上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Lecture 23: Probabilistic Generative Modeling

Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

https://taohuang.info/cs3317

https://oc.sjtu.edu.cn/courses/89538

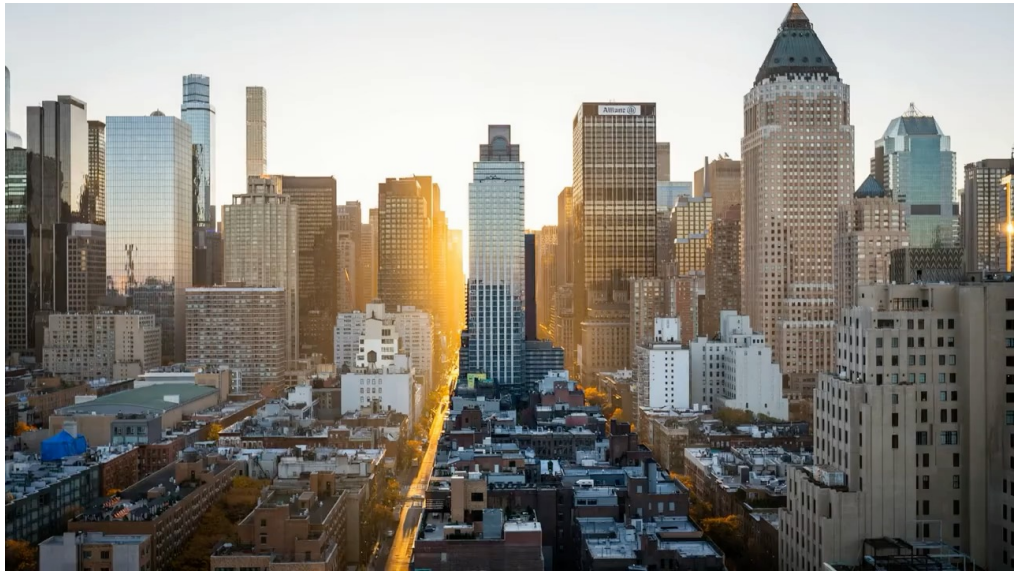
AI tools assisted in generating some figures in these slides. All such content has been reviewed, and the instructor is responsible for its accuracy.

