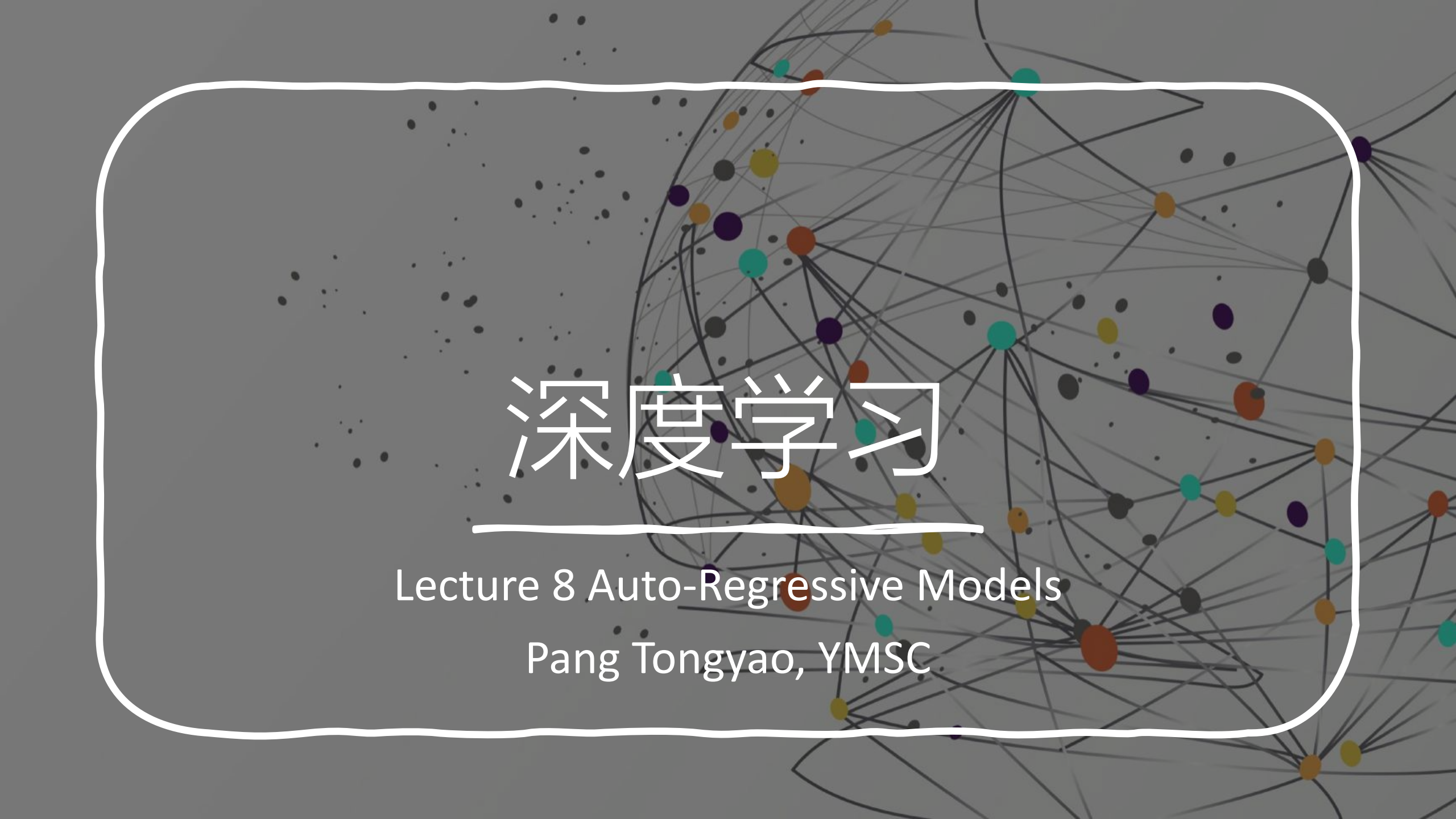


深度学习

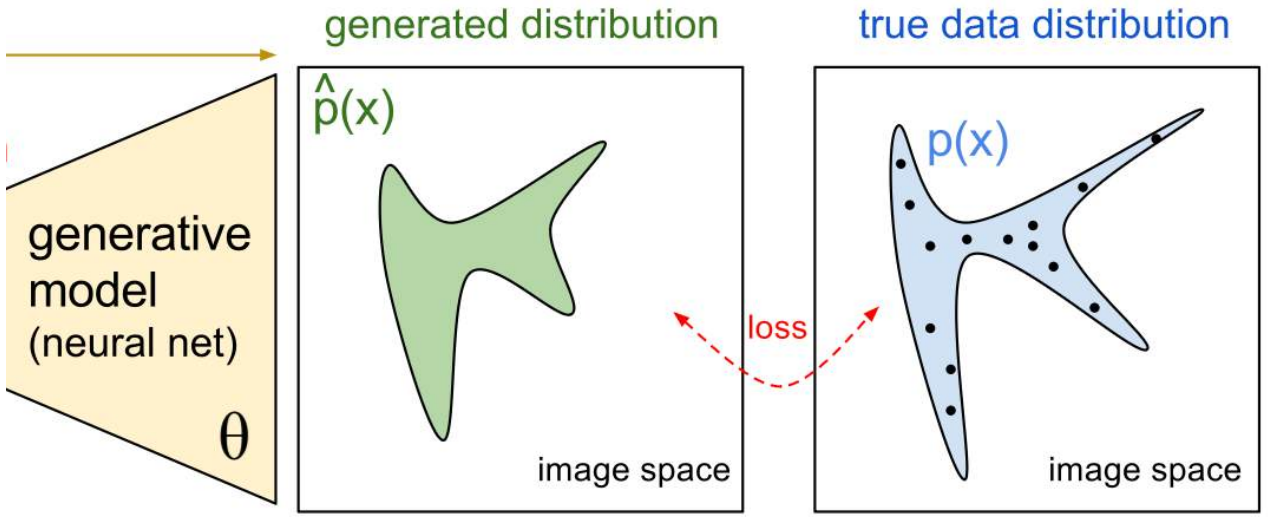
Lecture 8 Auto-Regressive Models

Pang Tongyao, YMSC

Generative Models

Generative Models

- Generative Models: Given some data, train a model to generate data like it!



"What I cannot create, I do not understand." ----**Richard Feynman**

Reference: https://openai.com/research/generative-models

Generative Models

Why generative models:

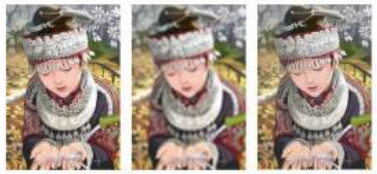
- The discriminative model is to approximate the MMSE Estimator (i.e., posterior mean):

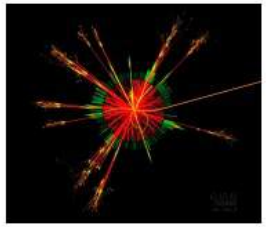
$$x_0^* = \arg \min_{g(y_0):\, g \in \{f_\theta\}} \mathbb{E}_{(x_0, y_0) \sim p_{data}} ||g(y_0) - x_0||_2^2 \approx \mathbb{E}(x|y)$$

- Generative models learn $p(x)$. For any task, knowing the conditional distribution $p(y|x)$ allows us to obtain the posterior distribution $p(x|y)$ through Bayes' theorem: $p(x|y) = \frac{p(x)p(y|x)}{p(y)}$.
- Generative models can learn the intrinsic characteristics of the data.

