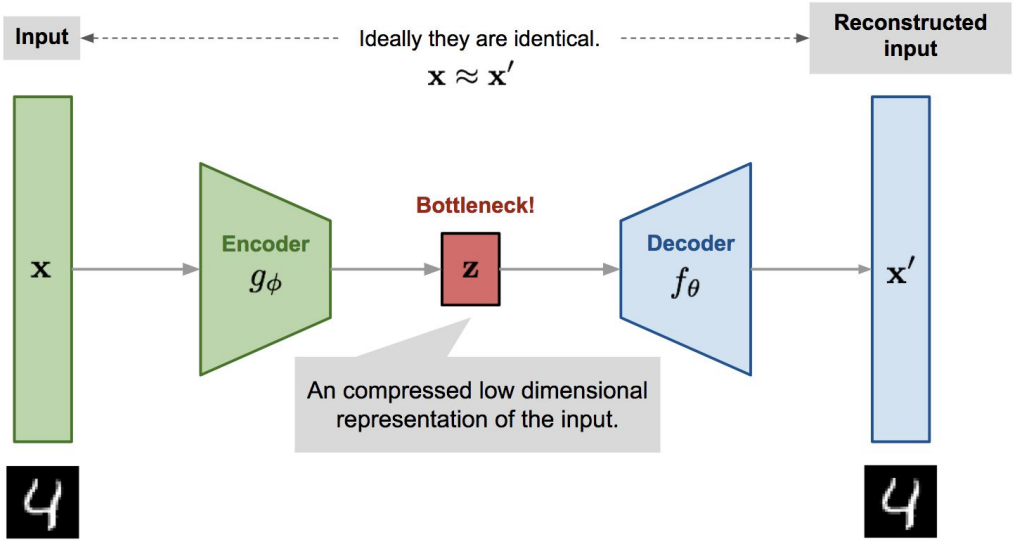
Model	Acc	Unsup Transfer	Sup Transfer
CIFAR-10			
ResNet-152	94		√
SimCLR	95.3	√	
iGPT-L	96.3	√	
CIFAR-100			
ResNet-152	78.0		√
SimCLR	80.2	√	
iGPT-L	82.8	√	
STL-10			
AMDIM-L	94.2	√	
iGPT-L	95.5	√	

Tokens for Images

- But iGPT still needed to train on image as small as 32 x 32
- Can we get better tokens for images?

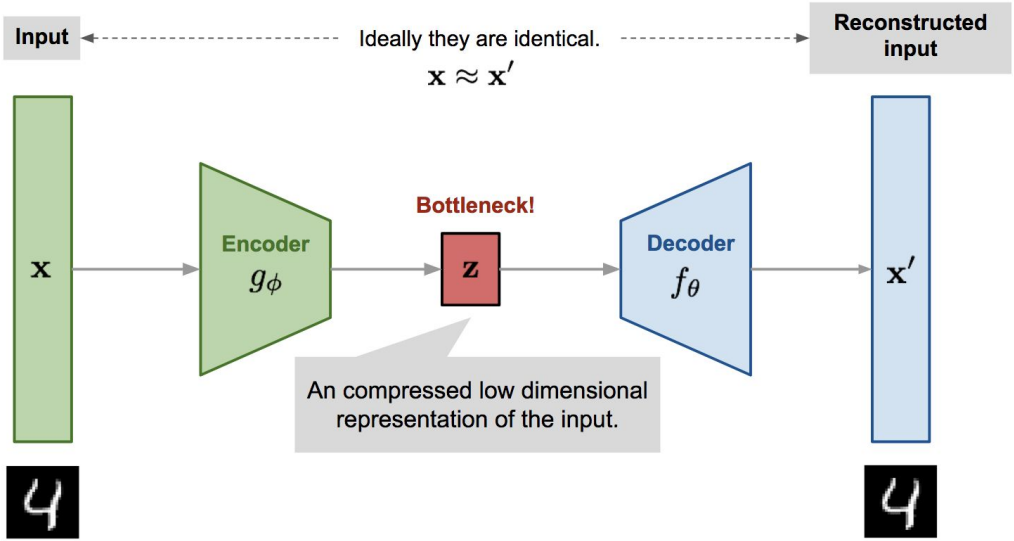
Discrete Autoencoders

Learn a discrete autoencoder!



Discrete Autoencoders

Learn a discrete autoencoder!



Discrete Autoencoders

Discretization methods:

- Gumbel-Softmax / Concrete Distribution: GS, CD
- Vector-Quantization (VQ): VQ-VAE
- Finite Scalar Quantization (FSQ) / Lookup-Free Quantization (LFQ): FSQ, LFQ

Discrete Autoencoders

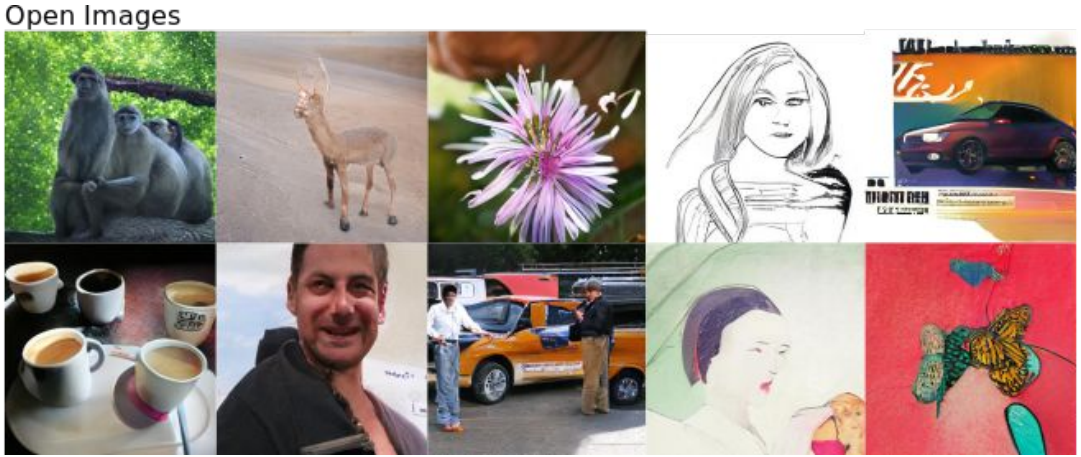
This method of tokenization is lossy

- 256 x 256 x 3 -> 16 x 16
  - 384x compression in bytes from raw pixels
  - 768x reduction in sequence length

VQGAN Transformer

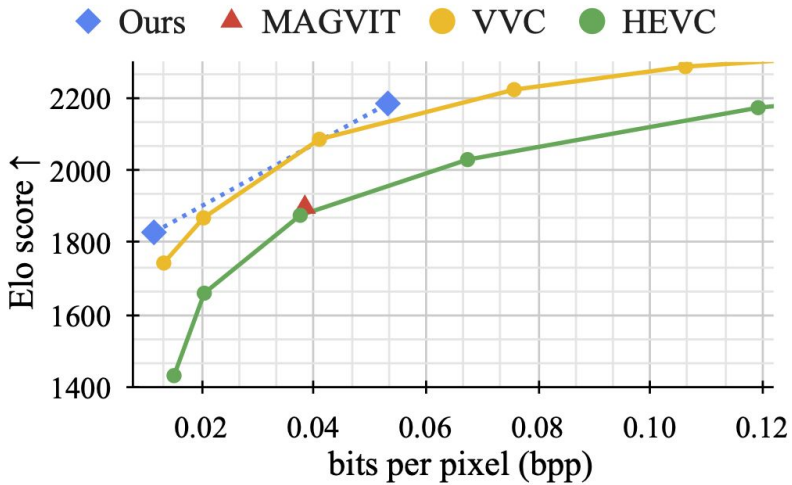
Train an autoregressive transformer to model discrete encodings

- Much more tractable than from pixels. Samples below



Esser, Patrick, Robin Rombach, and Bjorn Ommer. "Taming transformers for high-resolution image synthesis." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.

Discrete Encodings for Video

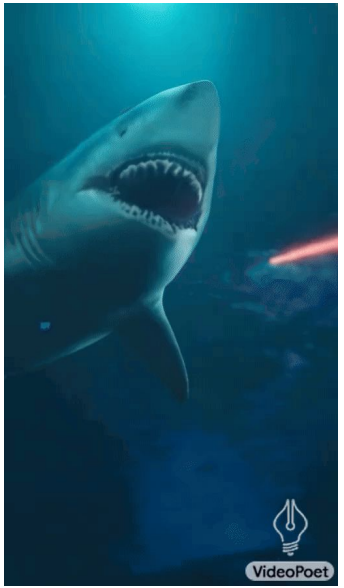


VideoPoet

The orient express driving through a fantasy landscape, animated oil on canvas

A golden retriever wearing VR goggles and eating pizza in Paris.

A chicken lifting weights

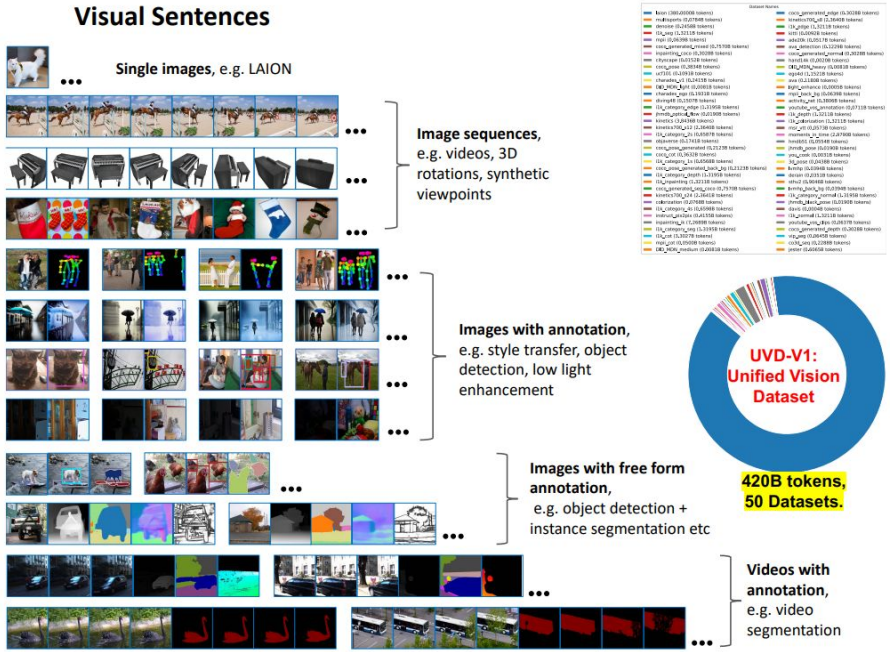


A shark with a laser beam coming out of its mouth

Kondratyuk, Dan, et al. "Videopoet: A large language model for zero-shot video generation." *arXiv preprint arXiv:2312.14125* (2023).

In-Context Learning for Vision

Train a large autoregressive transformer on sequential visual data



Bai, Yutong, et al. "Sequential modeling enables scalable learning for large vision models." *arXiv preprint arXiv:2312.00785* (2023).

