Deep Generative Models

Chapter 3: Generation by Explicit Distribution Learning

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Summer 2025

Need for a Computationally-Feasible Model

Clearly, modeling data distribution as a whole is computationally infeasible

- ! *The model needs to cover an exponentially large data-space*
- ! *Direct sampling could be exponentially hard*

To handle these issues, various modeling approaches can be deployed

1. *autoregressive (AR) Modeling* ←
2. *energy-based Modeling*
3. *flow-based Modeling*

In this section, we look into AR modeling

AR Modeling

Using the probability chain-rule: *we can green the data distribution as*

$$
\begin{aligned}
P(x) &= P(x_1, \ldots, x_d) \\
&= P(x_1)\, P(x_2|x_1)\, P(x_3|x_1, x_2) \ldots P(x_d|x_1, \ldots, x_{d-1}) \\
&\qquad \uparrow \quad\quad \uparrow \quad\quad\quad \uparrow \quad\quad\quad\quad\quad\quad \uparrow \\
&\qquad x_{<1} \quad\quad x_{<2} \quad\quad\quad x_{<3} \quad\quad\quad\quad\quad\quad x_{<d} \\
&= P(x_1|x_{<1})\, P(x_2|x_{<2})\, P(x_3|x_{<3}) \ldots P(x_d|x_{<d}) \\
&= \prod_i P(x_i|x_{<i})
\end{aligned}
$$

Key Point in AR Modeling

d-dimensional object is decomposed into product of d 1-dimensional objects

AR Model: *Generic*

AR Model: *Generic Form*

AR models represent the data distribution as

$$\prod_i P_{\mathbf{w}_i}\left(x_i | x_{<i}\right)$$

for some computational models $P_{\mathbf{w}_1}, \ldots P_{\mathbf{w}_d}$

+ *It does not seem to be any inaccuracy in this modeling! Right?!*
– Exactly! If each conditional distribution $P\left(x_i | x_{<i}\right)$ is learned perfectly, the model is exact!
+ *Then how does it address the key challenges?*
– We will see it after looking at generation! Stay tuned!

AR Modeling: *Example*

+ *The AR model reminds me of LMs!*
– Exactly! LMs are indeed AR models
+ *Can we generate any other modality using AR modeling?*
– Sure!

- *For text, we can generate one token at a time* $\rightsquigarrow$ *AR model*
- *For images, we can*
  - ↳ *generate a single pixel at a time* $\rightsquigarrow x_i \in \mathbb{R}$
  - ↳ *generate one RGB pixel at a time* $\rightsquigarrow x_i \in \mathbb{R}^3$
  - ↳ *generate a* $k \times k$ *RGB pixel-patch at a time* $\rightsquigarrow x_i \in \mathbb{R}^{k \times k \times 3}$
- *For audio, we can generate a single frame at a time*