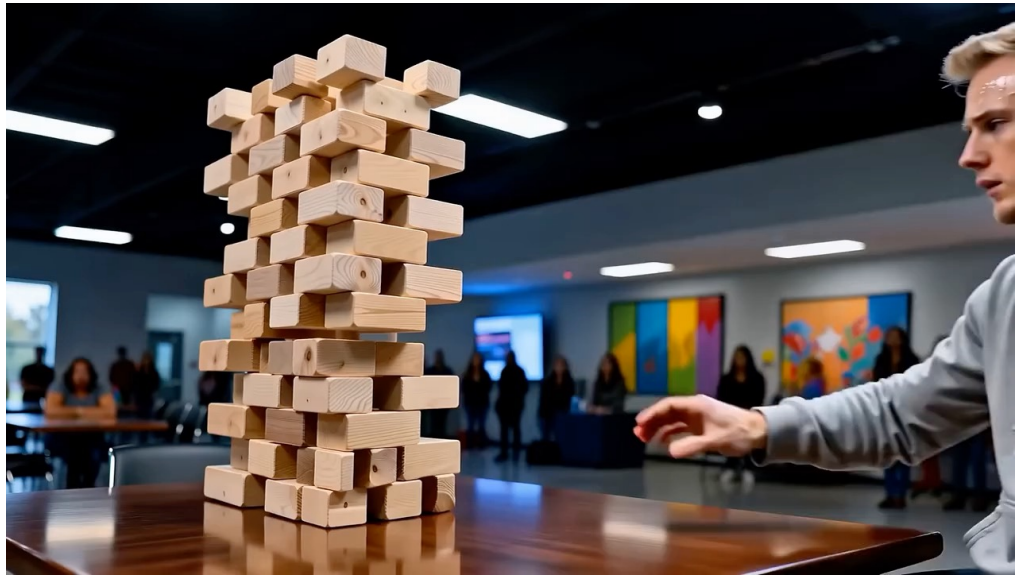
From Predicting to Imagining

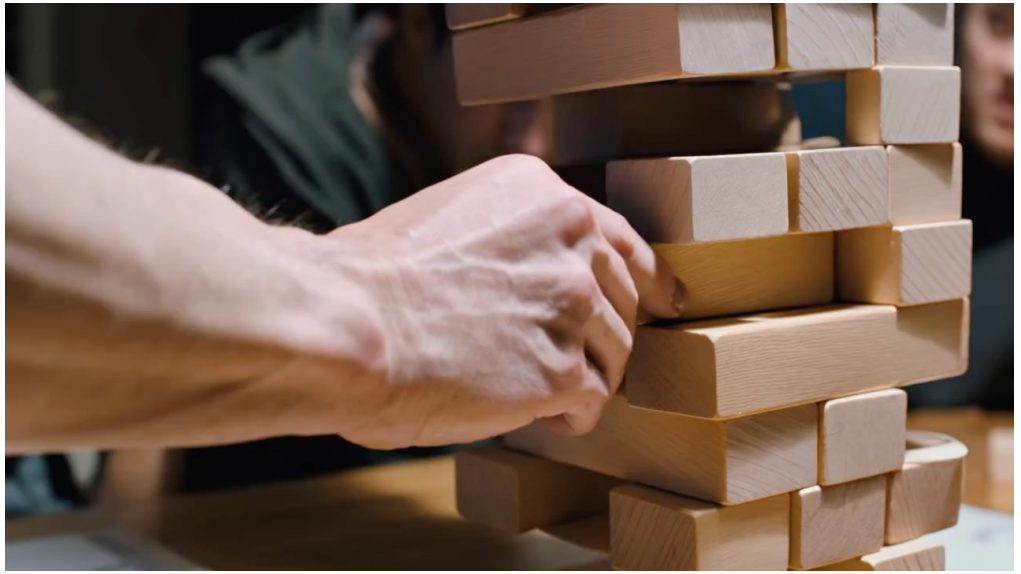
- **Modules 3–4** built models that *predict* — y from x. Classifiers, regressors, Transformers fine-tuned for sentiment, segmentation, translation.
- But the most striking AI systems of the last 5 years don't predict — they ***generate.***

- *ChatGPT writes essays. Kling/Seedance renders video. FLUX paints. AlphaFold designs proteins.*

From Predicting to Imagining







https://artificialanalysis.ai/video/arena

You've Already Built a Generative Model in L22

- Last week we built the Transformer as a "next-token predictor".
- **That framing was a half-truth.**
  A model that predicts the next token, conditioned on all previous tokens,
  is exactly a generative model of text.

- GPT, Qwen, DeepSeek — they are L22, scaled up, with one conceptual reframe.

When does prediction become generation?

Objectives

By the end of this lecture, you should be able to:

- **Distinguish** discriminative and generative models, and explain why generation is fundamentally harder.
- **Derive** the autoregressive factorization from the chain rule of probability.
- **Analyze** why autoregressive models give exact likelihood while latent-variable and implicit models do not.
- **Evaluate** when autoregressive modeling is the right tool — and when it's the wrong one.
- **Recognize** GPT-style language models as the canonical autoregressive generative model.

Roadmap of Module 5: Generative AI

- *Week 9 — Three classical answers to "how do we model p(x)?"*
  - **L25 (current): autoregressive — chain rule, exact likelihood**
  - L26: latent-variable — introduce z, optimize a lower bound (VAE)
  - L27: implicit — skip likelihood, learn via a game (GAN)

- *Week 10 — Modern answers and applications*
  - L28: score-based / diffusion — model $\nabla \log p(x)$, denoise
  - L29: LLMs — autoregressive at extreme scale
  - L30: VLMs — autoregressive + diffusion fused

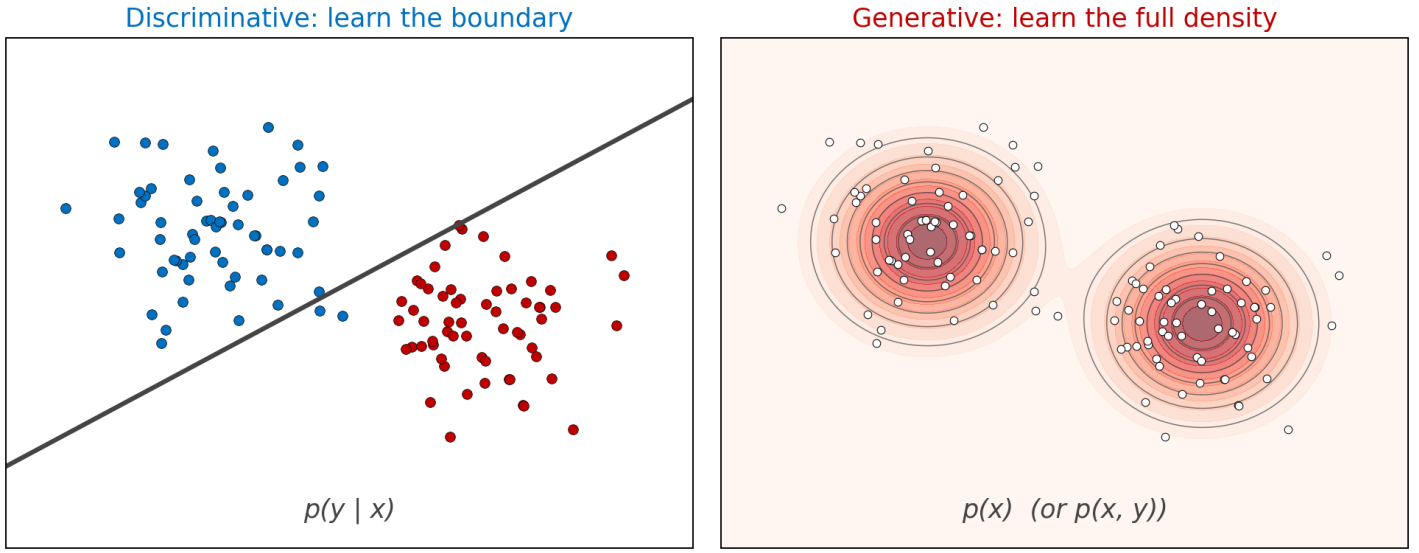
1. What Does It Mean to Model Data?