In-Context Learning for Vision

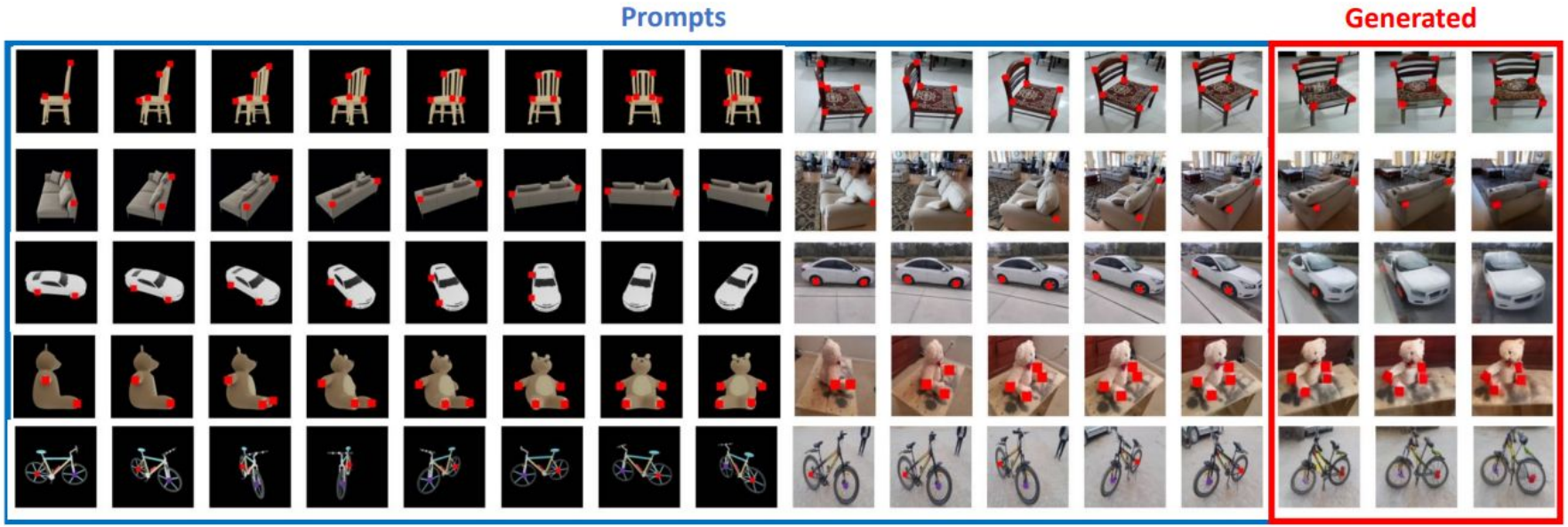
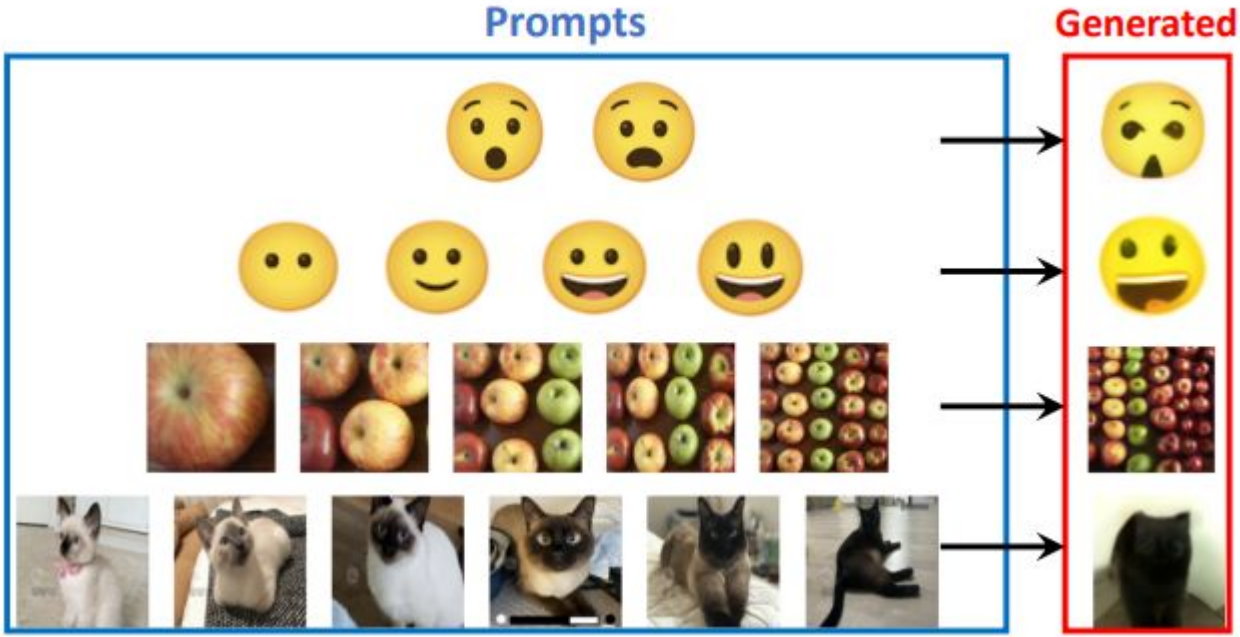
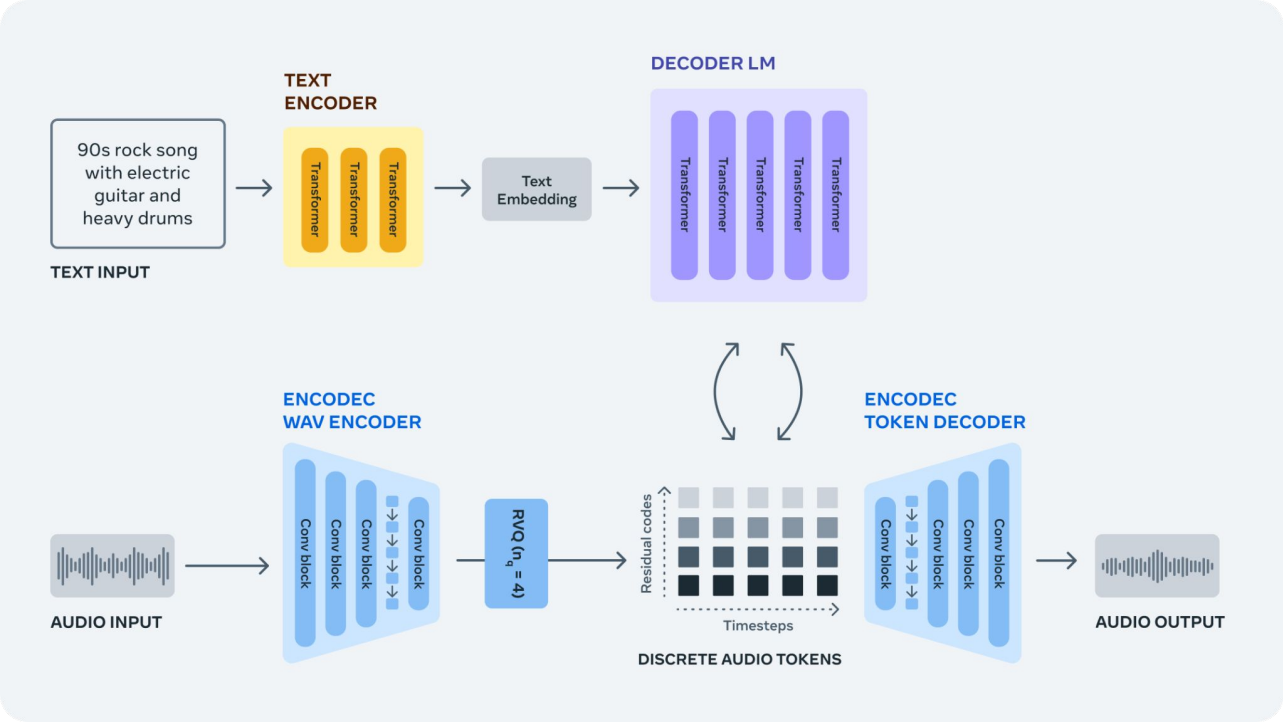


Figure 9. **Task Compositionality**. Examples of prompts that combine two different tasks – object rotation and keypoint tracking.

In-Context Learning for Vision

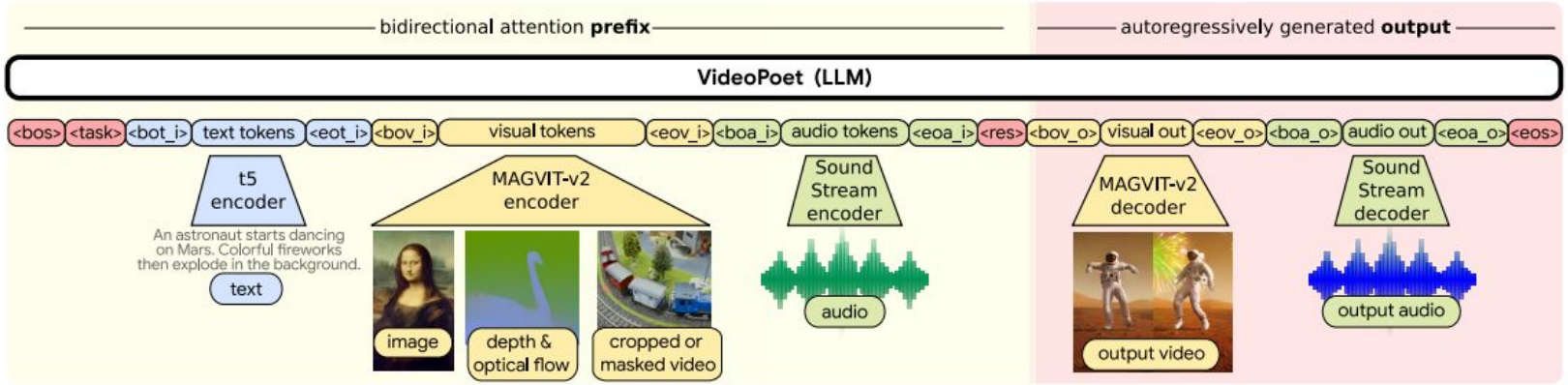


AudioCraft


TEXT ENCODER
DECODER LM
90s rock song with electric guitar and heavy drums
TEXT INPUT
Transformer
Transformer
Transformer
Text Embedding
Transformer
Transformer
Transformer
Transformer
Transformer
ENCODEC WAV ENCODER
ENCODEC TOKEN DECODER
AUDIO INPUT
Conv block
Conv block
Conv block
Conv block
RVQ (n_q = 4)
Residual codes
Timesteps
DISCRETE AUDIO TOKENS
Conv block
Conv block
Conv block
Conv block
AUDIO OUTPUT

Multi-Modal Models

- The general nature of transformers allows you to easily combine different modalities



Kondratyuk, Dan, et al. "Videopoet: A large language model for zero-shot video generation." *arXiv preprint arXiv:2312.14125* (2023).

Autoregressive Models – Lecture Outline

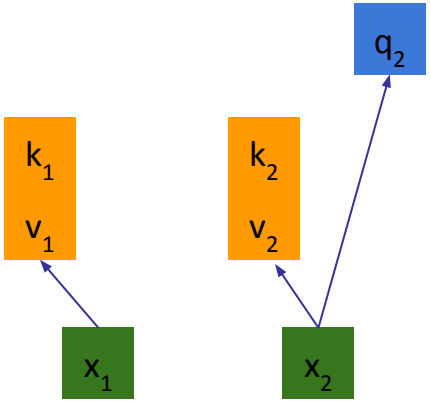
- Motivation
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- **Deeper Dive into Causal Masked Neural Models**
  - Convolutional
  - Attention
  - Tokenization
  - **Caching**
- Other things to be aware of
  - Decoder-only vs. Encoder-Decoder models
  - New incarnations of Recurrent Models
  - Alternative / Complementary ideas to tokenization