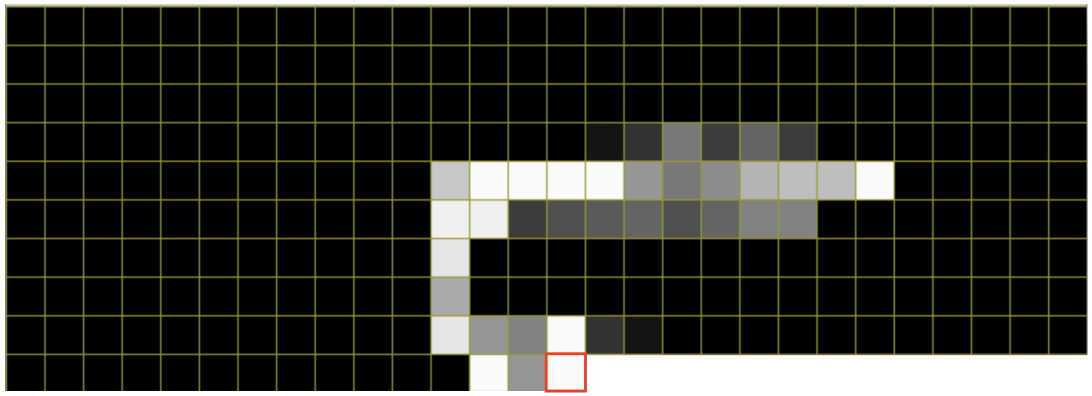
AR Modeling: *Example of MNIST*

$P_{\mathbf{w}}(x_1)$

AR Modeling: *Example of MNIST*

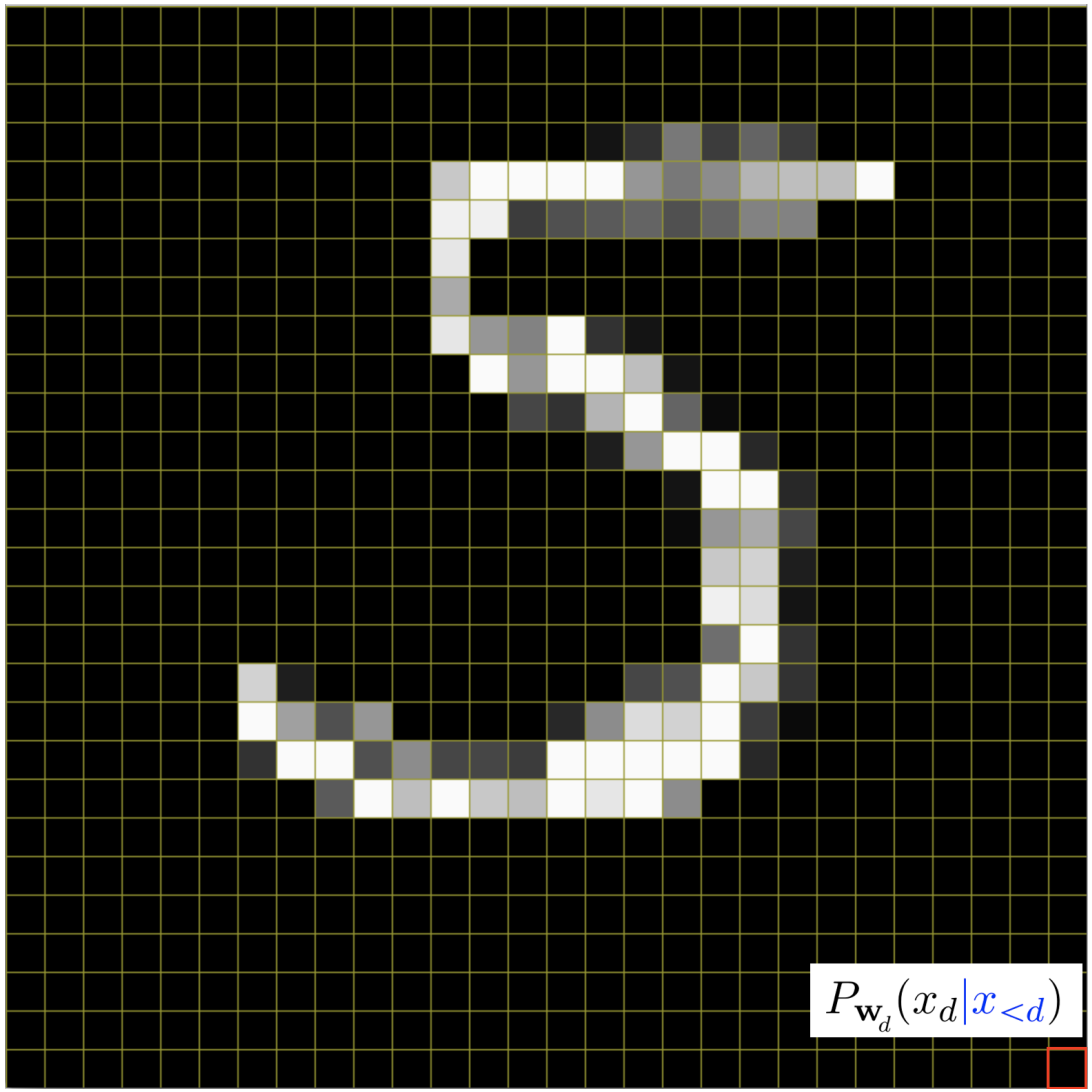
$P_{\mathbf{w}_2}(x_2|x_1)$

AR Modeling: *Example of MNIST*



$$P_{\mathbf{w}_i}\left(x_i | x_{<i}\right)$$

AR Modeling: *Example of MNIST*



Sampling AR Models

Say we have trained the AR model

$$\prod_i P_{\mathbf{w}_i}\left(x_i | x_{<i}\right)$$

to sample this model, we

- *first sample* $x_1 \sim P_{\mathbf{w}_1}\left(x_1\right) \rightsquigarrow$ *complexity* $= \mathcal{O}\left(C\right)$
- *then use* x_1 *to sample* $x_2 \sim P_{\mathbf{w}_2}\left(x_2 | x_1\right) \rightsquigarrow$ *complexity* $= \mathcal{O}\left(C\right)$

$\vdots$

- *finally use* $x_{<d}$ *to sample* $x_d \sim P_{\mathbf{w}_d}\left(x_d | x_{<d}\right) \rightsquigarrow$ *complexity* $= \mathcal{O}\left(C\right)$

Conclusion: *Linear Sampling Complexity*

We can sample with linear complexity, i.e., $\mathcal{O}\left(Cd\right)$*!*

Sampling AR Models: *Trade-off*

Conclusion: *Linear Sampling Complexity*

We can sample with linear complexity, i.e., $\mathcal{O}(Cd)$!

\+ *Sounds weird! Isn't there any cost we pay for this huge complexity reduction?*

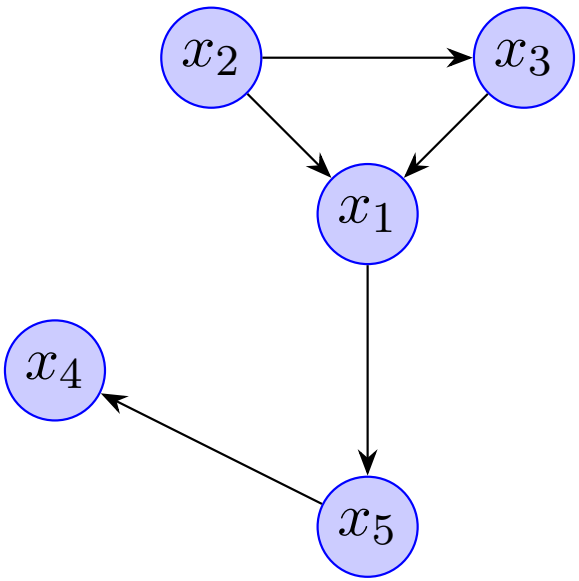
– Sure! The cost is *autoregressive generation* which is slow!

- *In a joint model $P_{\mathbf{w}}(x)$, we generate the sample in one-shot*
  - ↳ *For example, the whole image at once*
- *In AR models, we generate the sample entry-by-entry*
  - ↳ *For example, we generate the image pixel-by-pixel*

Favorable Partitioning

In general, the way we partition entries in AR modeling is important!