Discriminative: p(y | x)

Goal: given x, predict y.

- Examples: spam classifier, image classifier, machine translation.
- We learned a function $f_\theta: x \rightarrow y$, possibly probabilistic: $p_\theta(y|x)$.

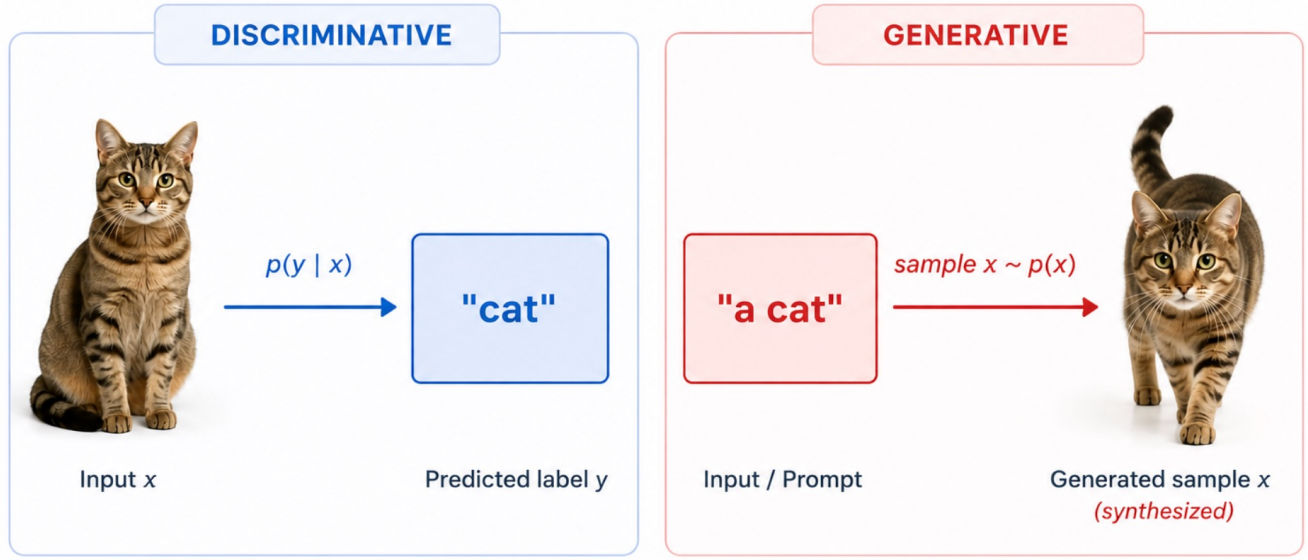
- The model only needs to know what separates classes.
- It doesn't need to know what cats look like — only how cats differ from dogs.



Generative: p(x), or p(x, y)

Goal: model the *full distribution* of the data itself.

- Examples: generate text, synthesize images, design molecules.
- **We learn p_θ(x) and can:**
  - Sample: draw new x ~ $p_\theta(x)$
  - Conditionally generate: x ~ $p_\theta(x \mid y)$

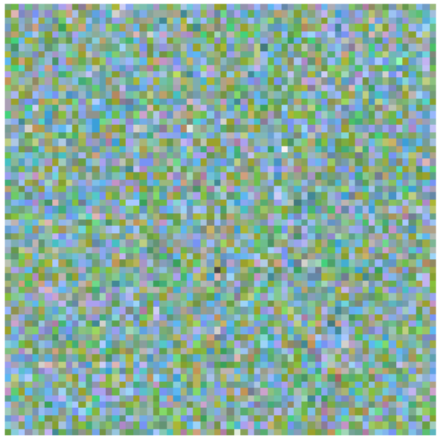


The model has to know what cats look like — not just how to recognize one.

🧠 Think: Which Is Harder?

- Discriminative: p(y | x). For ImageNet, y has 1000 values.
- Generative: p(x). For ImageNet, x is 224×224×3 — a distribution over a 150,528-D space.
- ***The output space of generative modeling is exponentially larger.***

Sample each pixel independently from its marginal



A real image

If x is a 1024×1024 image, the support of p(x) has more configurations than atoms (10^{80} ~ 10^{82}) in the universe.

How can any model possibly cover it?

Why Generation Is Fundamentally Harder

1. Higher-dimensional output.

- Discriminative outputs a scalar/vector; generative outputs the whole x.

2. No labels to anchor learning.

- Generative is mostly unsupervised — we have data, not (x, y) pairs.

3. Evaluation is hard.

- "Did the model classify correctly?" has a clear answer.
- "Did the model generate a good image?" does not.