Caching / Efficient Sampling

Naive Sampling

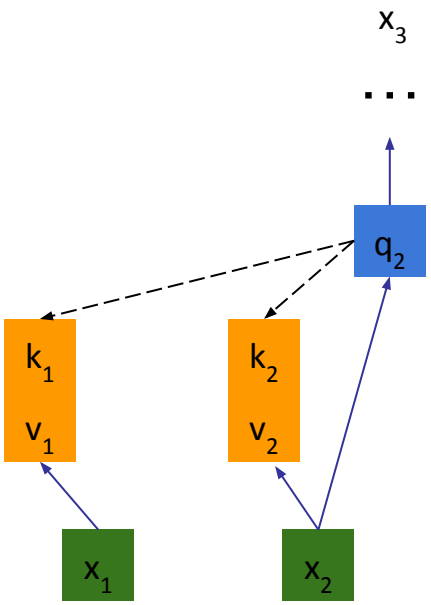
Caching / Efficient Sampling

Naive Sampling



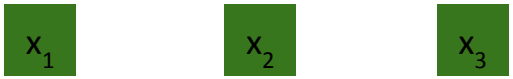
Caching / Efficient Sampling

Naive Sampling



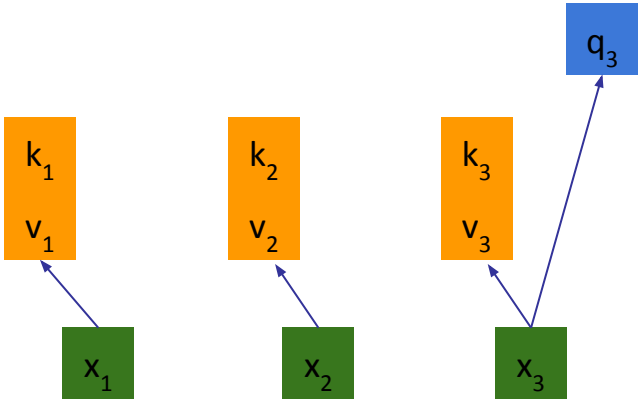
Caching / Efficient Sampling

Naive Sampling



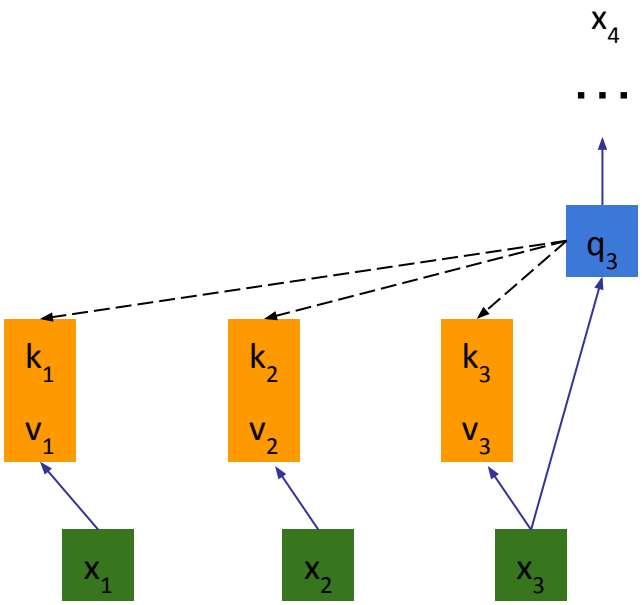
Caching / Efficient Sampling

Naive Sampling



Caching / Efficient Sampling

Naive Sampling

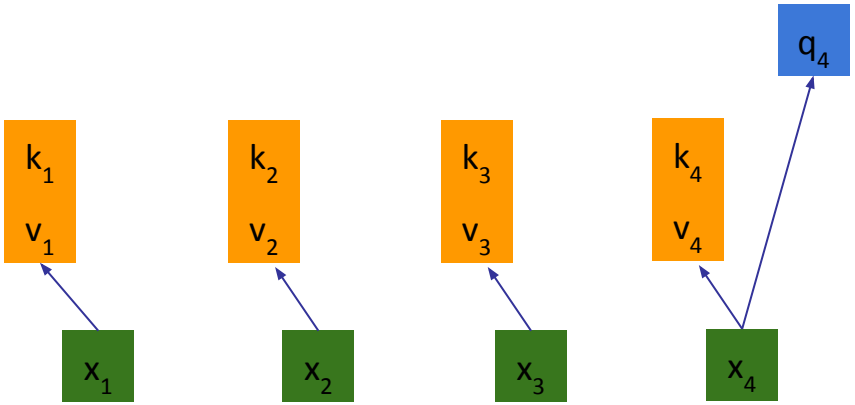


Caching / Efficient Sampling

Naive Sampling

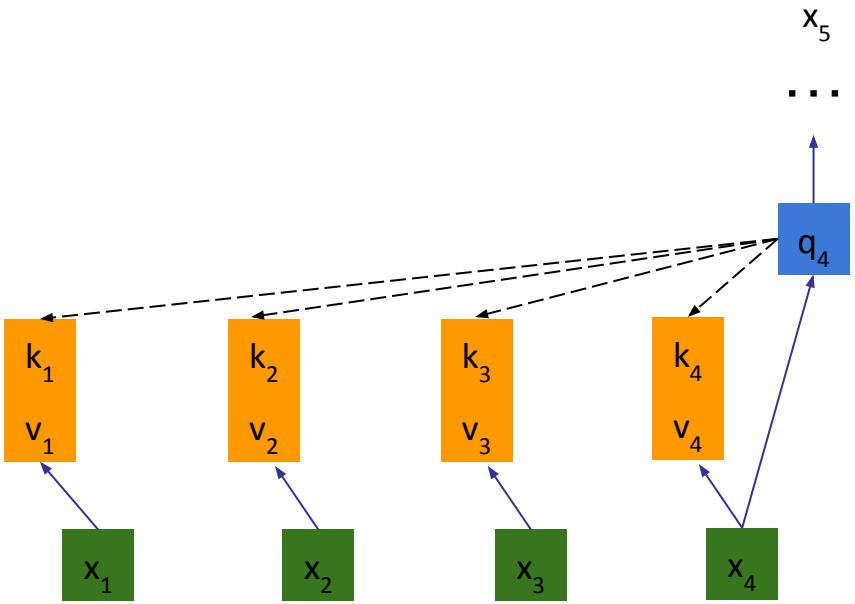
Caching / Efficient Sampling

Naive Sampling



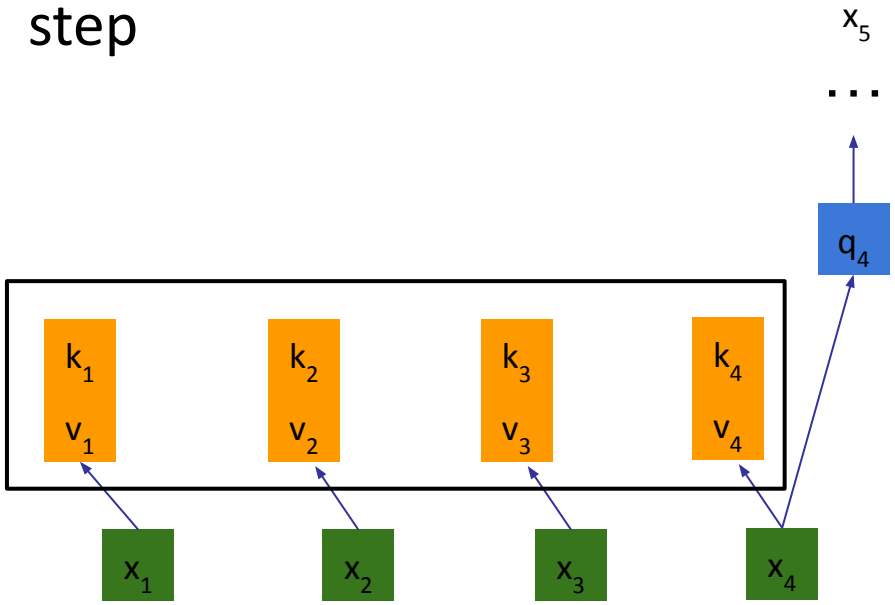
Caching / Efficient Sampling

Naive Sampling



Caching / Efficient Sampling

K / V for all timesteps need to be recomputed for every sampling step

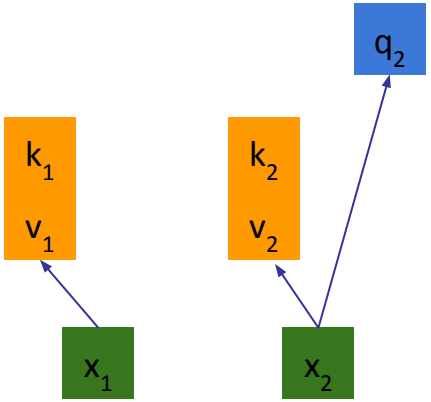


Caching / Efficient Sampling

Key Idea: Cache the K / V for all attention layers each sampling step

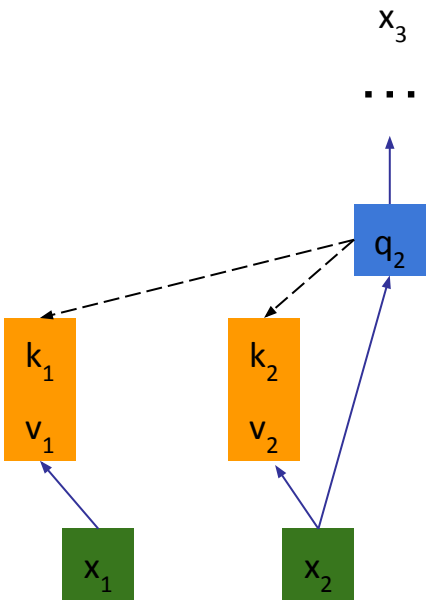
Caching / Efficient Sampling

Key Idea: Cache the K / V for all attention layers each sampling step



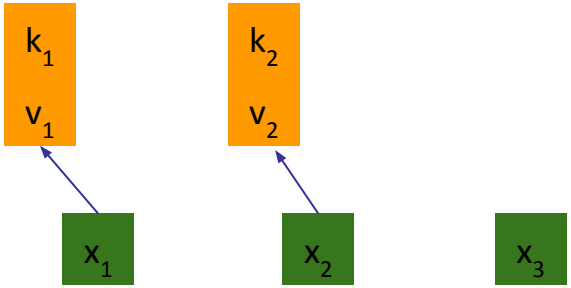
Caching / Efficient Sampling

Key Idea: Cache the K / V for all attention layers each sampling step



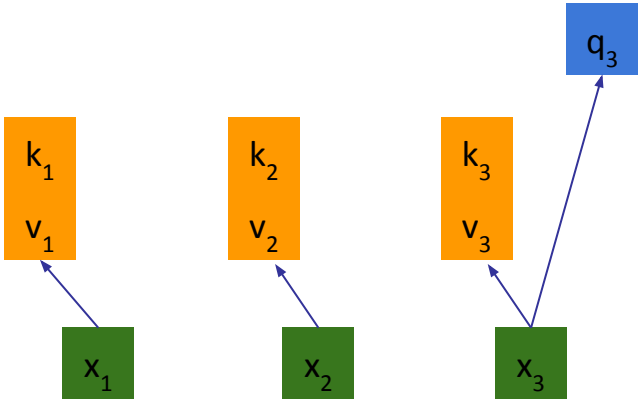
Caching / Efficient Sampling

Key Idea: Cache the K / V for all attention layers each sampling step



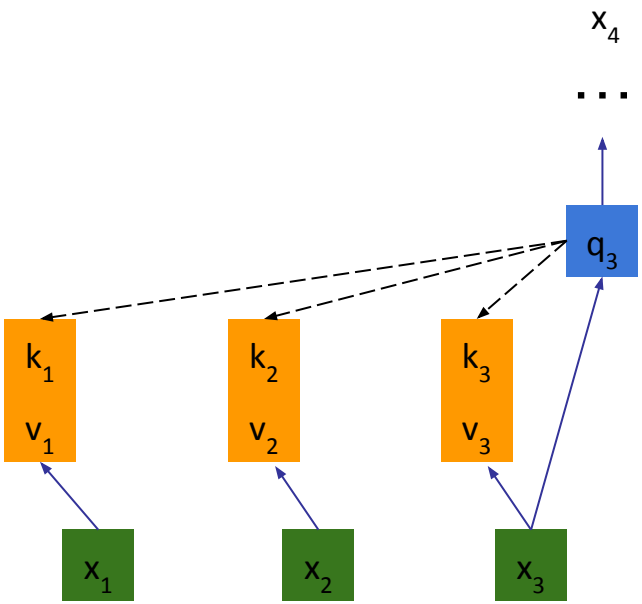
Caching / Efficient Sampling

Key Idea: Cache the K / V for all attention layers each sampling step



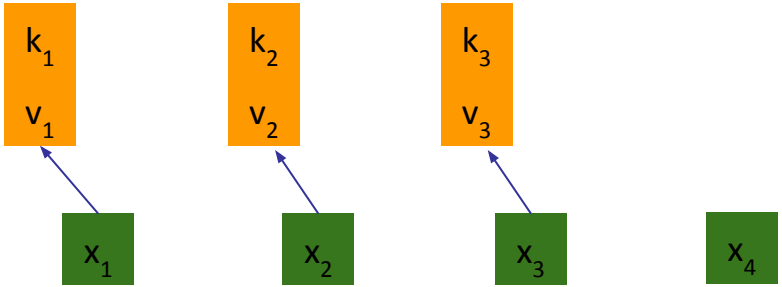
Caching / Efficient Sampling

Key Idea: Cache the K / V for all attention layers each sampling step



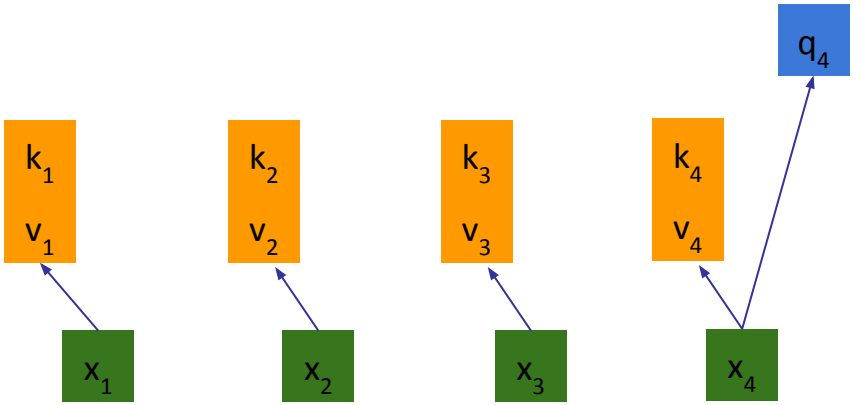
Caching / Efficient Sampling

Key Idea: Cache the K / V for all attention layers each sampling step



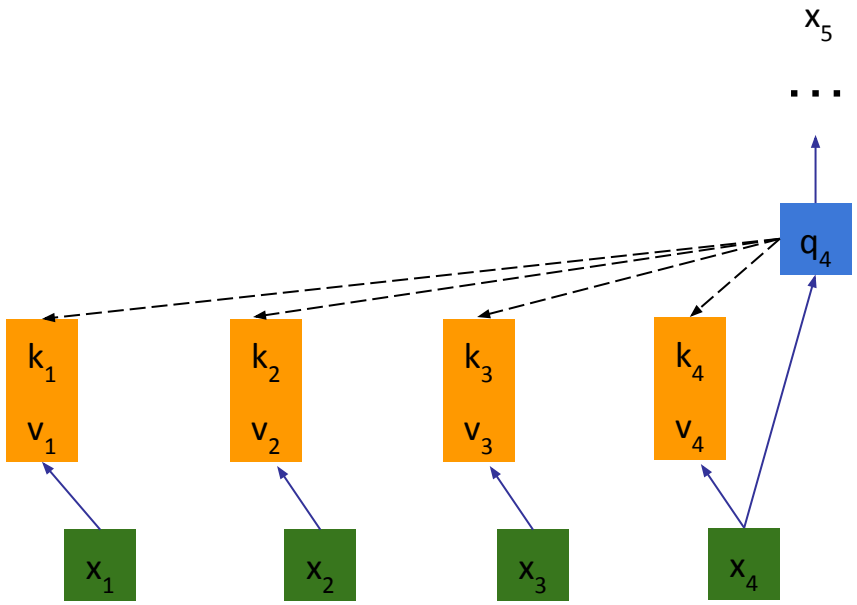
Caching / Efficient Sampling

Key Idea: Cache the K / V for all attention layers each sampling step



Caching / Efficient Sampling

Key Idea: Cache the K / V for all attention layers each sampling step



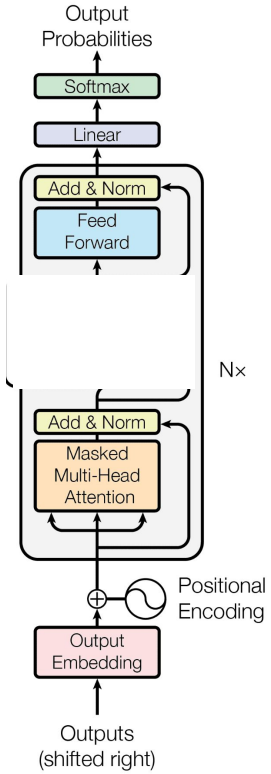
Per-Step

- Naive Sampling: $O(L^2)$
- Caching: $O(L)$

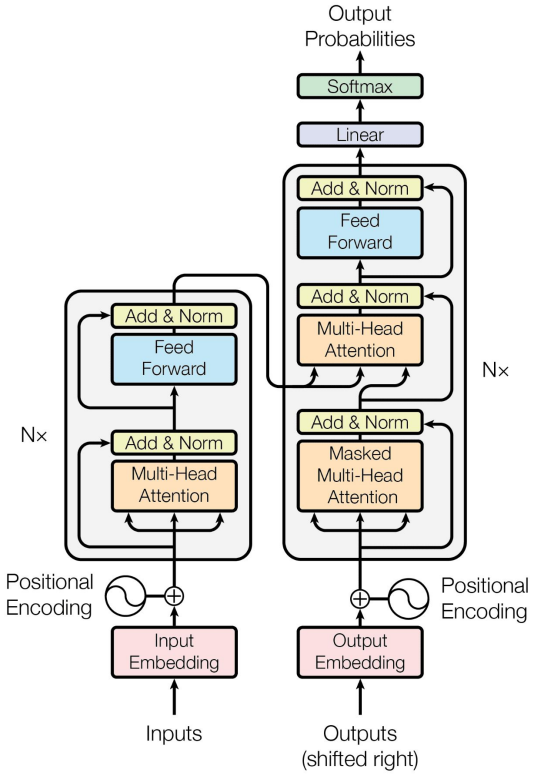
Autoregressive Models – Lecture Outline

- Motivation
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- Deeper Dive into Causal Masked Neural Models
  - Convolutional
  - Attention
  - Tokenization
  - Caching
- **Other things to be aware of**
  - **Decoder-only vs. Encoder-Decoder models**
  - New incarnations of Recurrent Models
  - Alternative / Complementary ideas to tokenization