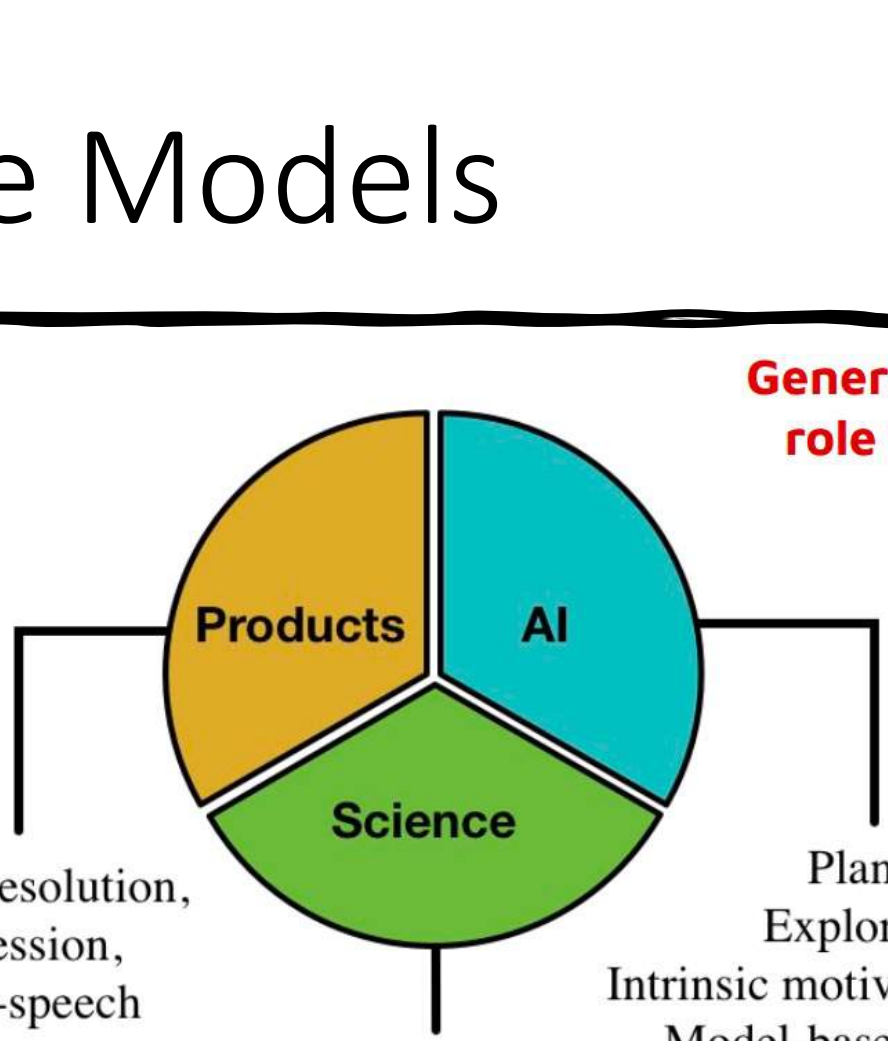
Generative Models

Generative models have a role in many problems.





Super-resolution,
Compression,
Text-to-speech



Proteomics,
Drug Discovery,
Astronomy,
High-energy physics

Planning,
Exploration
Intrinsic motivation
Model-based RL

Image Credit: Rezende and Mohamed, DeepMind

Generative Models

- Target: p_{model} is close to p_{data}
- KL divergence measures the similarities of distributions

$$KL(p_{data}||p_{model}) = \int p_{data} log p_{model} - \int p_{data} log p_{data}$$

$$\approx \sum_{x \in D} \log p_{model}(x|\theta)$$

loss: likelihood

constant

Key: how to model and compute p_{model}?

Autoregressive Models

Autoregressive Models

- Data in generation tasks: long text, high resolution images, codes et al.
- Model the data distribution $p(\boldsymbol{x})$ ($\boldsymbol{x} \in \mathbb{R}^N$) in an autoregressive way:

$$p_{model}(\boldsymbol{x}) = \prod_{i=1}^{N} p_{model}(x_i | x_1, \ldots, x_{i-1})$$

  - Autoregression is a way of modeling joint distribution by a product of conditional distributions.
  - This formulation is always valid w/o compromise or approximation.

Inductive Bias

- Generally, the distributions $p(x_i|x_1, \dots, x_{i-1})$ are different for different i.
- Representing them using similar architectures helps reducing model size and improving generalization.
  - e.g. model them with shared architectures and weights (like what we did in CNN, RNN and transformers): $p_\theta(x_i|x_1, \dots, x_{i-1})$
  - In auto-regressive language models, $p_\theta(x_i|x_1, \dots, x_{i-1})$ is discrete. The network works like classification: predict the probabilities of each word in the vocabulary that appears as next word.

Autoregressive Inference

- **Auto:** using its "own" out puts as inputs for next perditions
- **Regression:** estimating relationship between variables

$$p_\theta(\boldsymbol{x}) = \prod_{i=1}^{N} p_\theta(x_i | x_1, \ldots, x_{i-1})$$