Three Things a Generative Model Should Do

A useful generative model supports some subset of:

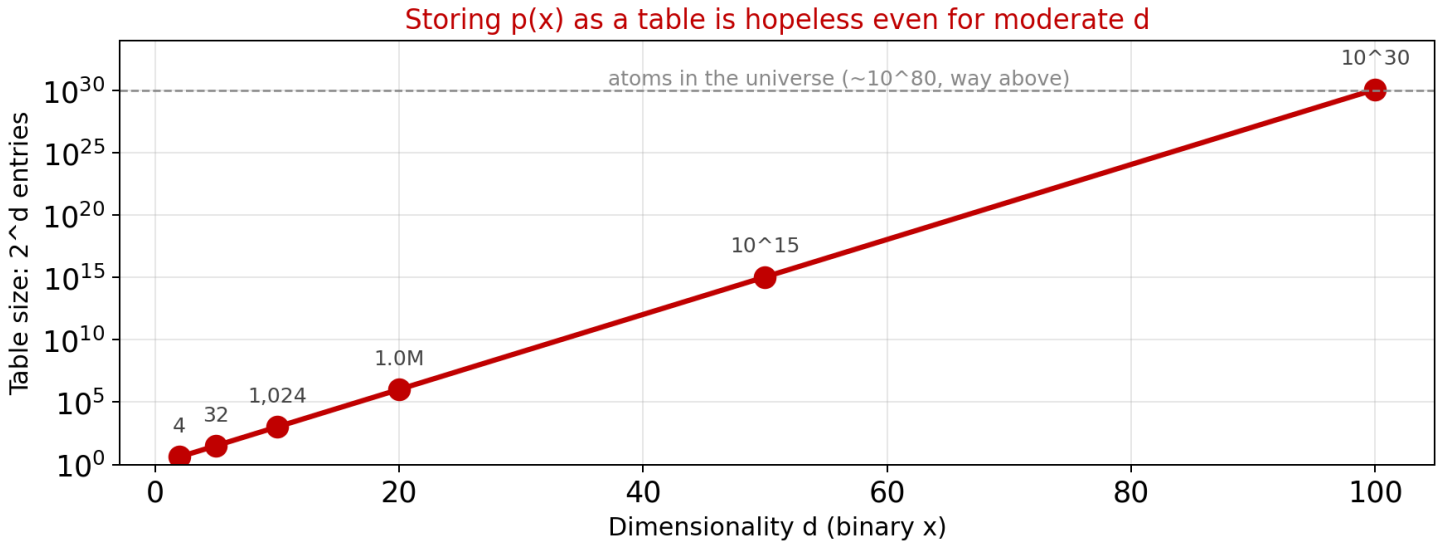
- Sampling: produce new x (required for generation)
- Density evaluation: compute p(x) or log p(x) (compression, anomaly detection, science)
- Latent representation: produce a useful z that summarizes x (downstream tasks)

2. Autoregressive Modeling

The Joint Distribution Problem

Suppose $x = (x_1, x_2, \ldots, x_d)$. We want $p(x)$.

- Naive table: if each x_i is binary, the table has 2^d entries.
- **For d = 100, that's 1.3×10^{30}.**



We can't store p(x), can't learn it, can't sample from it directly.

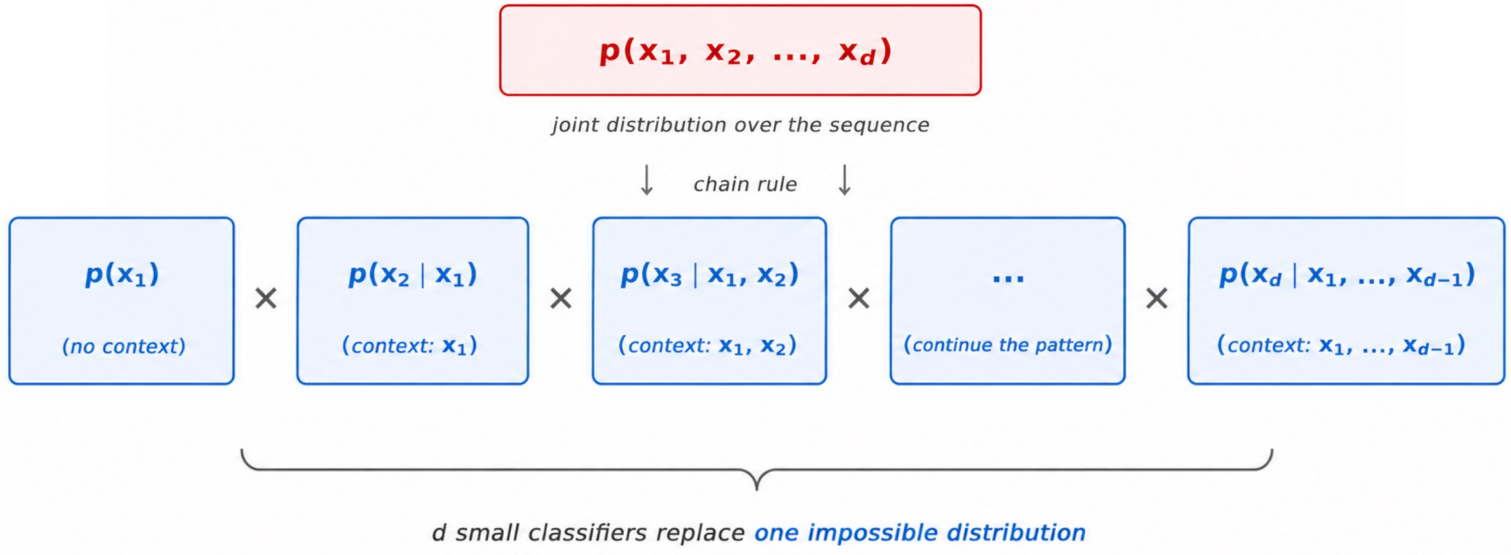
The Chain Rule of Probability

The chain rule is exact, no assumptions:

$$p(x_1, x_2, \ldots, x_d) = \prod_i p(x_i \mid x_1, \ldots, x_{i-1})$$

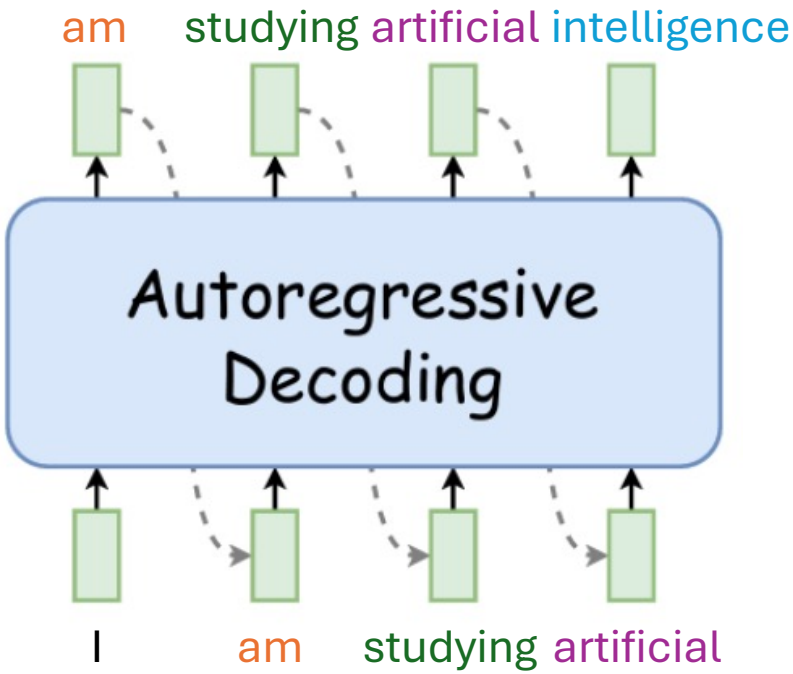
- We've reduced one impossible distribution to d conditional distributions.
- Each conditional is a much easier object: distribution over a single dimension, given a prefix.

This is the central trick of autoregressive modeling.



From Chain Rule to Autoregressive Model

- Replace each $p(x_i \mid x_{<i})$ with a parameterized model $p_\theta(x_i \mid x_{<i})$.
- One model, shared across positions, that takes a prefix and outputs a distribution over the next element.



- **Training objective: maximize log-likelihood**

$$\mathcal{L}(\theta) = \sum_i \log p_\theta(x_i \mid x_{<i})$$

🧠 Think: Why Is This Tractable?

- Discriminative models output a distribution over y, which is small (e.g., 1000 classes).
- Autoregressive output: distribution over a single dimension x_i, also small (50,000 vocab tokens, or 256 pixel values).

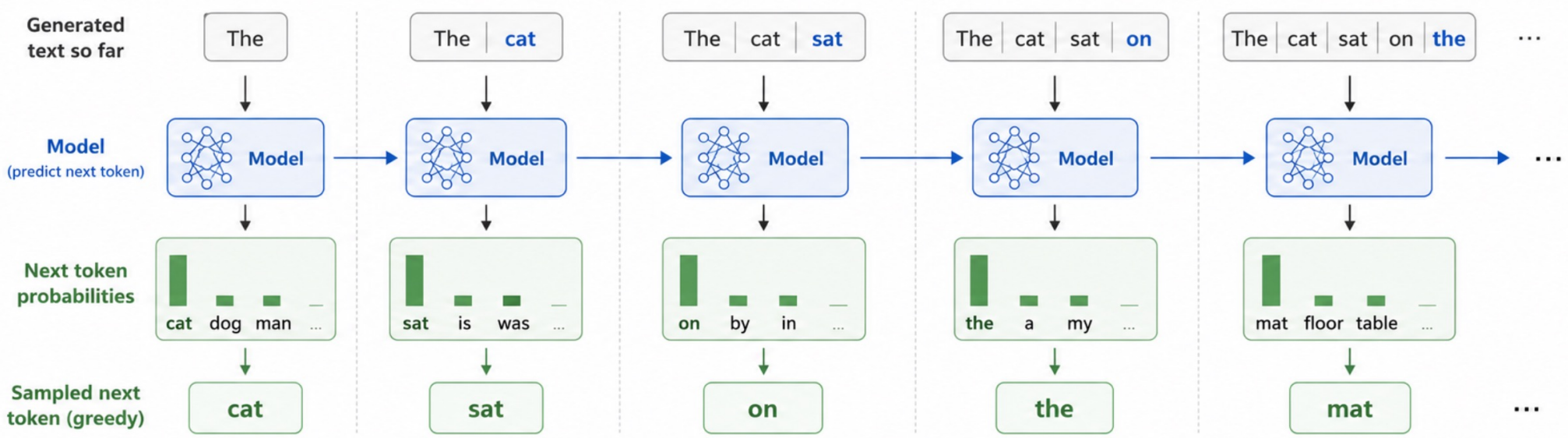
- The high-dimensional joint p(x) was intractable.
- But each factor $p(x_i \mid x_{<i})$ is just a classifier!