Encoder-Decoder Models



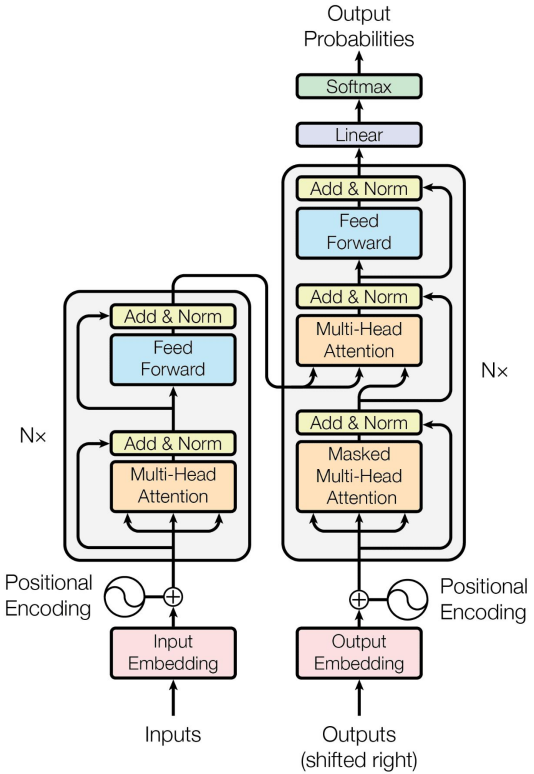
Encoder-Decoder Models



Encoder-Decoder Models

Good architecture choice for cases where you have a clear conditional distribution to model

- Machine Translation
- Text to image generation
- Image captioning
- Video captioning
- Summarization



Encoder-Decoder Models

- Examples:
  - T5 - text model
  - Parti - text to image generation
  - PaLI - image to text generation
- Both model types generally scale similarly, but encoder-decoder model seem to learn more useful (or easier to extract) representations
- Many existing text-image and video generation model condition on features from a T5 encoder. Have not seen as much success from decoder-only models

Parti

A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

Autoregressive Models – Lecture Outline

- Motivation
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- Deeper Dive into Causal Masked Neural Models
  - Convolutional
  - Attention
  - Tokenization
  - Caching
- **Other things to be aware of**
  - Decoder-only vs. Encoder-Decoder models
  - **New incarnations of Recurrent Models**
  - Alternative / Complementary ideas to tokenization

Modern Recurrent Models

- Transformers can efficiently model complex distributions
- But scaling to longer sequences can be expensive to do quadratic scaling
- RNNs are O(L) but sequentially unrolling the sequence is slow

How can we compute recurrences in parallel?

Linear State-Space Models

Consider a simple linear state-space model:

$$x_k = \overline{\boldsymbol{A}} x_{k-1} + \overline{\boldsymbol{B}} u_k$$
$$y_k = \overline{\boldsymbol{C}} x_k$$

- A, B, C are discretized version of learnable matrices
- Consider a single SSM layer as a replacement for an attention layer in a transformer

Linear State-Space Models

Parallelization:

$$x_k = \overline{\boldsymbol{A}} x_{k-1} + \overline{\boldsymbol{B}} u_k$$
$$y_k = \overline{\boldsymbol{C}} x_k$$

- Unroll the recurrence

$$x_0 = \overline{\boldsymbol{B}} u_0 \qquad x_1 = \overline{\boldsymbol{AB}} u_0 + \overline{\boldsymbol{B}} u_1 \qquad x_2 = \overline{\boldsymbol{A}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{AB}} u_1 + \overline{\boldsymbol{B}} u_2$$
$$y_0 = \overline{\boldsymbol{CB}} u_0 \qquad y_1 = \overline{\boldsymbol{CAB}} u_0 + \overline{\boldsymbol{CB}} u_1 \qquad y_2 = \overline{\boldsymbol{CA}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{CAB}} u_1 + \overline{\boldsymbol{CB}} u_2$$

- Reorder terms and rewrite as a convolution with kernel size L

$$y_k = \overline{\boldsymbol{CA}}^k \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{CA}}^{k-1} \overline{\boldsymbol{B}} u_1 + \cdots + \overline{\boldsymbol{CAB}} u_{k-1} + \overline{\boldsymbol{CB}} u_k$$
$$y = \overline{\boldsymbol{K}} * u$$

$$\overline{\boldsymbol{K}} \in \mathbb{R}^L = (\overline{\boldsymbol{CB}}, \overline{\boldsymbol{CAB}}, \ldots, \overline{\boldsymbol{CA}}^{L-1} \overline{\boldsymbol{B}})$$

Linear State-Space Models

Parallelization:

- Convolution method
- Parallel (associative) scan

Both parallelization methods are O(L * log(L)) complexity

Linear Attention

Consider the standard attention operation

$$V_i' = \frac{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right)}. \qquad \text{sim}\left(q, k\right) = \exp\left(\frac{q^T k}{\sqrt{D}}\right)$$

Can we try different similarity functions? We only need a non-negative kernel function (e.g. polynomial, or RBF kernel)

Linear Attention

Given a kernel with feature representation ϕ(x), write the attention as:

$$V_i' = \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)}, \longrightarrow V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)}.$$

$$\left(\phi(Q)\,\phi(K)^T\right) V = \phi(Q)\left(\phi(K)^T V\right)$$

Causal Linear Attention

Given a kernel with feature representation ϕ(x), write the attention as:

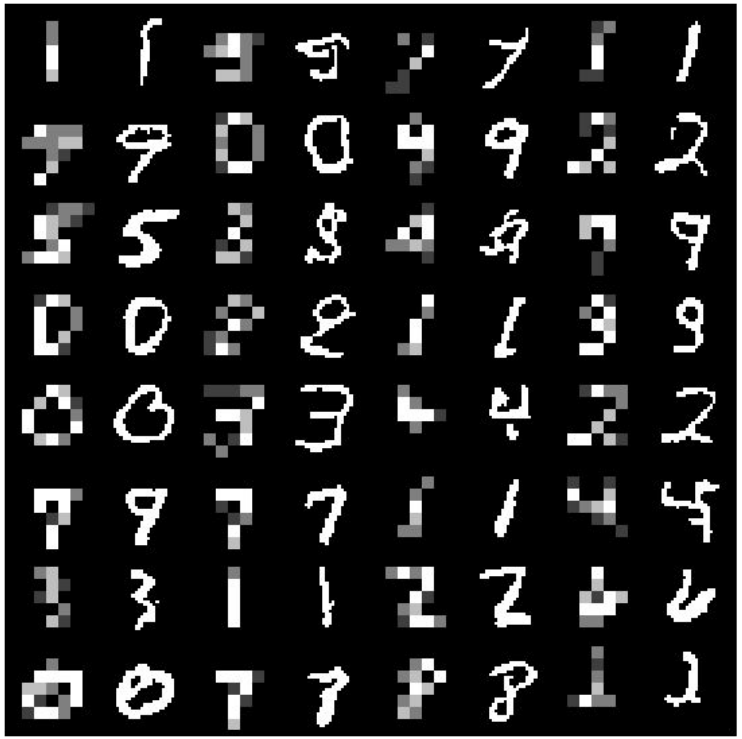
$$V_i' = \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)}, \longrightarrow V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{\cancel{N}\ i} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{\cancel{N}\ i} \phi(K_j)}.$$

Recurrent!

Autoregressive Models – Lecture Outline

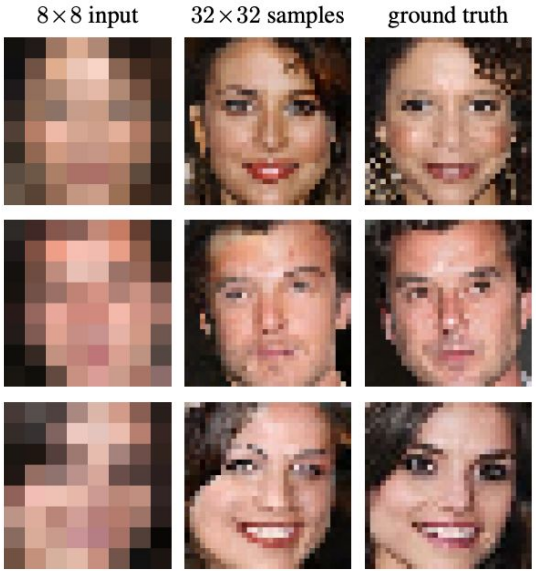
- Motivation
- 1-Dimensional Distributions
  - Simplest generative model: histogram
  - Parameterized distributions and maximum likelihood
- High-Dimensional Distributions
  - Chain rule
  - “Practical” Incarnations: Bayes’ Nets, MADE, Causal Masked Neural Models, RNNs
- Deeper Dive into Causal Masked Neural Models
  - Convolutional
  - Attention
  - Tokenization
  - Caching
- Other things to be aware of
  - Decoder-only vs. Encoder-Decoder models
  - New incarnations of Recurrent Models
  - **Alternative / Complementary ideas to tokenization**

Super-Resolution with PixelCNN



- A PixelCNN is conditioned on 7 x 7 subsampled MNIST images to generated the corresponding 28 x 28 image

Super-Resolution with PixelCNN



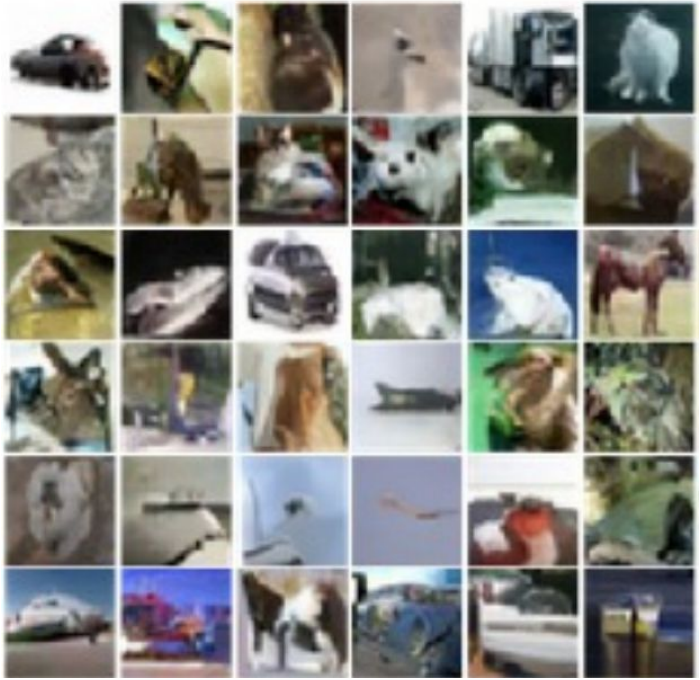
Hierarchical Autoregressive Models

[De Fauw, Jeffrey, Sander Dieleman, and Karen Simonyan. "Hierarchical autoregressive image models with auxiliary decoders." 2019]

Hierarchy: Grayscale PixelCNN

- PixelCNN1 generates grayscale
- PixelCNN2 conditionally generates colored images

Grayscale PixelCNN



Bibliography

char-rnn: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

MADE: Germain, Mathieu, et al. "Made: Masked autoencoder for distribution estimation." *International Conference on Machine Learning*. 2015.

WaveNet: Oord, Aaron van den, et al. "Wavenet: A generative model for raw audio." *arXiv preprint arXiv:1609.03499* (2016).

PixelCNN: Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." *arXiv preprint arXiv:1601.06759* (2016).

Gated PixelCNN: Van den Oord, Aaron, et al. "Conditional image generation with pixelcnn decoders." *Advances in Neural Information Processing Systems*. 2016.

PixelCNN++: Salimans, Tim, et al. "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications." *arXiv preprint arXiv:1701.05517* (2017)

Self-attention: Vaswani, Ashish, et al. "Attention is all you need." *Advances in Neural Information Processing Systems*. 2017.

PixelSNAIL: Chen, Xi, et al. "Pixelsnail: An improved autoregressive generative model." *arXiv preprint arXiv:1712.09763* (2017)

Fast PixelCNN++: Ramachandran, Prajit, et al. "Fast generation for convolutional autoregressive models." *arXiv preprint arXiv:1704.06001*(2017).

Multiscale PixelCNN: Reed, Scott, et al. "Parallel multiscale autoregressive density estimation." *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017.

Grayscale PixelCNN: Kolesnikov, Alexander, and Christoph H. Lampert. "PixelCNN models with auxiliary variables for natural image modeling." *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017.

Subscale Pixel Network: Menick, Jacob, and Nal Kalchbrenner. "Generating High Fidelity Images with Subscale Pixel Networks and Multidimensional Upscaling." *arXiv preprint arXiv:1812.01608*(2018)

Dirk Weissenborn, Oscar Tackstrom, Jakob Uszkoreit. “Scaling Autoregressive Video Models.” arXiv 1906.02634 (2019)

Sparse Attention: Rewon Child, Scott Gray, Alec Radford, Ilya Sutskever. “Generating Long Sequences with Sparse Transformers.” arXiv 1904.10509

Wilson Yan, Jonathan Ho, Pieter Abbeel. “Natural Image Manipulation for Autoregressive Models using Fisher Scores.” arXiv 1912.05015

PixelCNN Super Resolution: Dahl, Ryan, Mohammad Norouzi, and Jonathon Shlens. "Pixel recursive super resolution." *Proceedings of the IEEE International Conference on Computer Vision*. 2017.

Bibliography (part 2)

GPT-1: Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).

GPT-2: Radford, Alec, et al. "Language models are unsupervised multitask learners." *OpenAI blog* 1.8 (2019): 9.

GPT-3: Brown, Tom, et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901.

Transformer: Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

VQ-VAE: Van Den Oord, Aaron, and Oriol Vinyals. "Neural discrete representation learning." *Advances in neural information processing systems* 30 (2017).

VQ-GAN: Esser, Patrick, Robin Rombach, and Bjorn Ommer. "Taming transformers for high-resolution image synthesis." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.

MAGVIT-v2: Yu, Lijun, et al. "Language Model Beats Diffusion--Tokenizer is Key to Visual Generation." *arXiv preprint arXiv:2310.05737* (2023).

VideoPoet: Kondratyuk, Dan, et al. "Videopoet: A large language model for zero-shot video generation." *arXiv preprint arXiv:2312.14125* (2023).

S4: Gu, Albert, Karan Goel, and Christopher Ré. "Efficiently modeling long sequences with structured state spaces." *arXiv preprint arXiv:2111.00396* (2021).

Linear Attention: Katharopoulos, Angelos, et al. "Transformers are rnns: Fast autoregressive transformers with linear attention." *International conference on machine learning*. PMLR, 2020.