Bayes graph



There are various ways we can write the chain-rule

$$
\begin{aligned}
P(x) &= P(x_1)\, P(x_2|x_1)\, P(x_3|x_1, x_2) \\
&\quad P(x_4|x_1, x_2, x_3)\, P(x_5|x_1, x_4) \\
&= P(x_2)\, P(x_3|x_2)\, P(x_1|x_2, x_3) \\
&\quad P(x_5|x_1)\, P(x_4|x_5)
\end{aligned}
$$

Finding optimal partitioning is though NP-hard

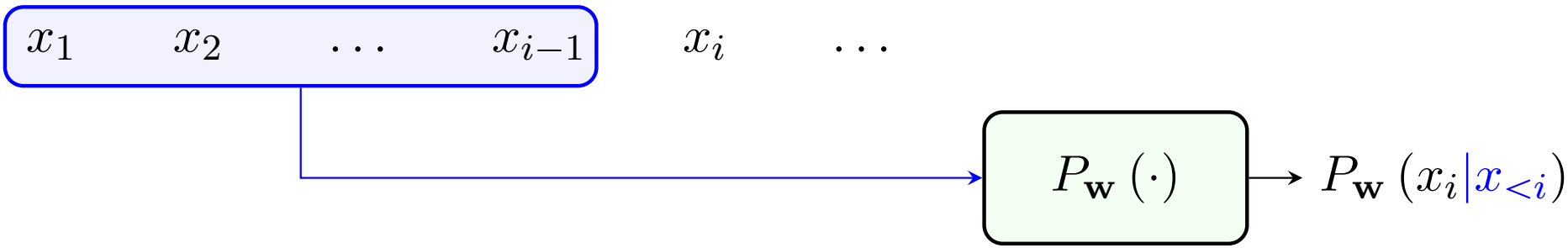
We typically use intuition to find good partitioning

AR Model *with Shared Parameters*

AR Model *with Shared Parameters*

In practice, all conditional distributions are represented by the same model

$$P_{\mathbf{w}}(x) = \prod_i P_{\mathbf{w}}(x_i | x_{<i})$$



Probability Computation via *Context Extraction*

Computational AR models follow the same approach as we did in LMs

- *For each x_i, a context is extracted from $x_{<i}$, i.e.,*

$$\mathbf{c}_i = f_{\boldsymbol{\theta}}\left(x_{<i}\right)$$

where $\boldsymbol{\theta} \subset \mathbf{w}$ contain the major subset of model parameters

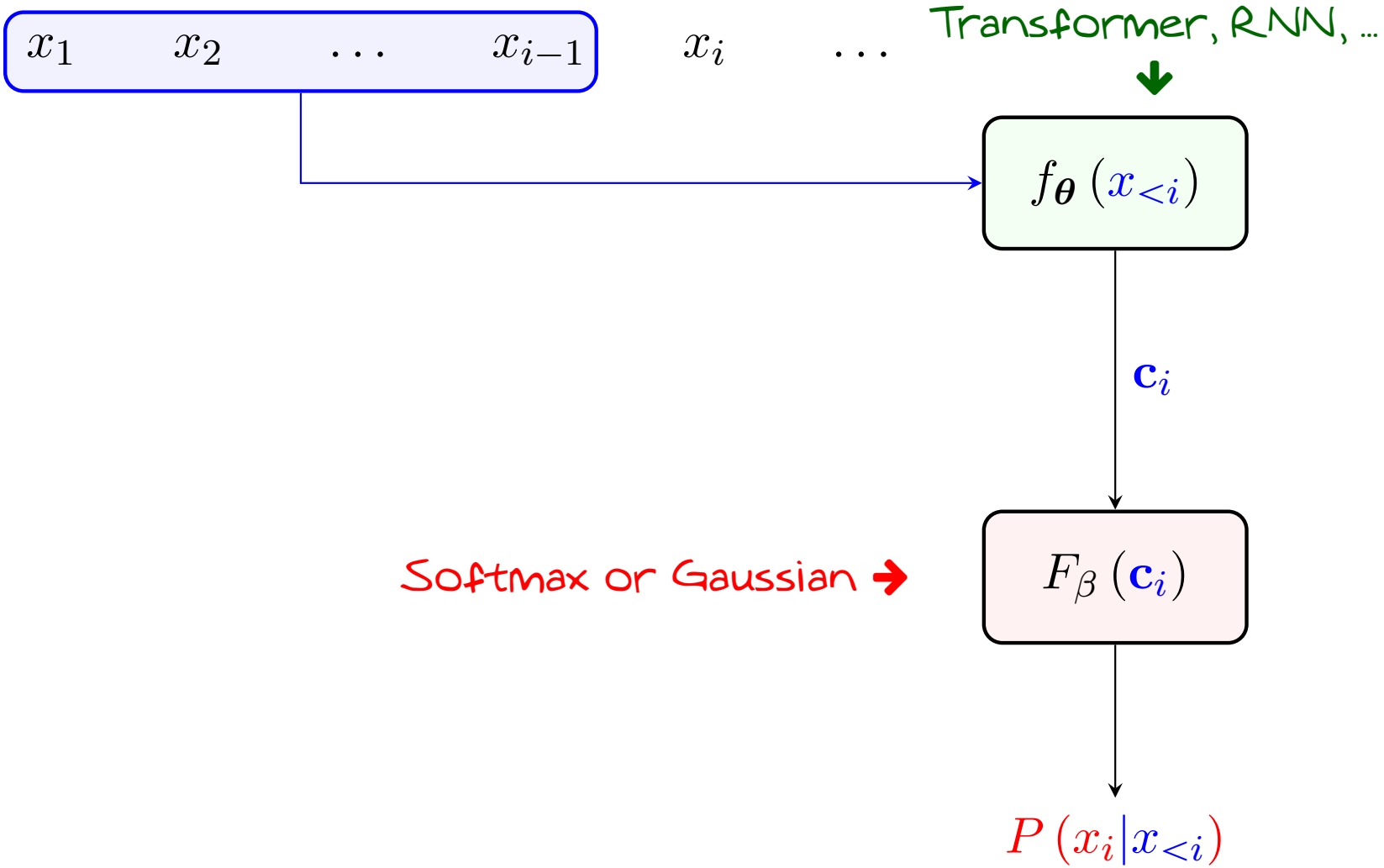
- *We compute $P\left(x_i|x_{<i}\right)$ from the context using a probability-defining layer*