Generative modeling reduced to d classification problems.

3. Language Modeling

Text Is Already Sequential

- Text decomposes naturally into tokens: $t_1, t_2, \ldots, t_T$.
- A language model is exactly $p(t_1, t_2, \ldots, t_T)$ — a probability distribution over sequences.



Joint probability factorization (chain rule):

$$p(\text{The, cat, sat, on, the, mat}) = p(\text{The}) \cdot p(\text{cat}|\text{The}) \cdot p(\text{sat}|\text{The cat}) \cdot p(\text{on}|\text{The cat sat}) \cdot p(\text{the}|\text{The cat sat on}) \cdot p(\text{mat}|\text{The cat sat on the})$$

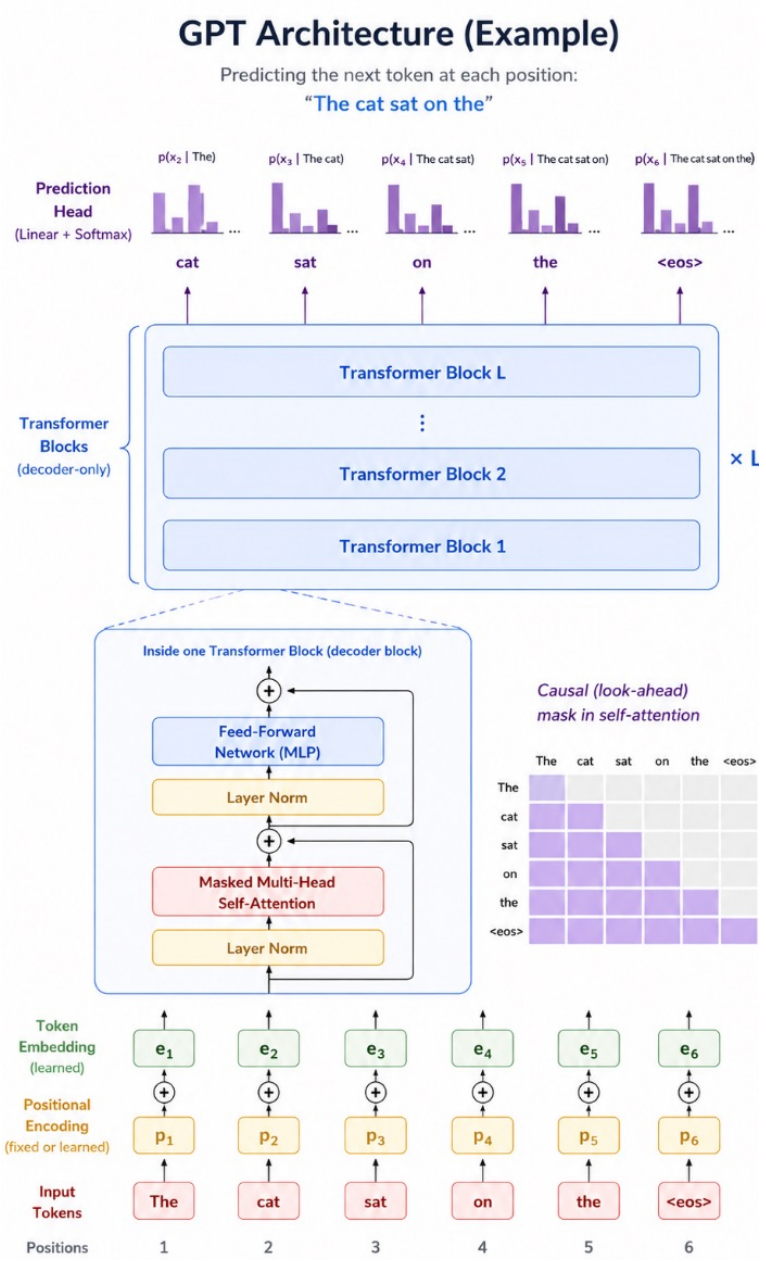
GPT

- **Architecture:**
  - stack of decoder-only Transformer blocks
  - causal self-attention (lower-triangular mask)

- **Input:** token sequence + positional encoding
- **Output:** distribution over next token (softmax over vocab)

- **Training:**
  - maximize log-likelihood over a giant corpus

- **Sampling:**
  - feed prefix → sample t_i → append → repeat



[1] Improving language under- standing by generative pre-training. OpenAI Blog. Radford et al.
[2] Language Models are Few-Shot Learners. arXiv 2005.14165. Brown et al.

Sampling Strategies

Once trained, $p_\theta(t_i \mid t_{<i})$ gives a distribution.

How do we draw from it?

- Greedy: argmax. Deterministic, often boring/repetitive.
- Temperature sampling: $p_\tau(t) \propto p(t)^{(1/\tau)}$. $\tau \to 0$ = greedy, $\tau \to \infty$ = uniform.
- Top-k: restrict to k most likely tokens, renormalize.
- Top-p (nucleus): smallest set with cumulative prob ≥ p, renormalize.