FSQ: Mentzer, Fabian, et al. "Finite scalar quantization: Vq-vae made simple." *arXiv preprint arXiv:2309.15505* (2023).

Gumbel-Softmax: Jang, Eric, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax." *arXiv preprint arXiv:1611.01144* (2016).

Concrete Distribution: Maddison, Chris J., Andriy Mnih, and Yee Whye Teh. "The concrete distribution: A continuous relaxation of discrete random variables." *arXiv preprint arXiv:1611.00712* (2016).

Image Transformer: Parmar, Niki, et al. "Image transformer." *International conference on machine learning*. PMLR, 2018.

Sparse Transformer: Child, Rewon, et al. "Generating long sequences with sparse transformers." *arXiv preprint arXiv:1904.10509* (2019).

Sequential Modeling enables scalable learning for large vision models: Bai, Yutong, et al. "Sequential modeling enables scalable learning for large vision models." *arXiv preprint arXiv:2312.00785* (2023).

APA

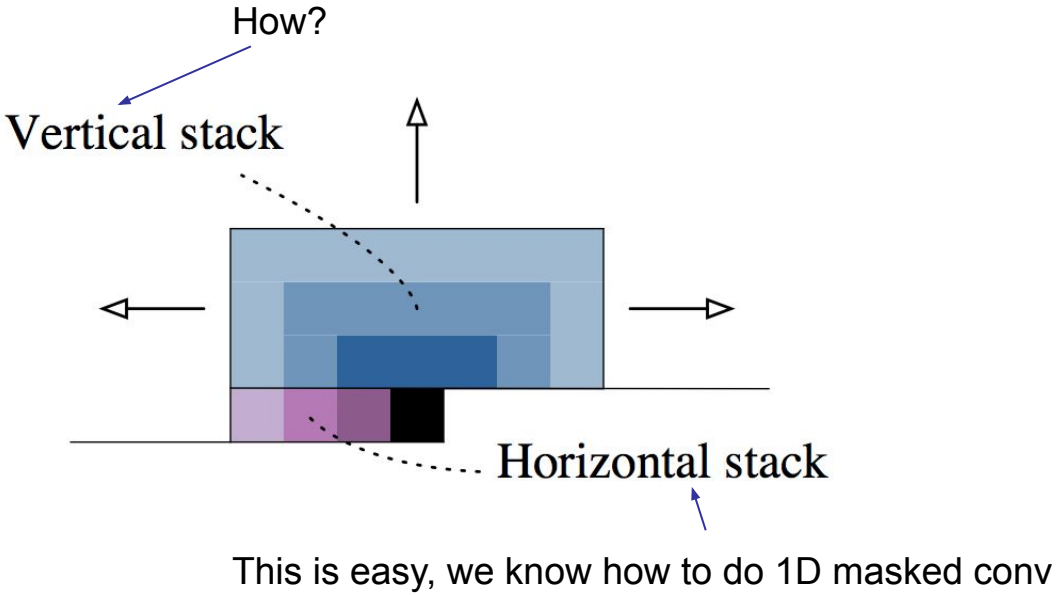
Colab

- Demo 1: Fitting a Simple 1D Discrete Distribution
  - Initializing Hyperparameters and Visualizing Dataset
  - Training Code
  - Model 1: Histogram
  - Model 2: Discretized Mixture of Logistics
  - Comparing with Test Data (with missing data)
- Demo 2: A Simple Autoregressive Model on 2D Data
- Demo 3 Different Autoregressive Model Architectures
  - RNN
  - RNN with Pixel Location Appended as Features
  - MADE
  - WaveNet
  - WaveNet with Pixel Location Appended as Features
  - PixelCNN
- Demo 4: Comparing Receptive Fields of PixelCNN and GatedPixelCNN (Blindspot)
  - PixelCNN Blindspot
  - PixelCNN with Horizontal and Vertical Stacked Convolutions (No Blindspot)
- Demo 5: Self-Attention Autoregressive Model
  - Multi-Head Attention Module (Key Component in Self-Attention)
- Demo 6: Different Autoregressive Orderings
  - Order 1: Random Permutation
  - Order 2: Even Indices Then Odd Indices
  - Order 3: Rows (Raster Scan)
  - Order 4: Columns
  - Order 4: Top to Middle then Bottom to Middle
- Demo 7: Conditional Autoregressive Models
  - Class-Conditional PixelCNN
  - Image Super-Resolution with a PixelCNN
- Demo 8: Hierarchy (Grayscale PixelCNNs)
- Demo 9: Fast Sampling (Parallel PixelCNNs)

Appendix A: Not covered in lecture, included fyi

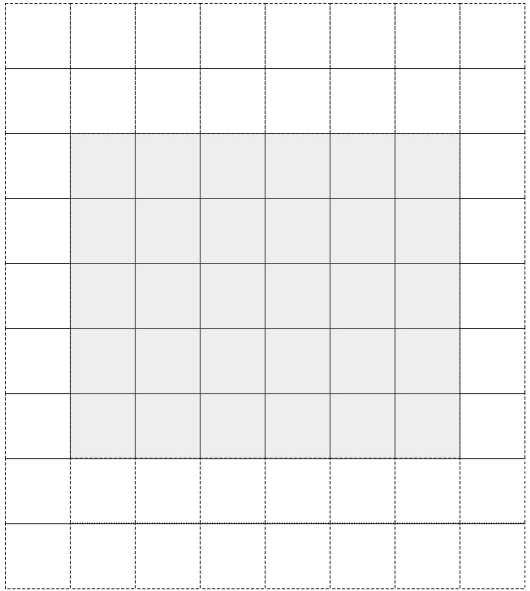
Gated PixelCNN

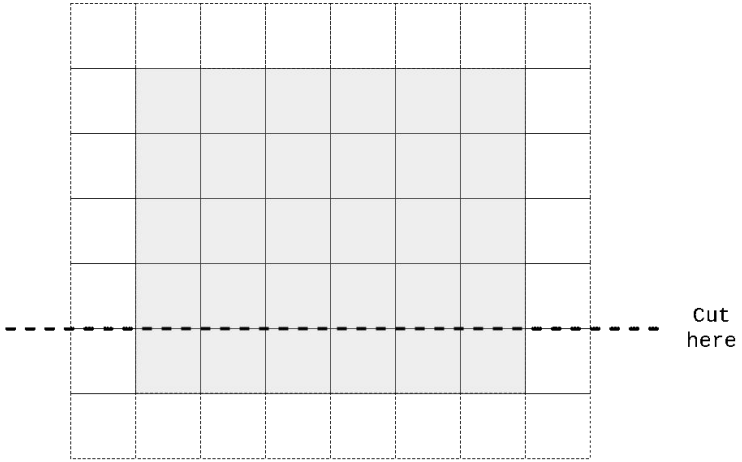
- Gated PixelCNN (2016) introduced a fix by combining two streams of convolutions



Gated PixelCNN

- Vertical stack: through padding, activations at *i*th row only depend on input before *i*th row

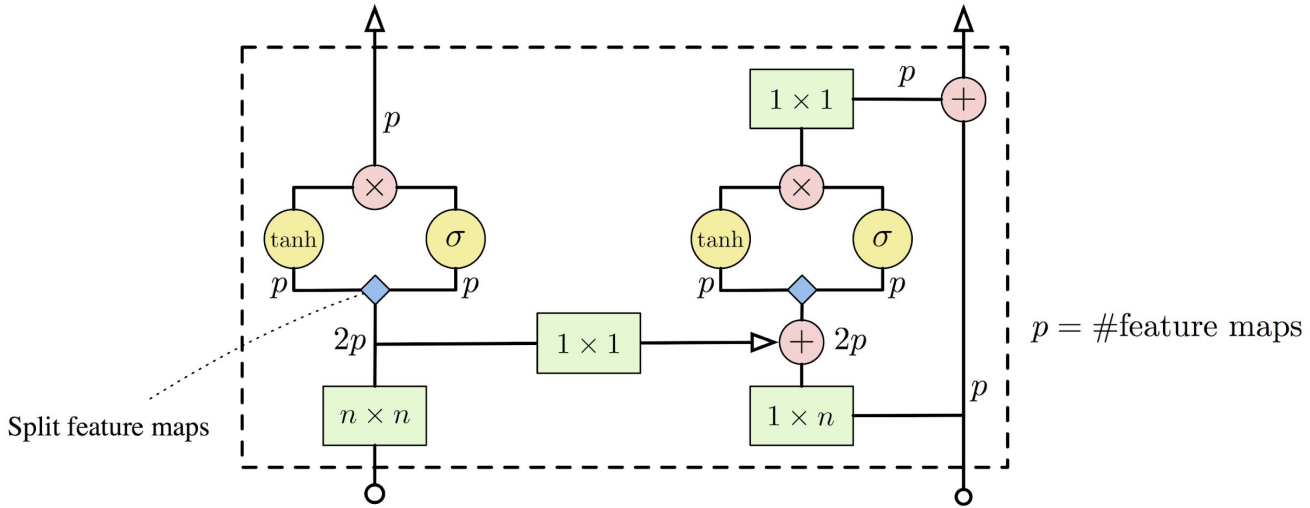




Gated PixelCNN

- Improved ConvNet architecture: Gated ResNet Block

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



Gated PixelCNN

- Better receptive field + more expressive architecture = better performance

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)

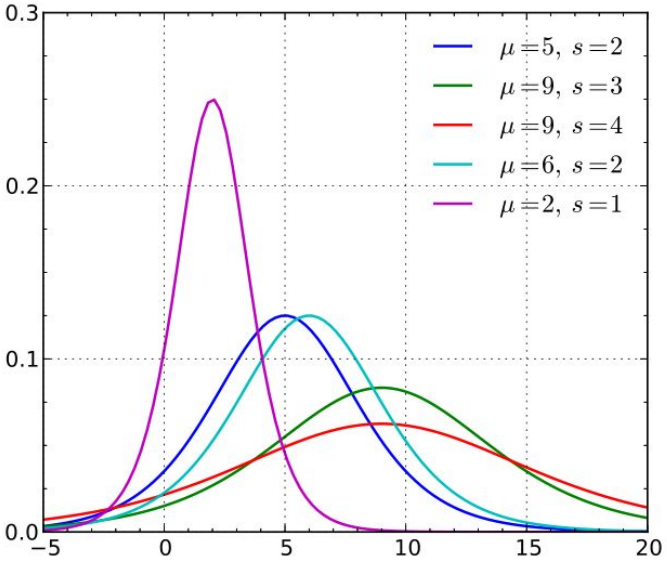
PixelCNN++

- Moving away from softmax: we know nearby pixel values are likely to co-occur!

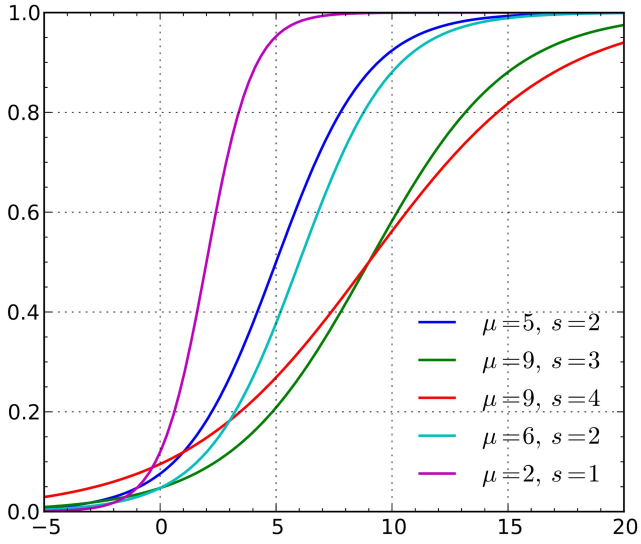
$$\nu \sim \sum_{i=1}^{K} \pi_i \text{logistic}(\mu_i, s_i)$$

$$P(x|\pi, \mu, s) = \sum_{i=1}^{K} \pi_i \left[\sigma((x + 0.5 - \mu_i)/s_i) - \sigma((x - 0.5 - \mu_i)/s_i)\right],$$

Recap: Logistic distribution

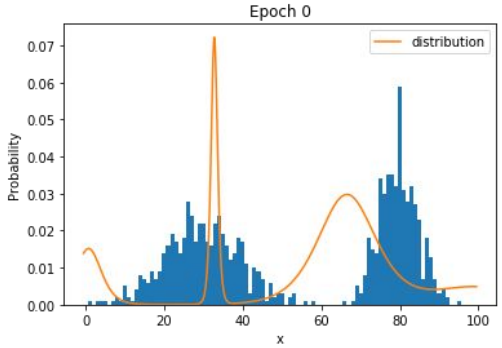


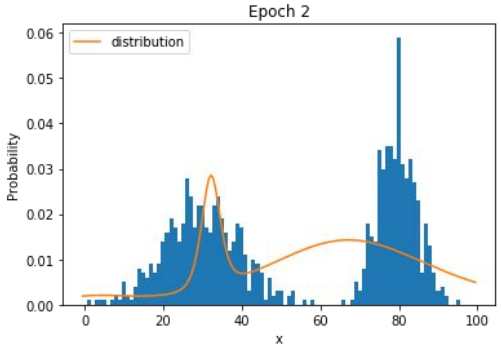
pdf

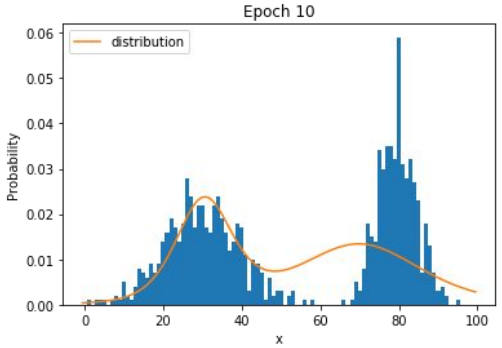


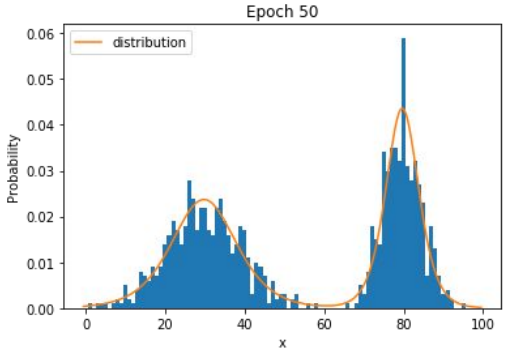
cdf
= sigmoid((x - mu) / scale)