$$P_{\mathbf{w}}\left(x_i|x_{<i}\right) = F_{\beta}\left(\mathbf{c}_i\right) = F_{\beta}\left(f_{\boldsymbol{\theta}}\left(x_{<i}\right)\right)$$

where $\mathbf{w} = \{\boldsymbol{\theta}, \beta\}$; examples of such layers are

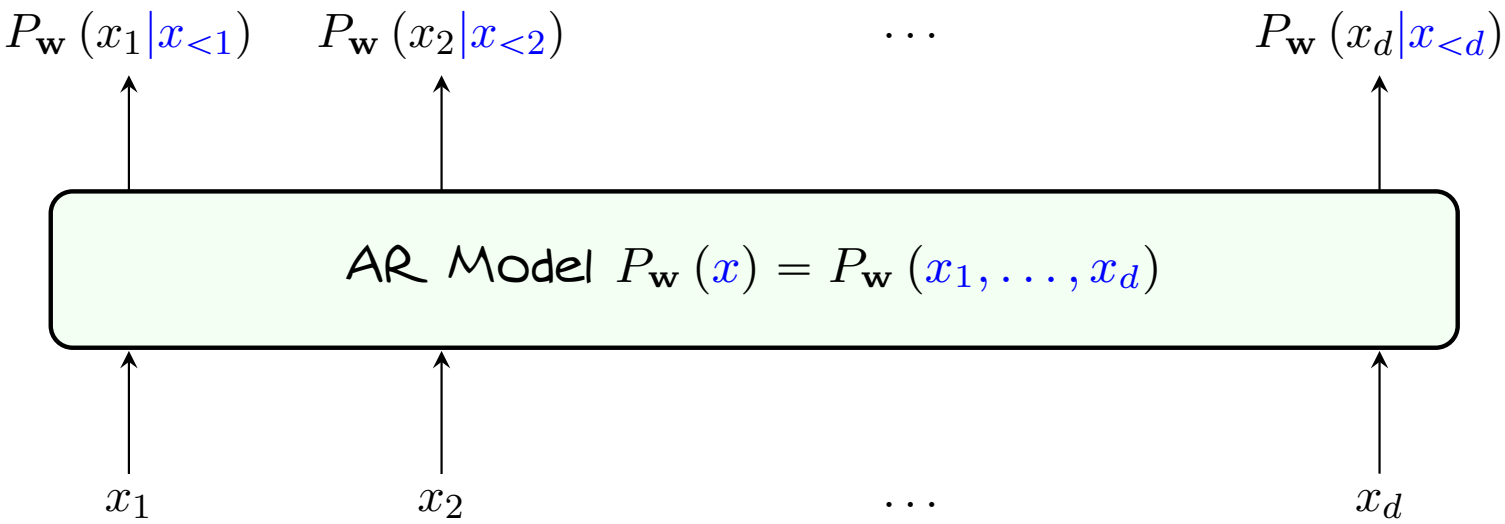
  - ↳ *a $\text{Soft}_{\max}$ activated layer, or*
  - ↳ *a Gaussian distribution whose mean and covariance depend on $\mathbf{c}_i$*

Probability Computation via *Context Extraction*



AR as *Computational Model*

From a computational perspective: *AR is a model with d inputs and d outputs*



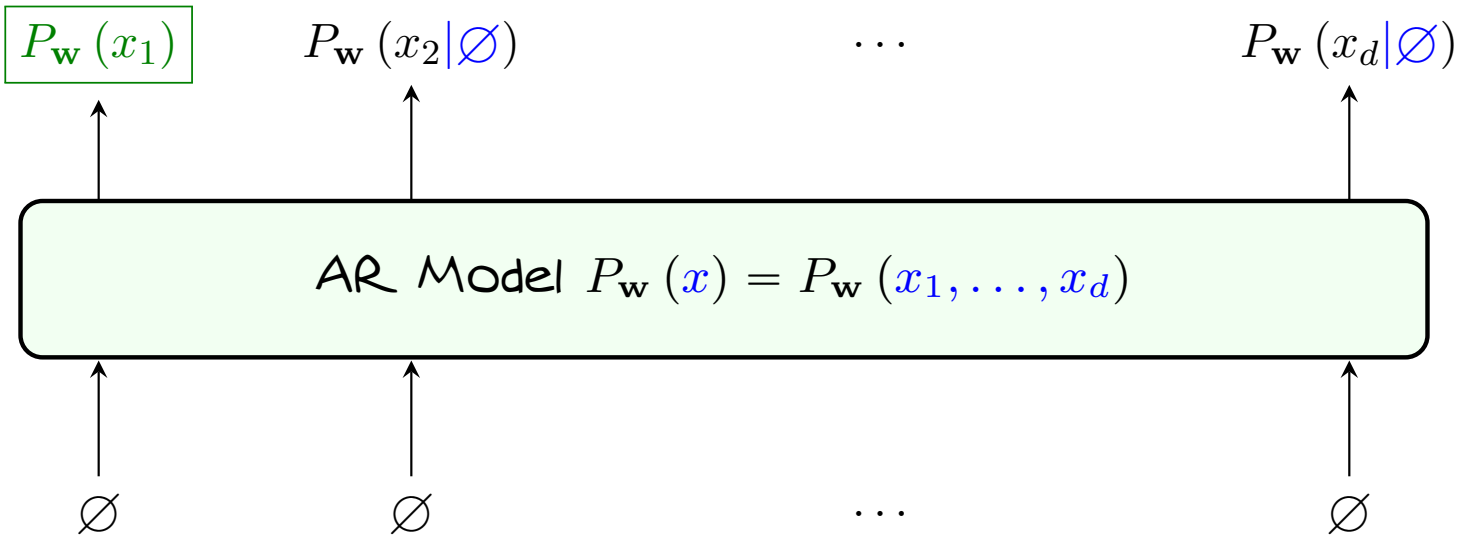
This computational model can be realized by any architecture