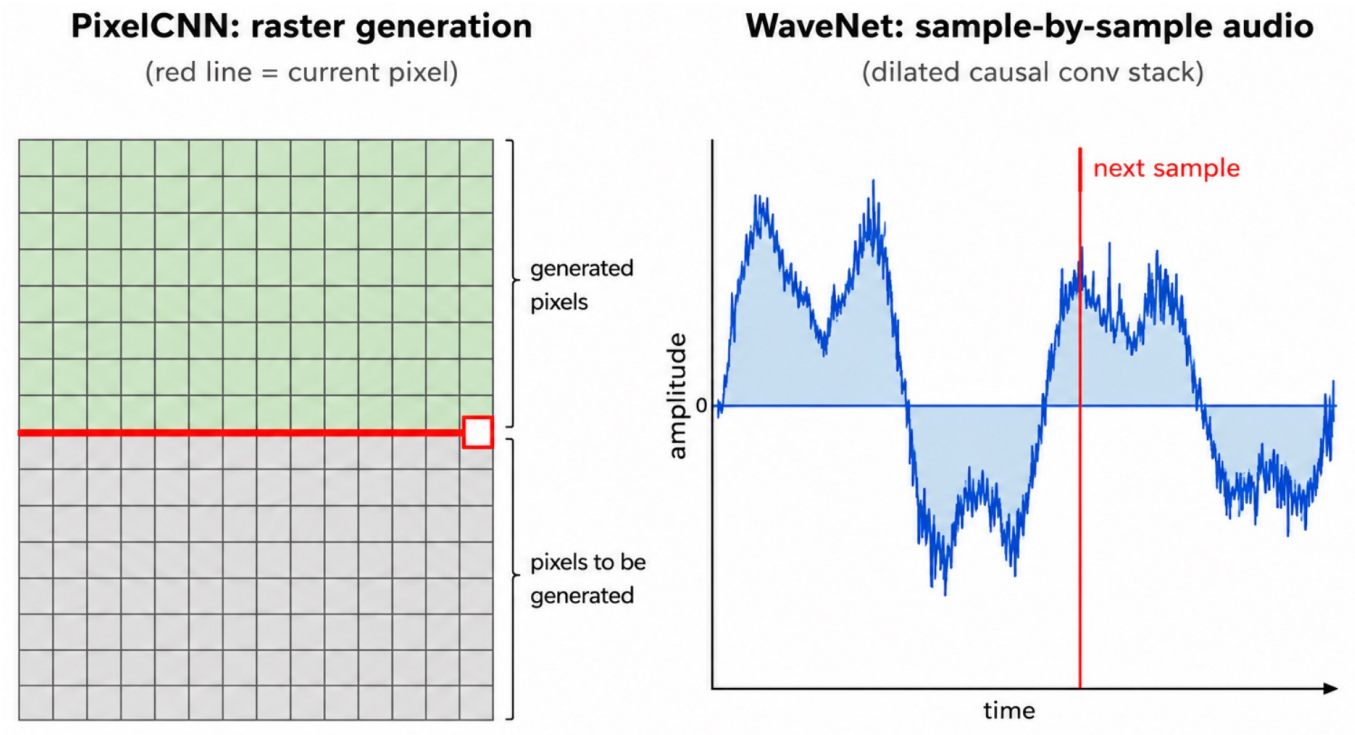
How Do We Evaluate a Language Model?

- Held-out log-likelihood: how probable is unseen text under the model?
- Perplexity: $\mathrm{PP} = \exp\left(-\frac{1}{T}\sum \log p(t_i \mid t_{<i})\right)$ — geometric mean inverse probability per token.
- *Lower perplexity = better. PP = 50 means "model averages 50 equally-likely options per token."*

Brief Cameos: Beyond Text

The autoregressive recipe isn't just for text. Same factorization, different domain:

- PixelRNN / PixelCNN: image factored pixel-by-pixel in raster order
- WaveNet: audio waveform sample-by-sample (dilated causal conv)



4. The Limits of Autoregressive Modeling

Strengths

✓ Exact likelihood.

- No bound, no approximation. p(x) is the model's output.

✓ Stable training.

- Just cross-entropy. Same machinery you know from L7–L22. No adversarial dynamics, no posterior collapse.

✓ Universal.