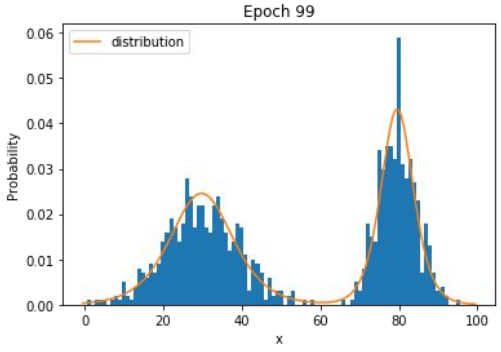
Ex. Training Mixture of Logistics

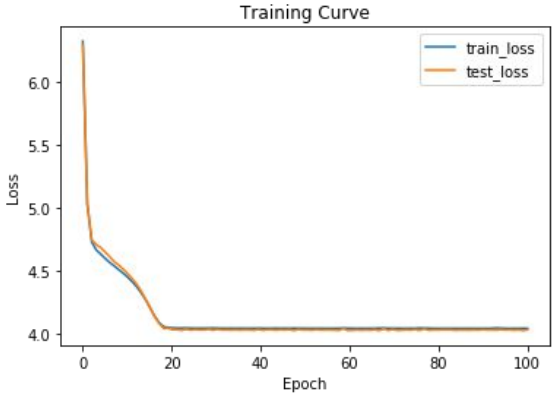






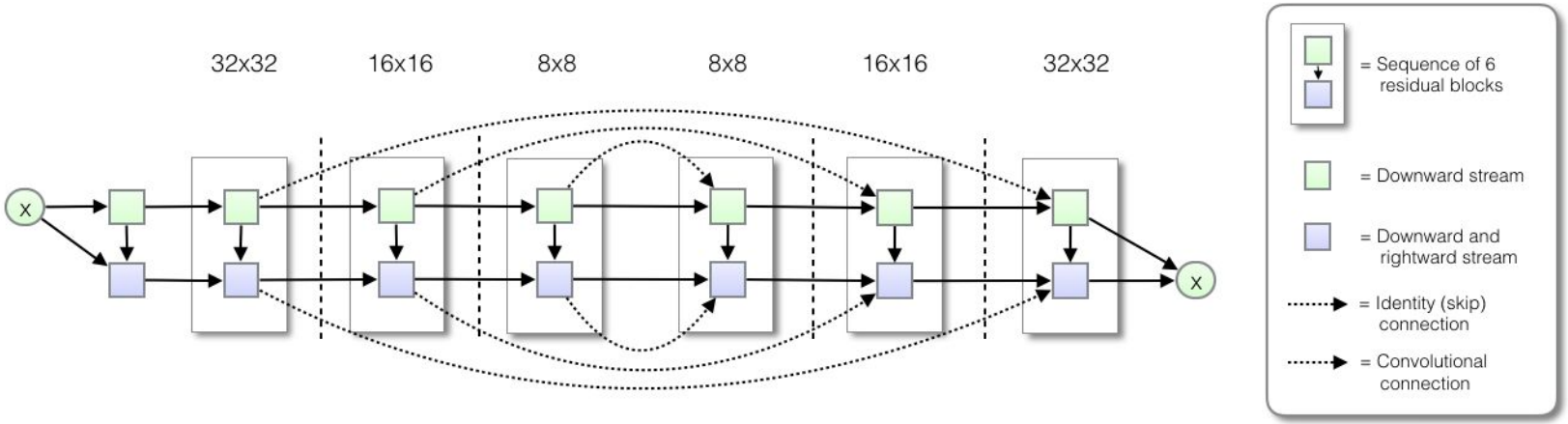






PixelCNN++

- Capture long dependencies efficiently by downsampling



PixelCNN++

Model	Bits per sub-pixel
Deep Diffusion (Sohl-Dickstein et al., 2015)	5.40
NICE (Dinh et al., 2014)	4.48
DRAW (Gregor et al., 2015)	4.13
Deep GMMs (van den Oord & Dambre, 2015)	4.00
Conv DRAW (Gregor et al., 2016)	3.58
Real NVP (Dinh et al., 2016)	3.49
PixelCNN (van den Oord et al., 2016b)	3.14
VAE with IAF (Kingma et al., 2016)	3.11
Gated PixelCNN (van den Oord et al., 2016c)	3.03
PixelRNN (van den Oord et al., 2016b)	3.00
PixelCNN++	**2.92**

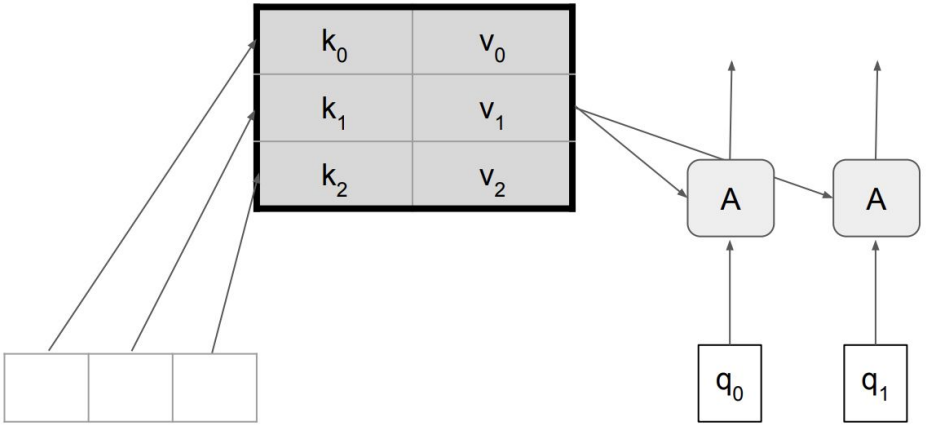
Masked Attention

- A recurring problem for convolution: limited receptive field -> hard to capture long-range dependencies
- (Self-)Attention: an alternative that has
  - unlimited receptive field!!
  - also O(1) parameter scaling w.r.t. data dimension
  - parallelized computation (versus RNN)

Attention

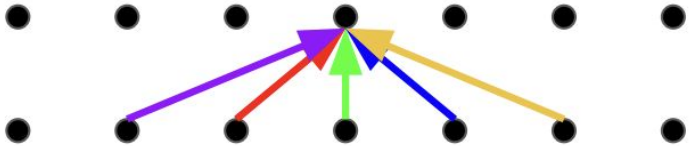
Dot-Product Attention

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

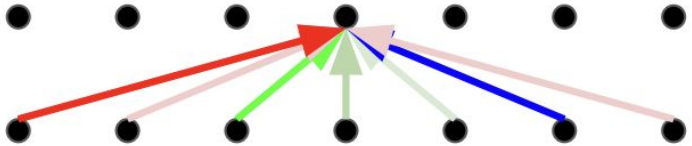


Self-attention when q_i also generated from x

Self-Attention



Convolution

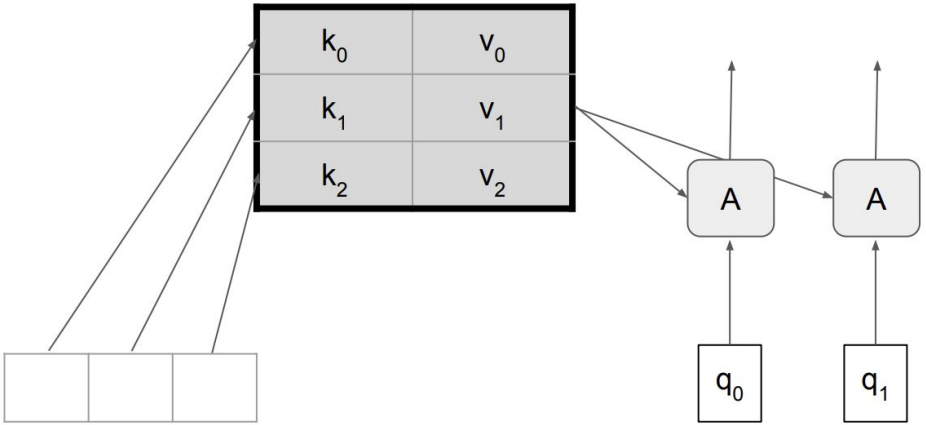


Self-attention

Masked Attention

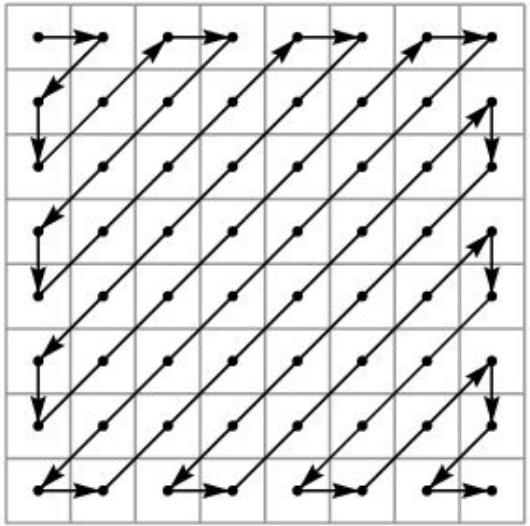
Dot-Product Attention

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i} \text{ - masked}(k_i, q) * 10^{10}}{\sum_j e^{q \cdot k_j} \text{ - masked}(k_i, q) * 10^{10}} v_i$$



Masked Attention

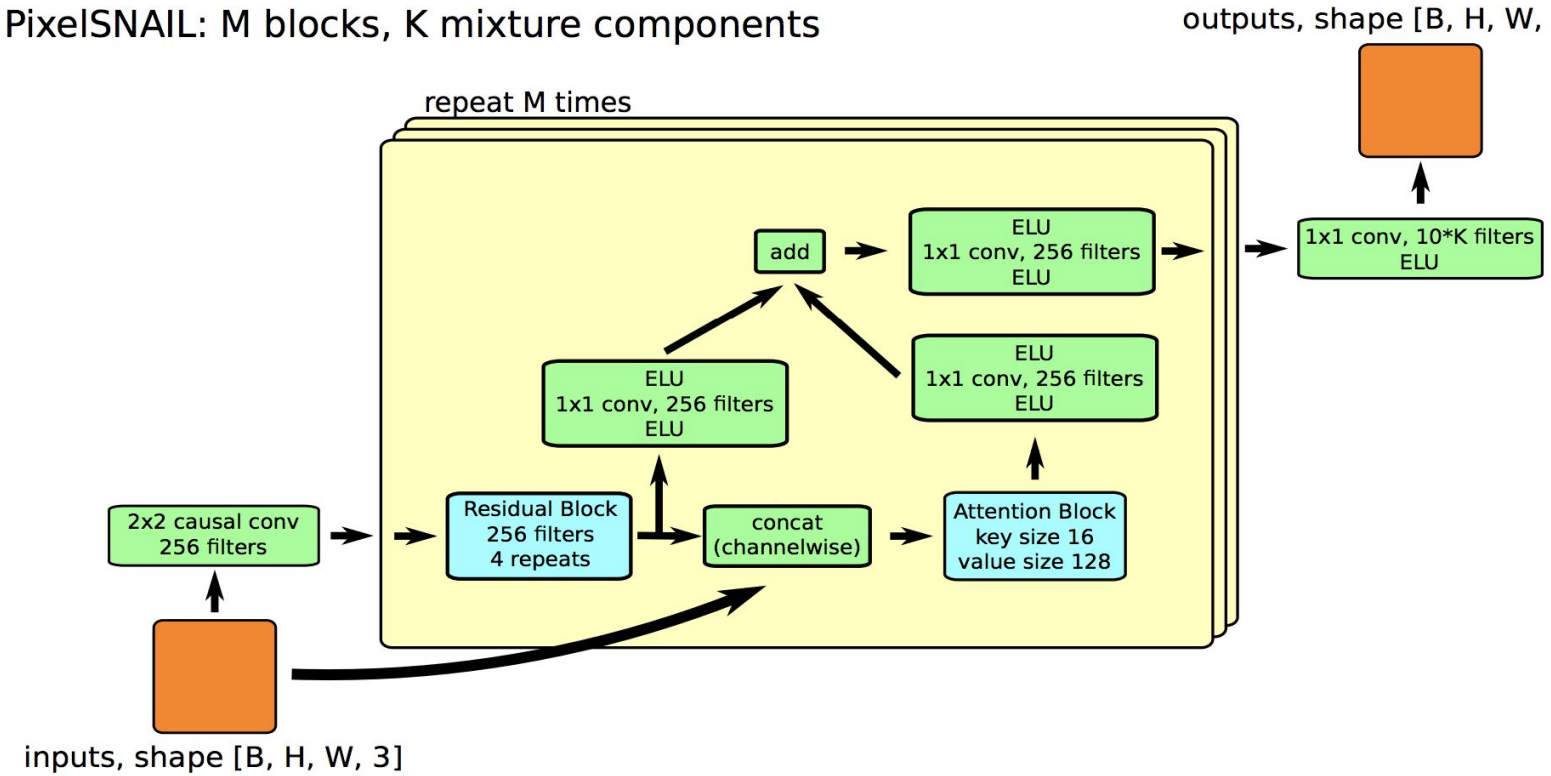
- Much more flexible than masked convolution. We can design any autoregressive ordering we want
- An example:



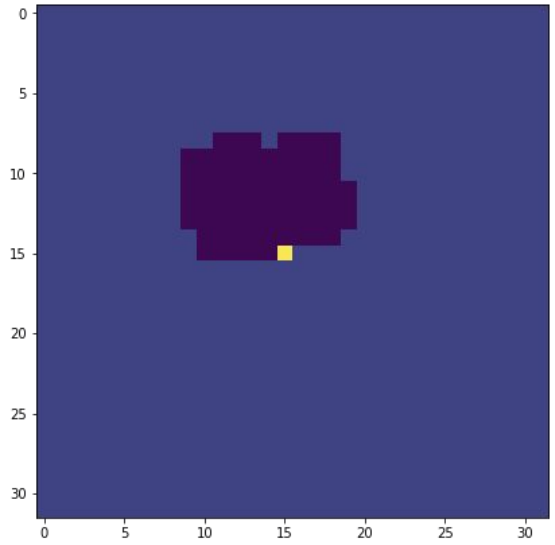
Zigzag ordering

- How to implement with masked conv?
- Trivial to do with masked attention!