- *Basic MLP or an RNN* ✔
- *Transformer* ✔
- *Convolutional network* ✖

Autoregressive Generation

+ *Say we trained the model, how do we generate a sample?*
– We should do it autoregressive!

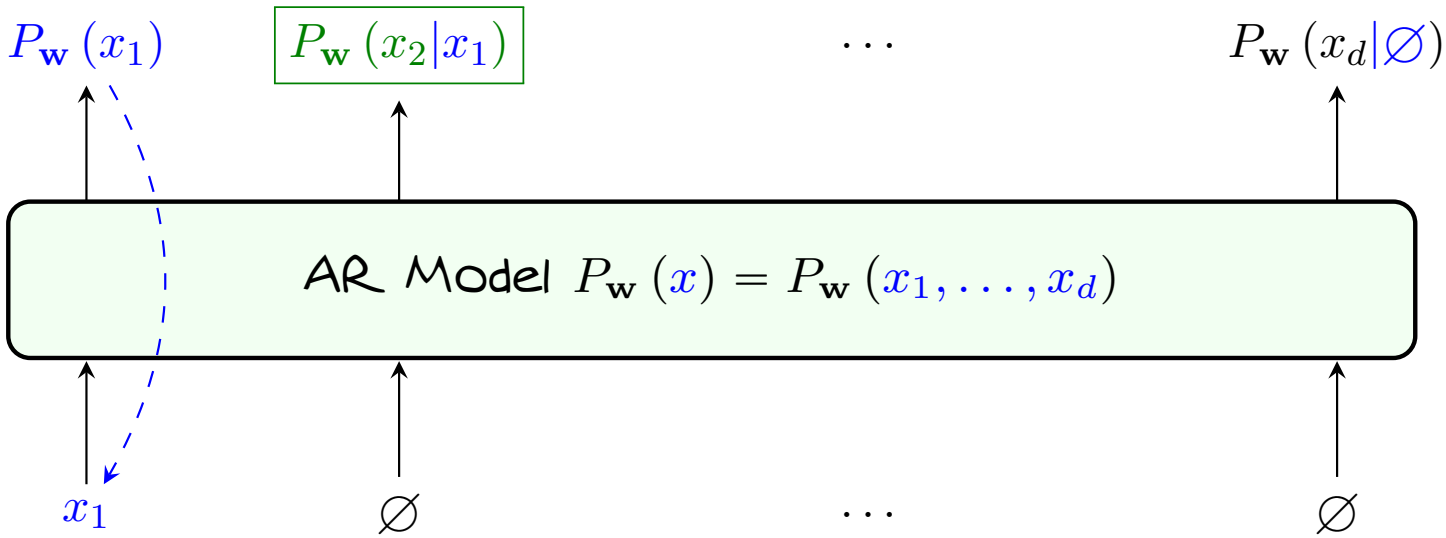
We first sample the first component



Autoregressive Generation

+ *Say we trained the model, how do we generate a sample?*
– We should do it autoregressive!

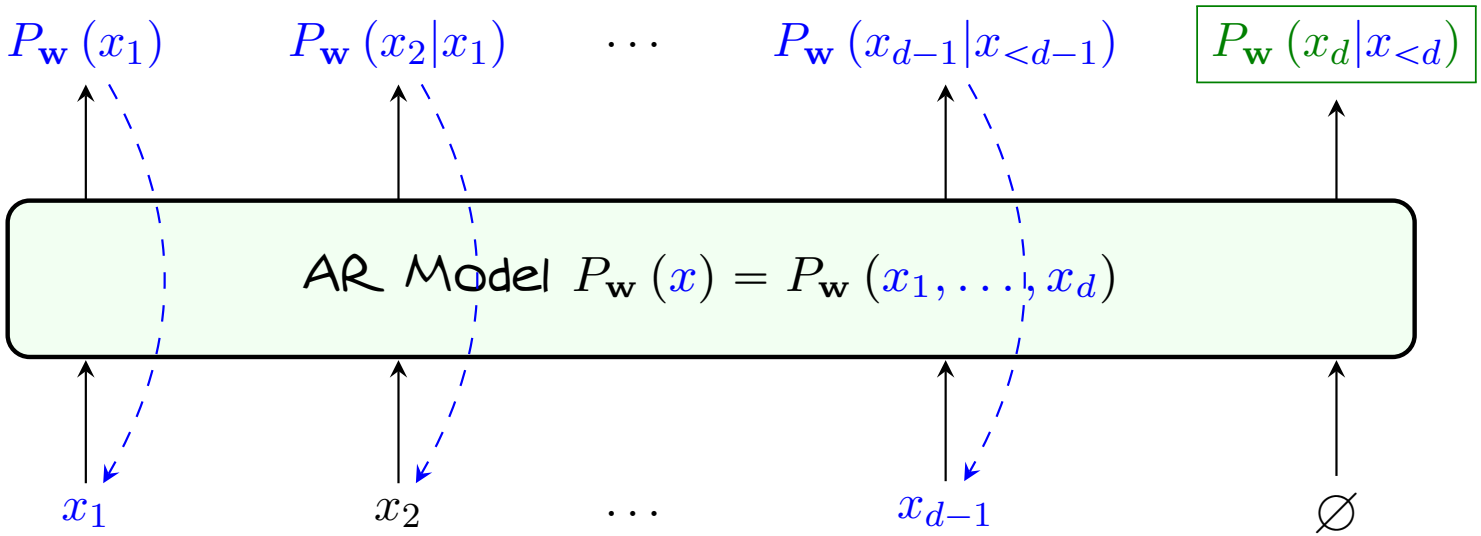
We then inject first component and sample the second component



Autoregressive Generation

+ *Say we trained the model, how do we generate a sample?*
– We should do it autoregressive!