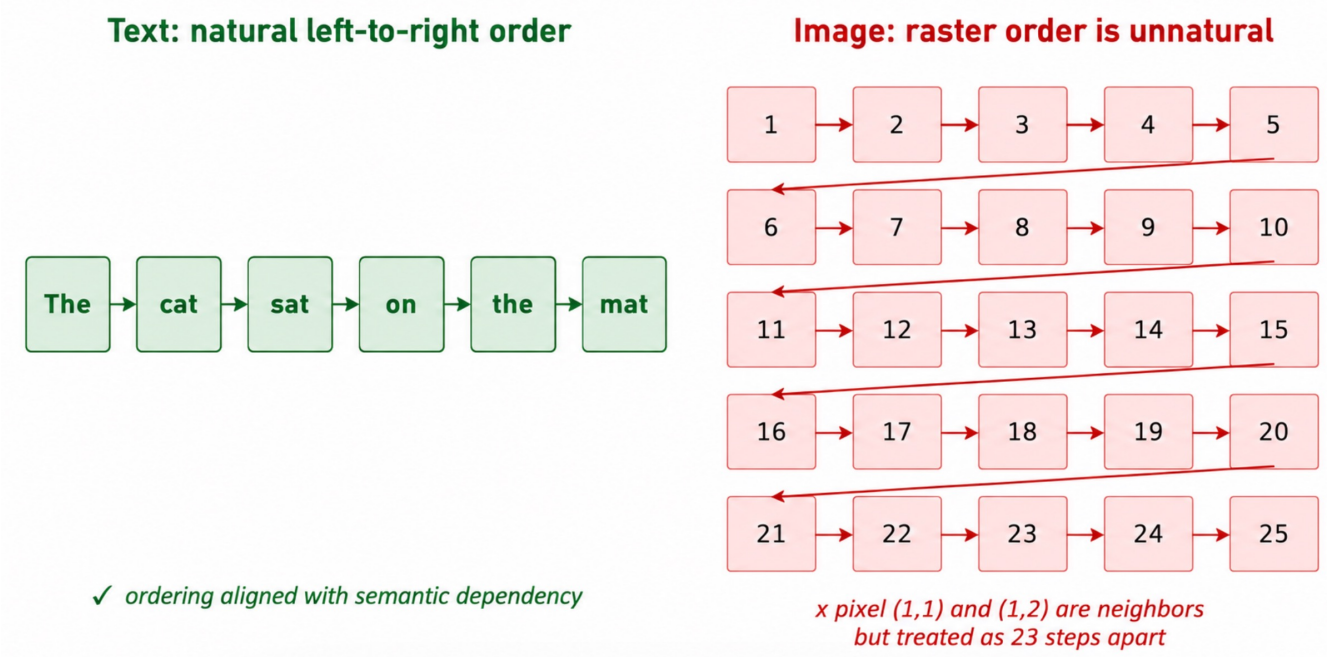
- Any data with a natural ordering admits an autoregressive factorization.

✓ Scales beautifully.

- GPT-2 → GPT-3 → GPT-4 → … the recipe doesn't break under scaling.

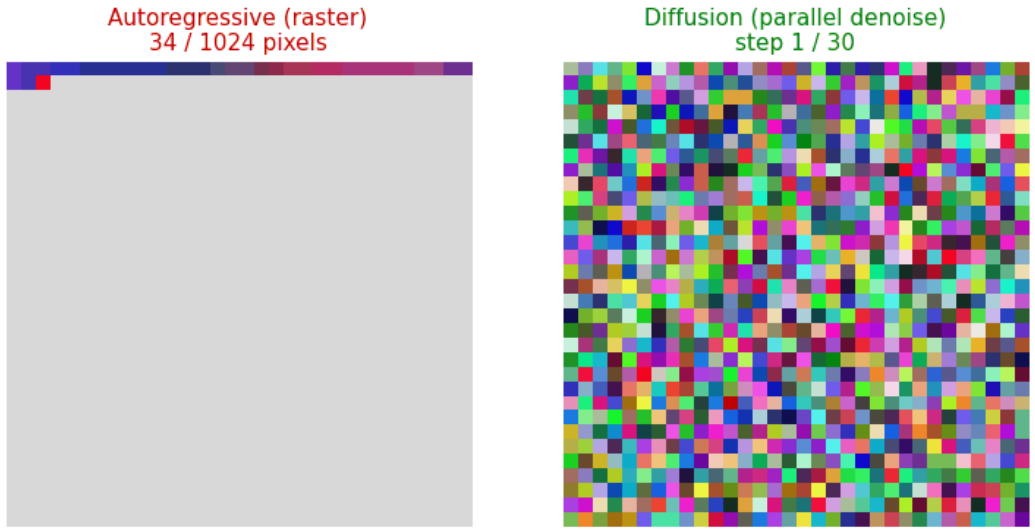
Weakness 1: The Ordering Problem

- The model you fit depends on the ordering.
  - Text: left-to-right is natural (causal).
  - Images: raster order is unnatural — pixel (1,1) and (1,2) are more related than pixel (1,1) and (224,224), but raster ordering treats them differently.
- **Implication:** autoregressive image models lag behind diffusion models on natural images. The ordering isn't aligned with the data's geometry.



Weakness 2: Slow Sampling

- Training is parallel (teacher forcing — all positions computed at once).
- **Sampling is inherently sequential: must produce x_1 before x_2.**
  - 1024-token sequence → 1024 forward passes
  - 1024 × 1024 image → ~10^6 forward passes → hours per image
- Diffusion (next week): same 1024 × 1024 image in 20–50 forward passes, in parallel across pixels → seconds



Weakness 3: No Useful Latent Representation

- The “representation” of x in an autoregressive model is just the prefix $x_{<i}$ itself.
- **There’s no z that summarizes x globally.**
- For tasks like clustering, interpolation, controllable generation — you want a latent variable.

- Next lecture: Variational Autoencoders

Summary

- Generative modeling = modeling p(x) or p(x|y), not p(y|x).
- The chain rule turns the joint into a sequence of conditionals → autoregressive modeling.
- L22's Transformer + chain rule = GPT.
- Strengths: exact likelihood, stable training, universal.
- Weaknesses: ordering bias, slow sampling, no latent.