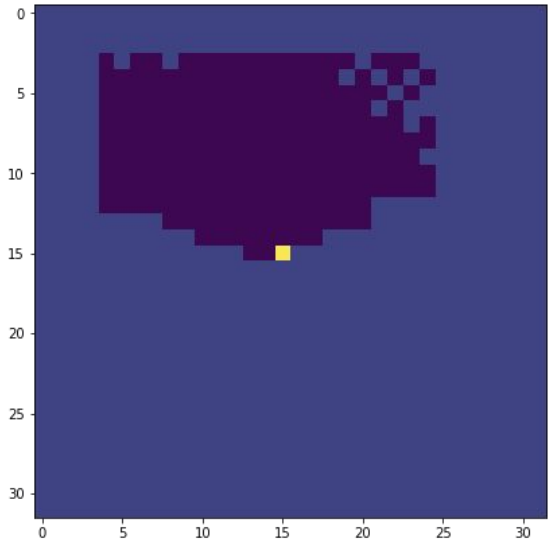
Masked Attention + Convolution



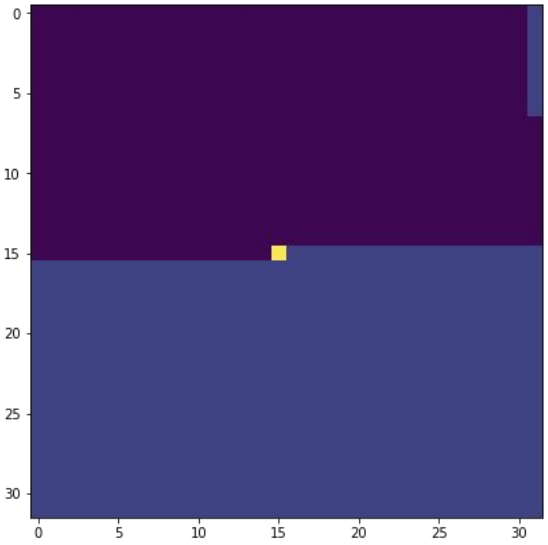
Masked Attention + Convolution



Gated PixelCNN

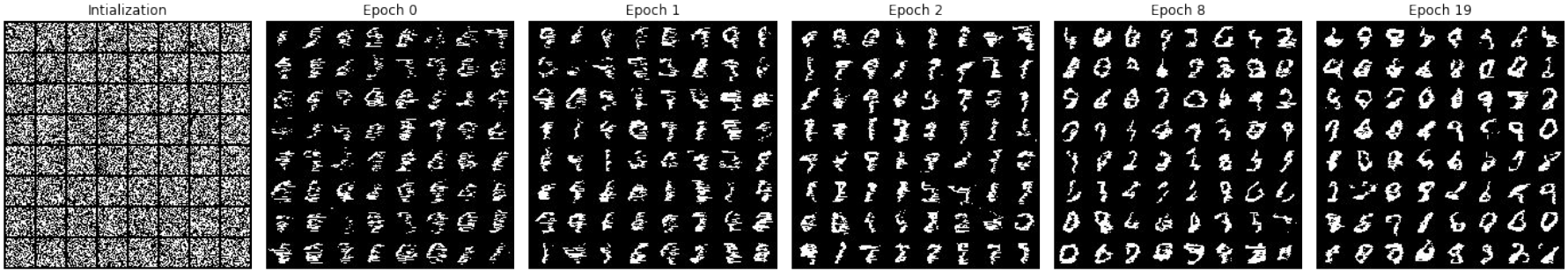


PixelCNN++



PixelSNAIL

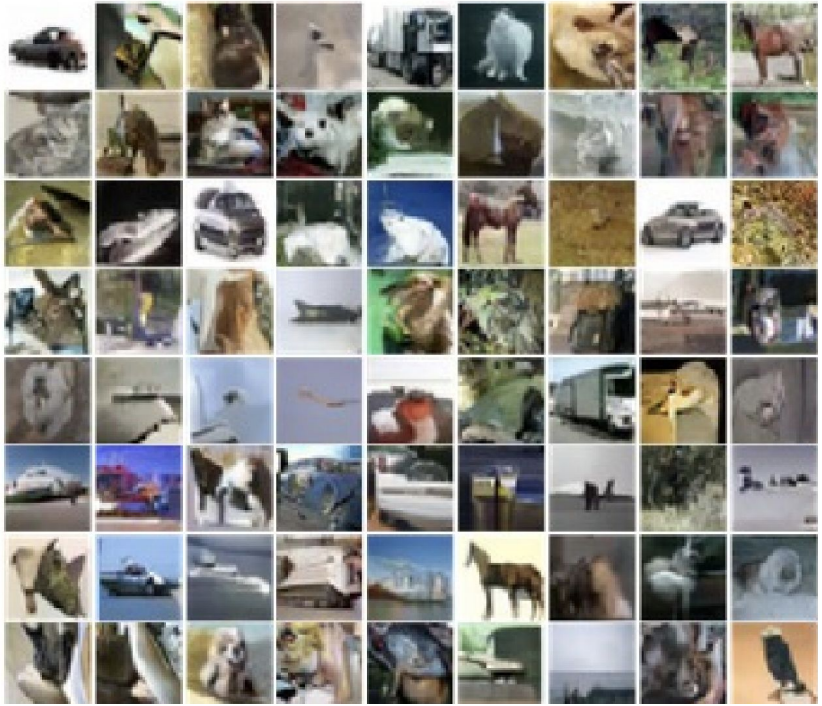
Multi-Head Self-Attention on MNIST

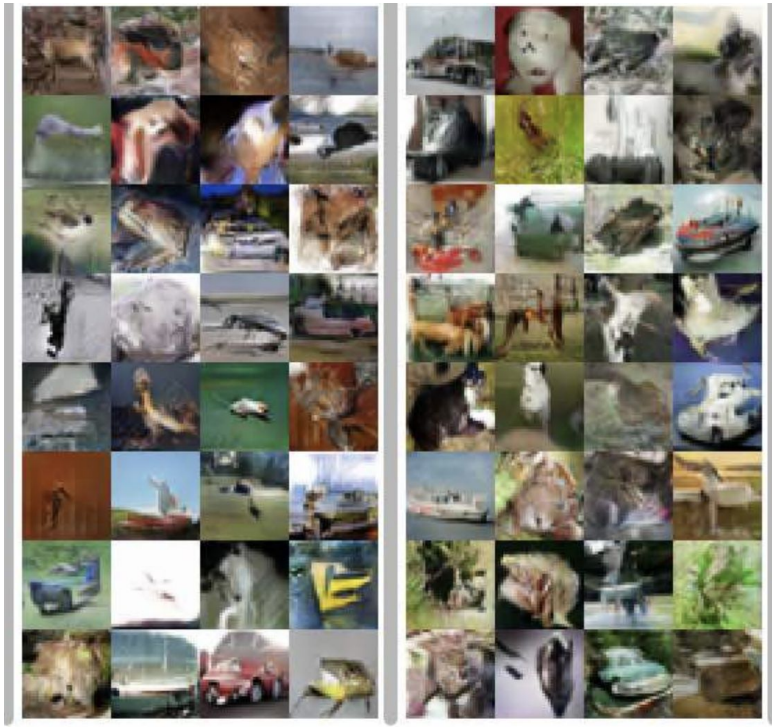


Masked Attention + Convolution

Method	CIFAR-10
Conv DRAW (Gregor et al., 2016)	3.5
Real NVP (Dinh et al., 2016)	3.49
VAE with IAF (Kingma et al., 2016)	3.11
PixelRNN (Oord et al., 2016b)	3.00
Gated PixelCNN (van den Oord et al., 2016b)	3.03
Image Transformer (Anonymous, 2018)	2.98
PixelCNN++ (Salimans et al., 2017)	2.92
Block Sparse PixelCNN++ (OpenAI, 2017)	2.90
PixelSNAIL (ours)	**2.85**

Sample Quality

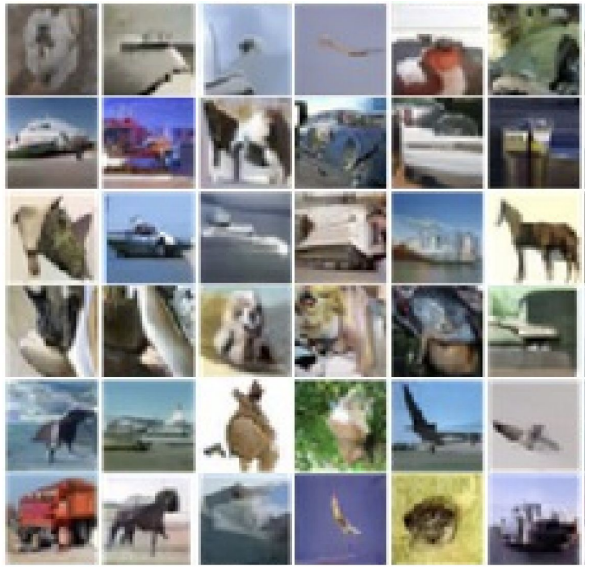


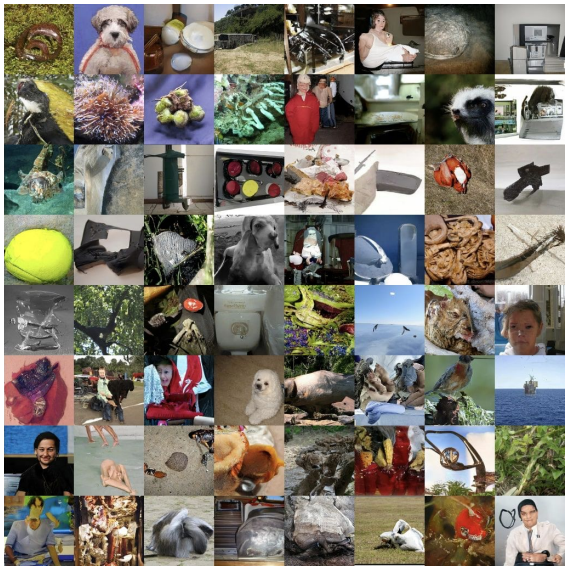


Which set of samples are generated by a GAN versus an AR model?

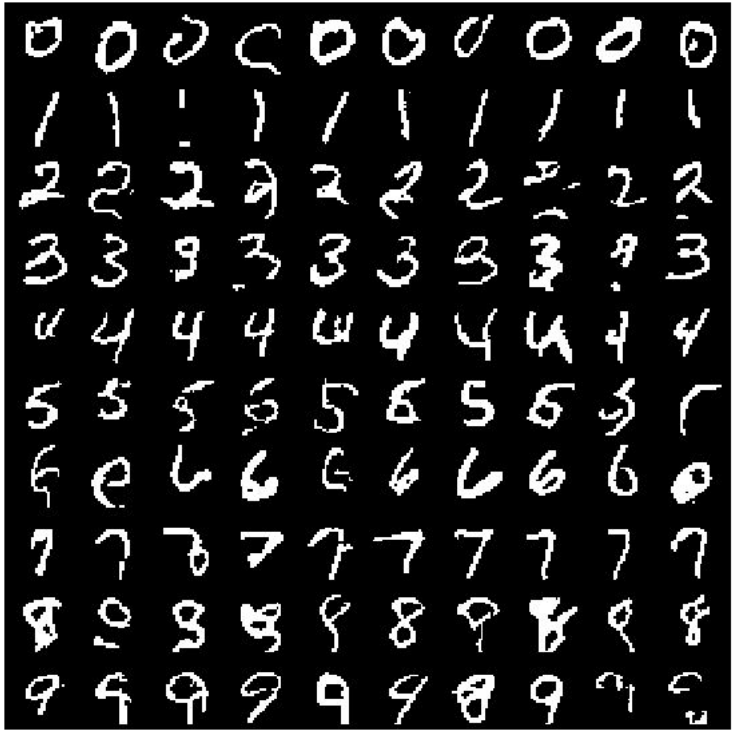
AR models can have good samples

- Good samples can be achieved by selective bits conditioning
  - Grayscale PixelCNN
  - Subscale Pixel Network





Class-Conditional PixelCNN



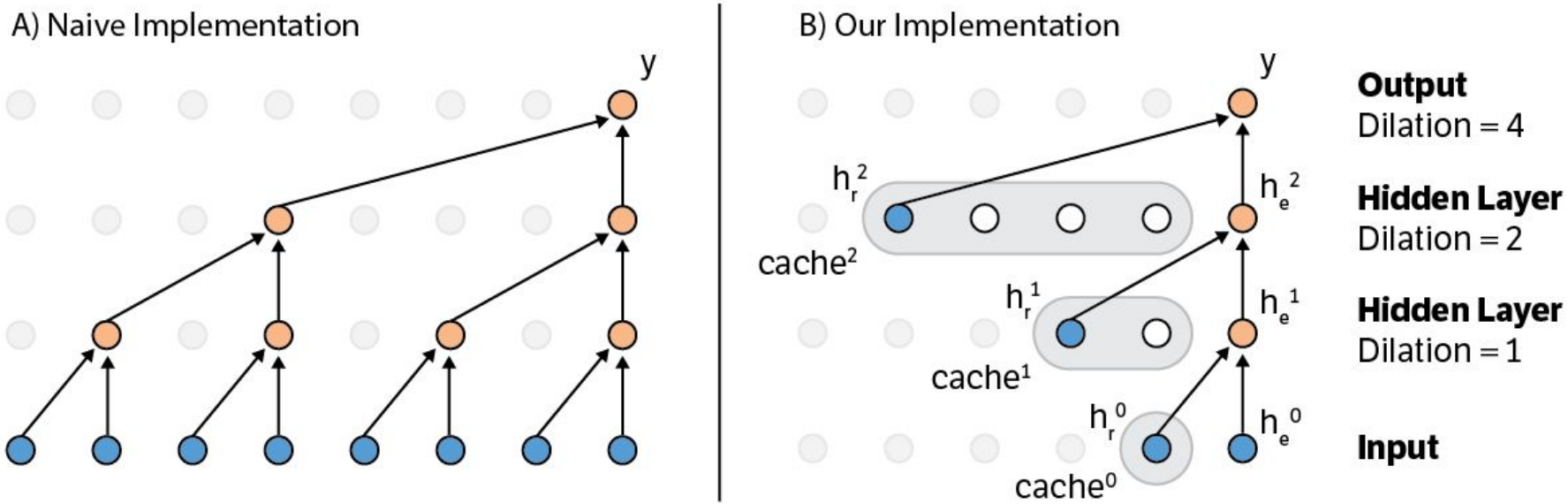
How to condition?