We keep repeating till we sample the last component



Bottleneck: *Slow Generation*

+ *This sounds to take a long time!*
– Sure! And this is a key challenge of *autoregressive generation*

Example: *Say we want to generate samples from CelebA*

↳ *CelebA is a dataset of 200K celebrity face images*

↳ *Each image is RGB with resolution* 178×218

Say each $x_i \in \mathbb{R}^3$ *is an RGB pixel* $\rightsquigarrow$ *we need 39K forward passes!*

Moral of Story

AR models are slow in generation due to their autoregressive nature of sampling

Training AR Models via MLE

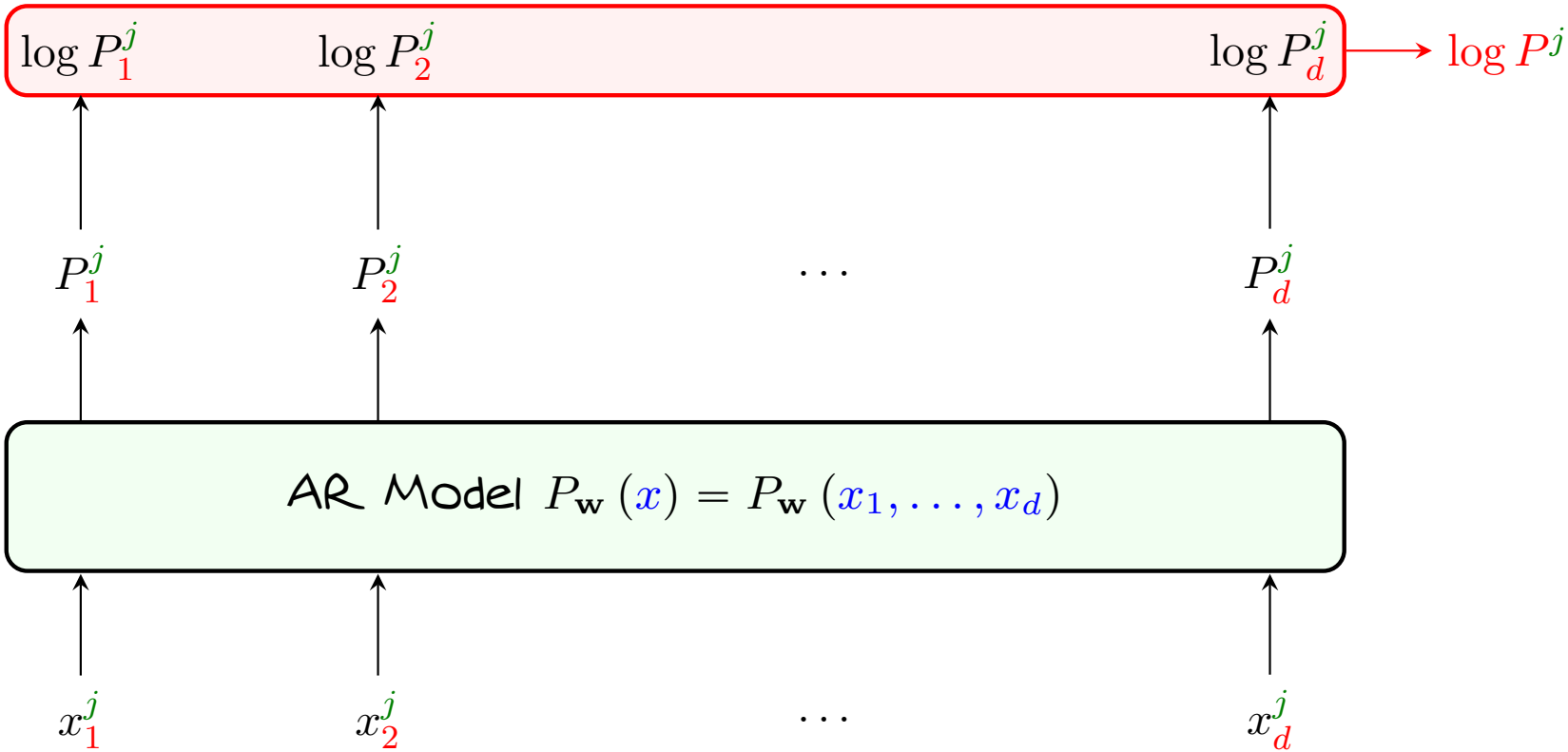
+ *What about their training?*
– Well, we should follow the general recipe $\rightsquigarrow$ MLE

Say we have $\mathbb{D} = \left\{x^j \text{ for } j = 1, \ldots, n\right\}$*: the log-likelihood is computed ad*

$$\frac{1}{n}\sum_{j=1}^{n} \log P_{\mathbf{w}}\left(x^j\right) = \frac{1}{n}\sum_{j=1}^{n} \log \prod_{i} P_{\mathbf{w}}\left(x_i^j | x_{<i}^j\right)$$

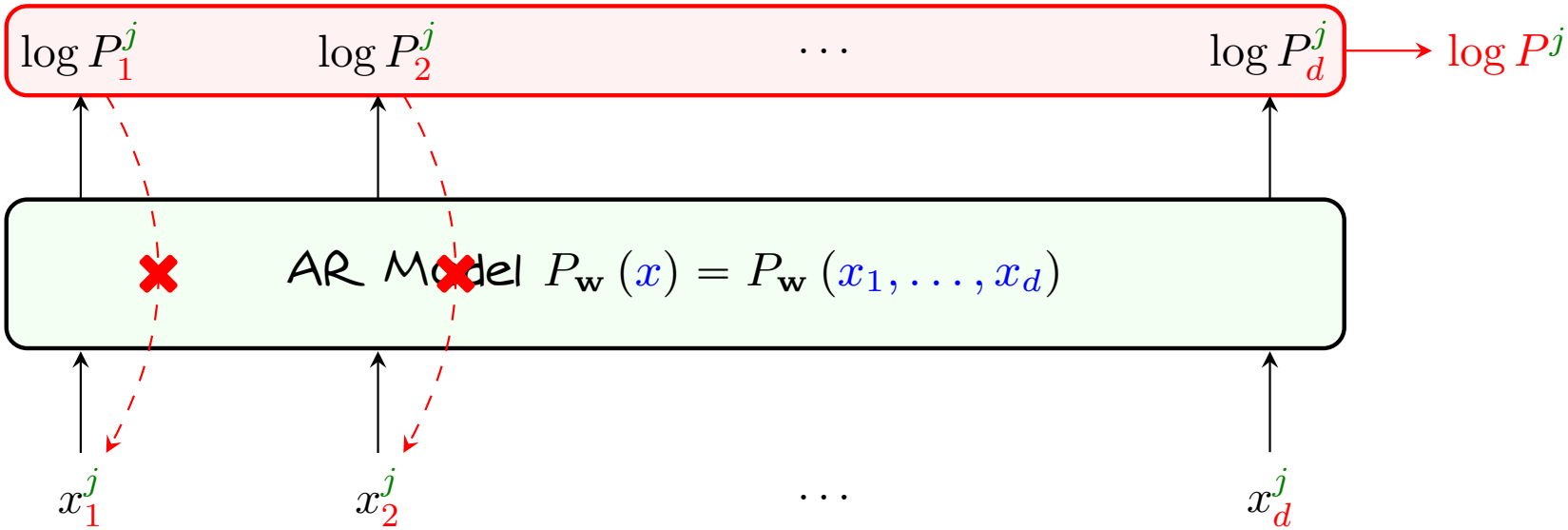
$$= \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{d} \log P_{\mathbf{w}}\left(x_i^j | x_{<i}^j\right)$$

Training AR Models via MLE



+ *This does not seem to be autoregressive anymore?*
- If model does parallel computation, e.g., as in transformers, No!
  - ↳ *This is the teacher-forcing training!*

Teacher-Forcing Training



Teacher-Forcing Training

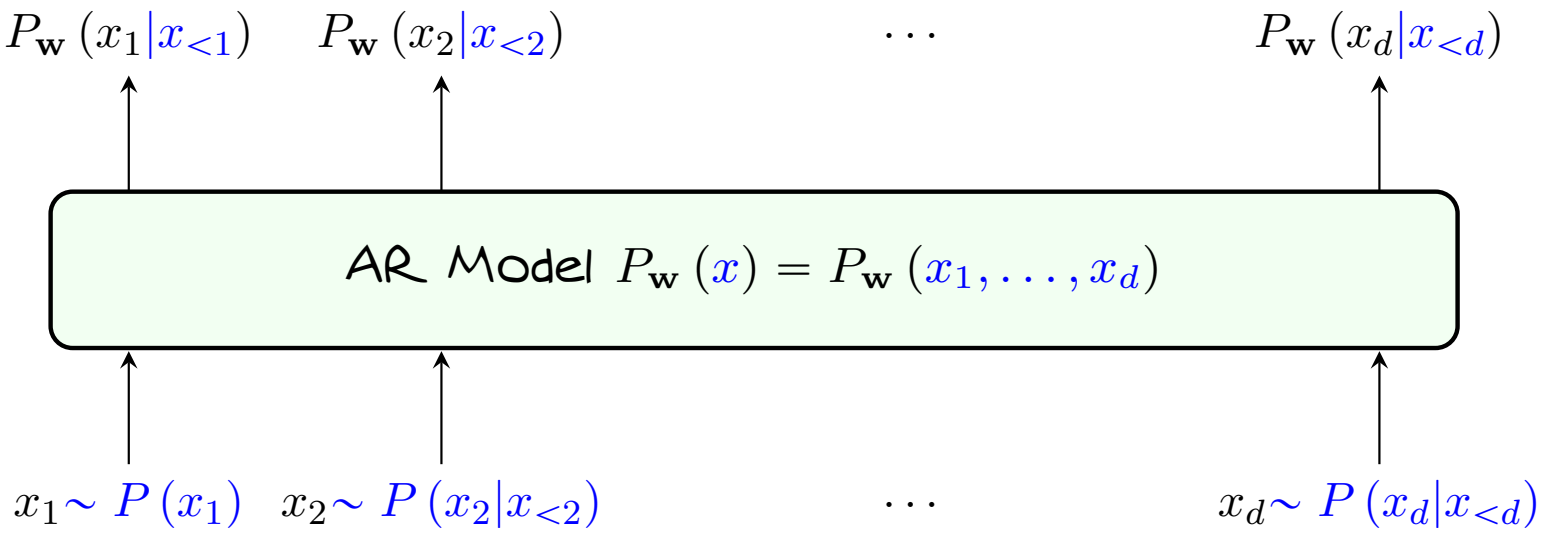
In AR models, likelihood computation can be in parallel, i.e., non-autoregressive, using the ground truth