IN: One-hot encoding of the labels

THEN: multiplying by different learned weight matrices in each convolutional layer, and added as a bias channel-wise and broadcasted spatially

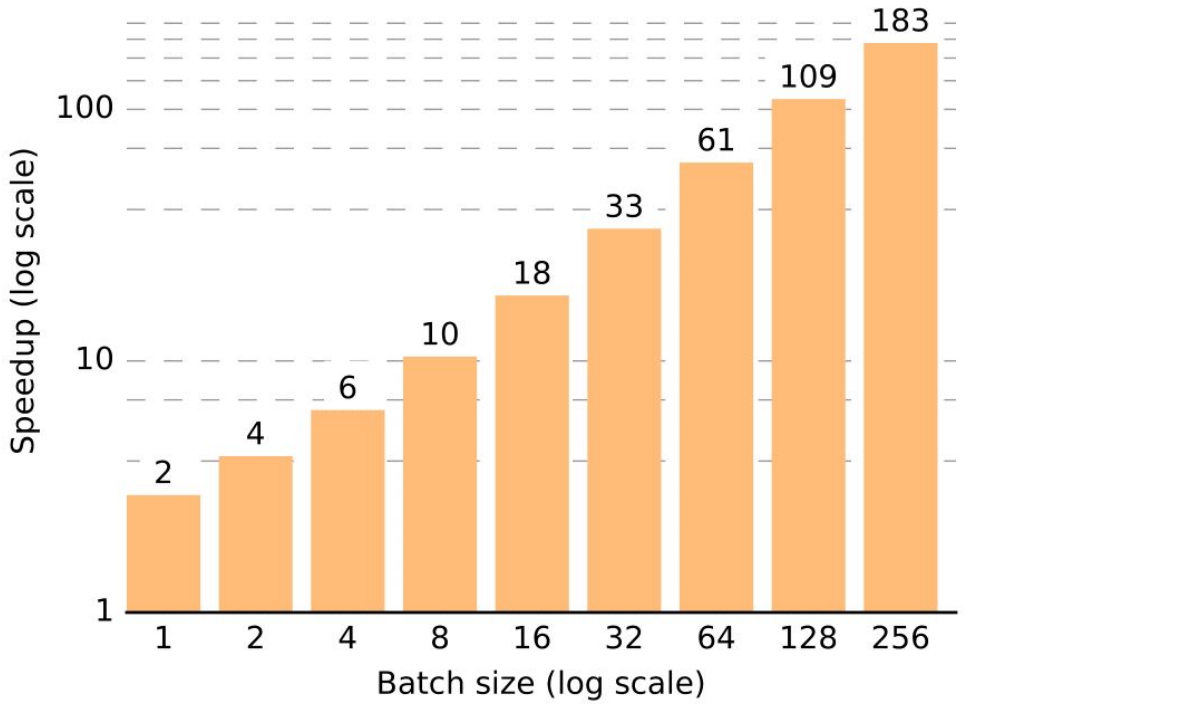
Speedup by caching activations

https://github.com/PrajitR/fast-pixel-cnn



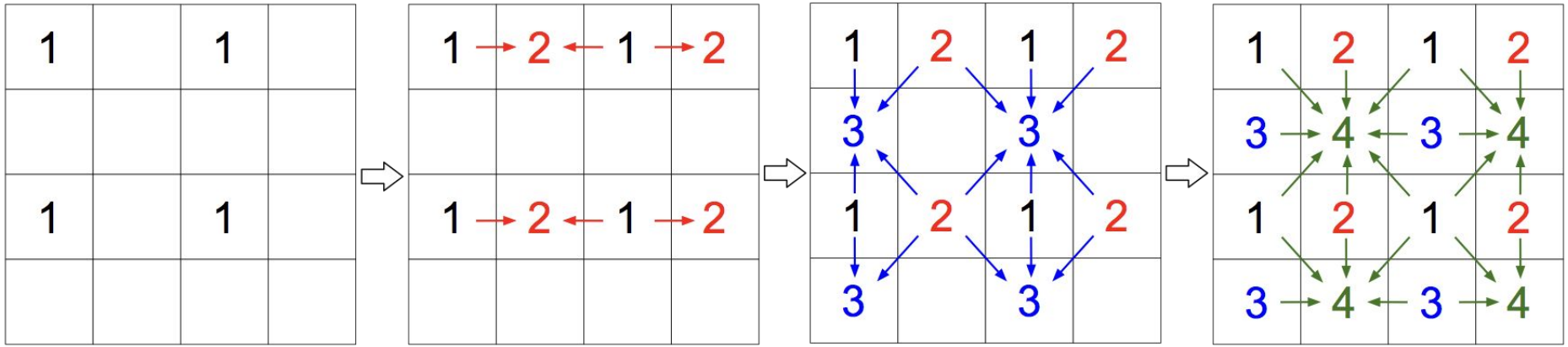
Speedup by caching activations

https://github.com/PrajitR/fast-pixel-cnn



Speedup by breaking autoregressive pattern

- O(d) -> O(log(d)) by parallelizing within groups {2, 3, 4}
- Cannot capture dependencies within each group: this is a fine assumption if all pixels in one group are conditionally independent
  - Most often they are not, then you trade expressivity for sampling speed



Multiscale PixelCNN

Model	scale	time	speedup
$O(N)$ PixelCNN	32	120.0	1.0×
$O(\log N)$ PixelCNN	32	1.17	102×
$O(\log N)$ PixelCNN, in-graph	32	1.14	105×

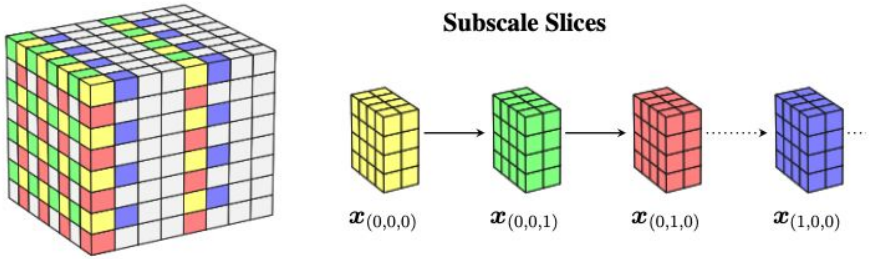
Improved sampling speed

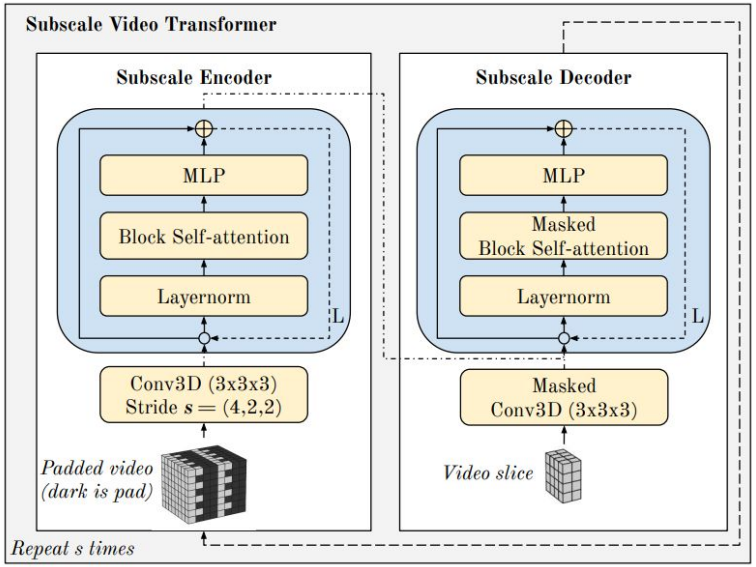
Model	32	64	128
PixelRNN	3.86 (3.83)	3.64(3.57)	-
PixelCNN	3.83 (3.77)	3.57(3.48)	-
Real NVP	4.28(4.26)	3.98(3.75)	-
Conv. DRAW	4.40(4.35)	4.10(4.04)	-
Ours	3.95(3.92)	3.70(3.67)	3.55(3.42)

Table 3. ImageNet negative log-likelihood in bits per sub-pixel at 32×32, 64×64 and 128×128 resolution.

More limited modelling capacity

Scaling Autoregressive Video Models



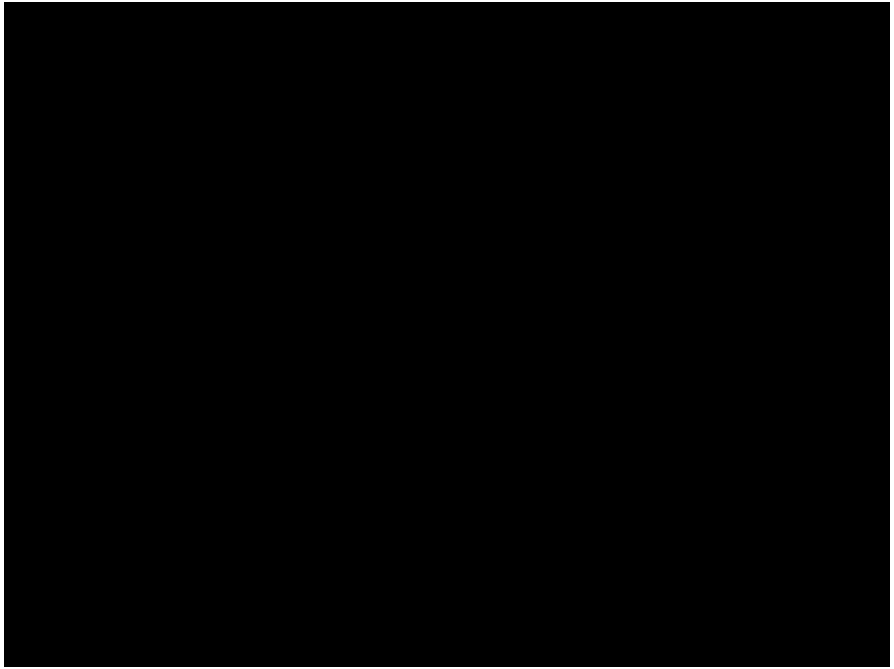


[Dirk Weissenborn, Oscar Tackstrom, Jakob Uszkoreit. "Scaling Autoregressive Video Models." arXiv 1906.02634 (2019)]

Scaling Autoregressive Video Models -- BAIR Robot Pushing

Large Spatiotemporal Subscaling

Small Spatiotemporal Subscaling



[Dirk Weissenborn, Oscar Tackstrom, Jakob Uszkoreit. “Scaling Autoregressive Video Models.” arXiv 1906.02634 (2019)]

Scaling Autoregressive Video Models -- Kinetics

Cooking (left-to-right by likelihood)

Full Kinetics (left-to-right by likelihood)

[Dirk Weissenborn, Oscar Tackstrom, Jakob Uszkoreit. “Scaling Autoregressive Video Models.” arXiv 1906.02634 (2019)]

Natural Image Manipulation for Autoregressive Models using Fisher Scores

- Main challenge:
  - How to get a latent representation from PixelCNN?
  - Why hard? The random input happens on a per-pixel sample basis

- Proposed solution
  - Use Fisher score

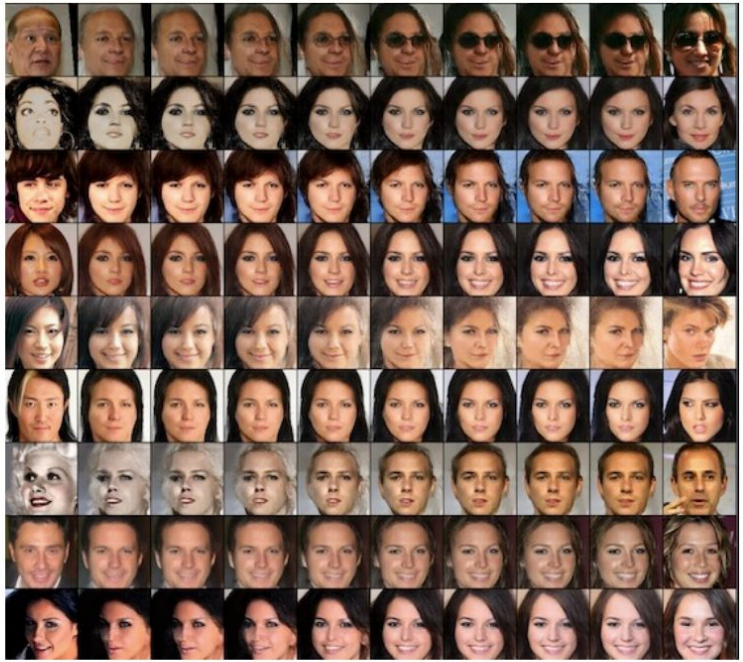
$$\dot{\ell}(x;\theta) = \nabla_\theta \log p_\theta(x)$$

Note: applicable to any likelihood model

[Wilson Yan, Jonathan Ho, Pieter Abbeel. ""Natural Image Manipulation for Autoregressive Models using Fisher Scores." arXiv 1912.05015

Natural Image Manipulation for Autoregressive Models using Fisher Scores

(c) Activations (Interpolation)



(d) Fisher score (Interpolation)

[Wilson Yan, Jonatha Ho, Pieter Abbeel. ""Natural Image Manipulation for Autoregressive Models using Fisher Scores." arXiv 1912.05015

Natural Image Manipulation for Autoregressive Models using Fisher Scores

[Wilson Yan, Jonatha Ho, Pieter Abbeel. ""Natural Image Manipulation for Autoregressive Models using Fisher Scores." arXiv 1912.05015