- ↳ *This requires parallelized computational model, e.g., transformers*
- ↳ *This looks like a teacher forcing input to be sampled from data distribution*

Teacher-Forcing: *Exposure Bias*

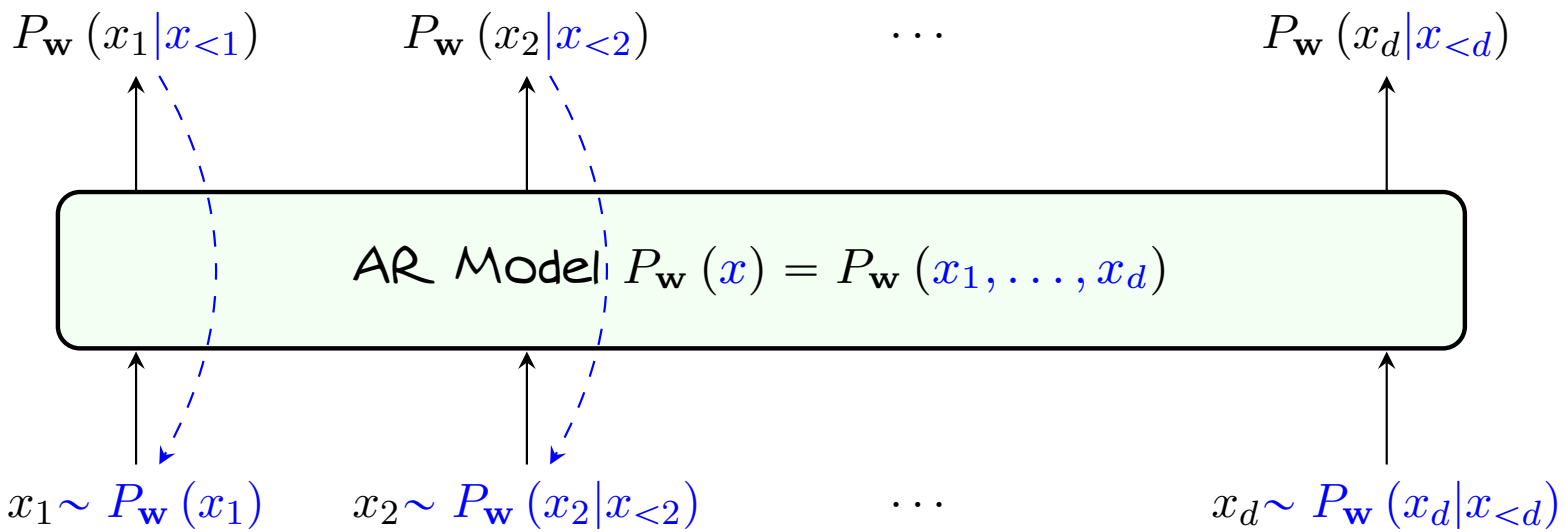
Teacher forcing makes difference between training and generation

During training the input is sampled from data distribution



Teacher-Forcing: *Exposure Bias*

However, during generation inputs are sampled from model



This means that during generation, model sees samples of different distribution

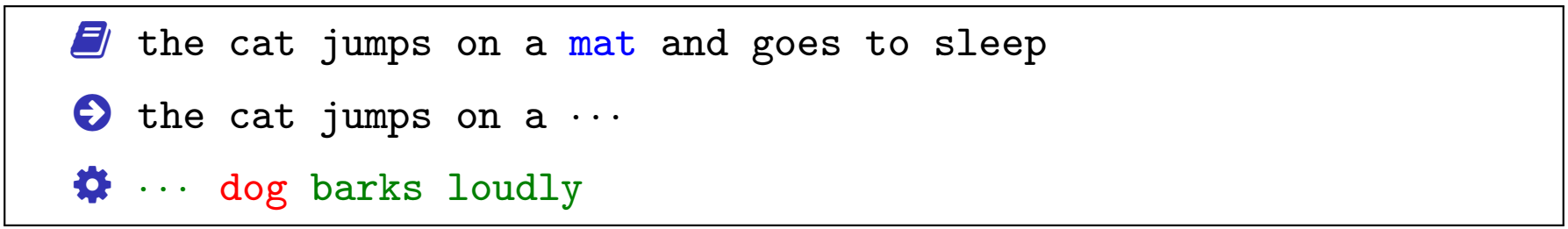
+ *Don't we learn the data distribution?!*
– *At the end of the day, we are still far away from it!*
  ↳ *We really cannot learn exact distribution! We get close to in a tiny subspace*

Teacher-Forcing: *Exposure Bias*

Exposure Bias

Exposure bias occurs due to mismatch between input distributions AR model sees during training and generation

Example: *For instance in LMs, exposure bias can cause such a scenario*



+ *Is there any solution to that?!*
– *Yes, there are several solutions*
  ↳ *Scheduled sampling gradually replaces true samples with generated ones*
  ↳ *Reinforcement learning can be also used to learn a reward system that improves model generalization*

Wrapping Up: *AR Models*

- *AR models represent the distribution as product of conditionals*
- *Sampling these models is very easy*
  - ↳ *It is however very slow*
- *Training can speed up using teacher forcing approach*
  - ↳ *It can however lead to exposure bias*

There are various examples of AR models

- ✔ *Language models*
- ➔ *RNN and CNN based AR models of image generation*
  - ↳ *PixelRNN and PixelCNN*
- ➔ *Transformer-based AR models of image generation*
  - ↳ *Image GPT*

We next take a look at visual AR models