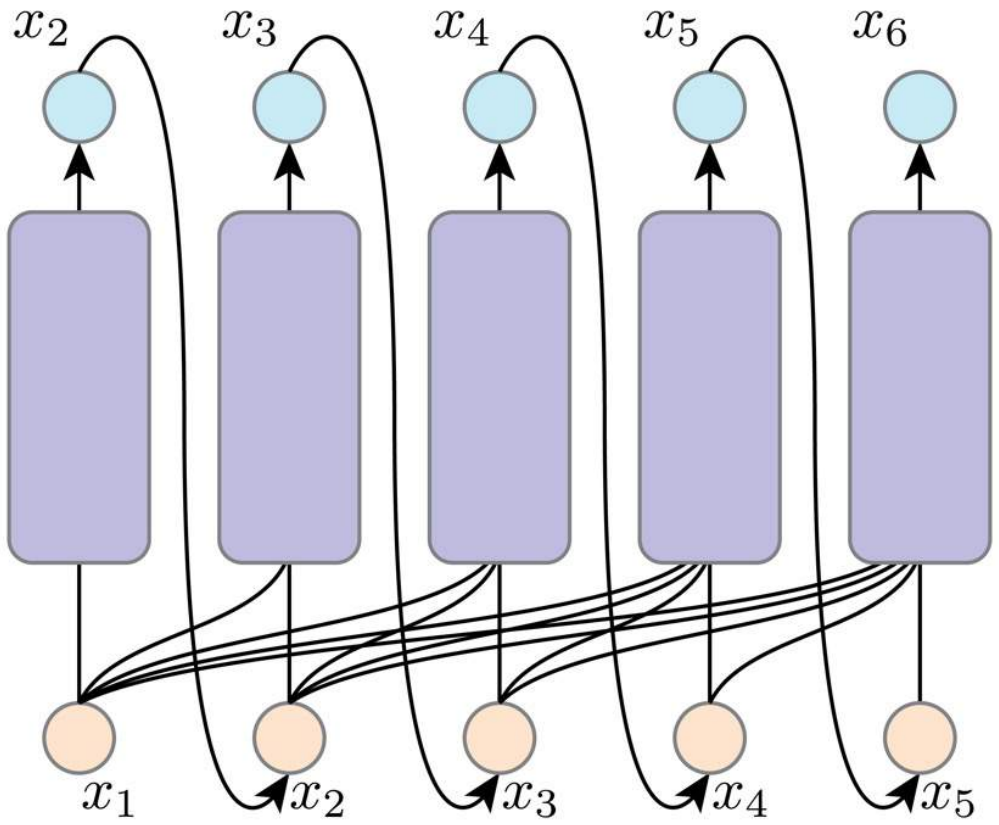


Image credit: Kaiming He

Autoregressive Inference

- This net models $p(x_2 \mid x_1)$
- 1 input
- 1 output

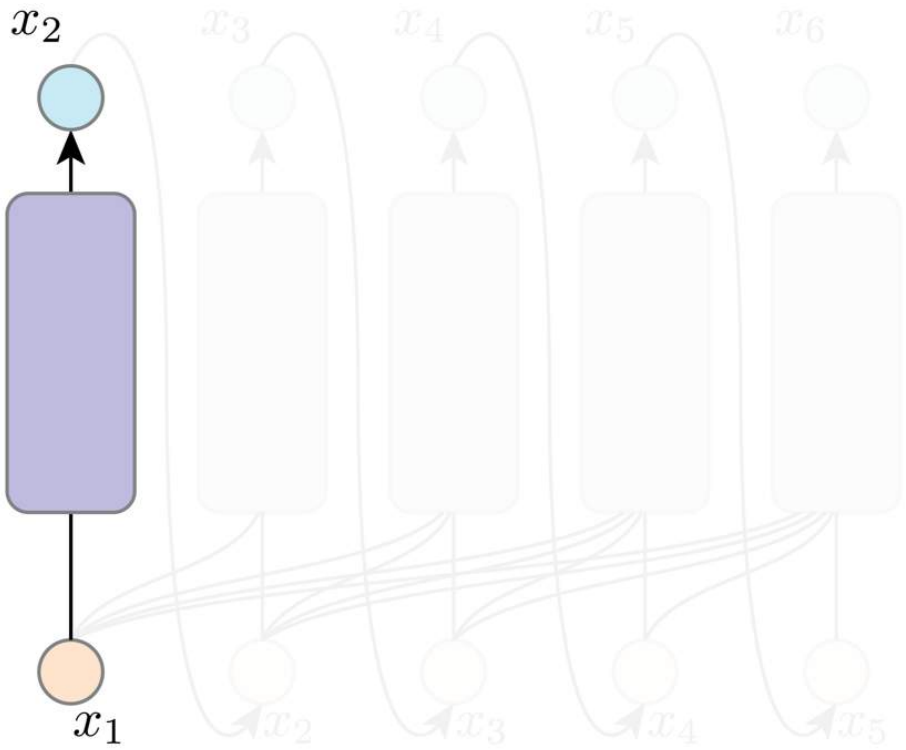


Image credit: Kaiming He

Autoregressive Inference

- This net models $p(x_3 \mid x_{1,2})$
- **2 inputs**
- 1 output

- inputs: outputs from previous steps

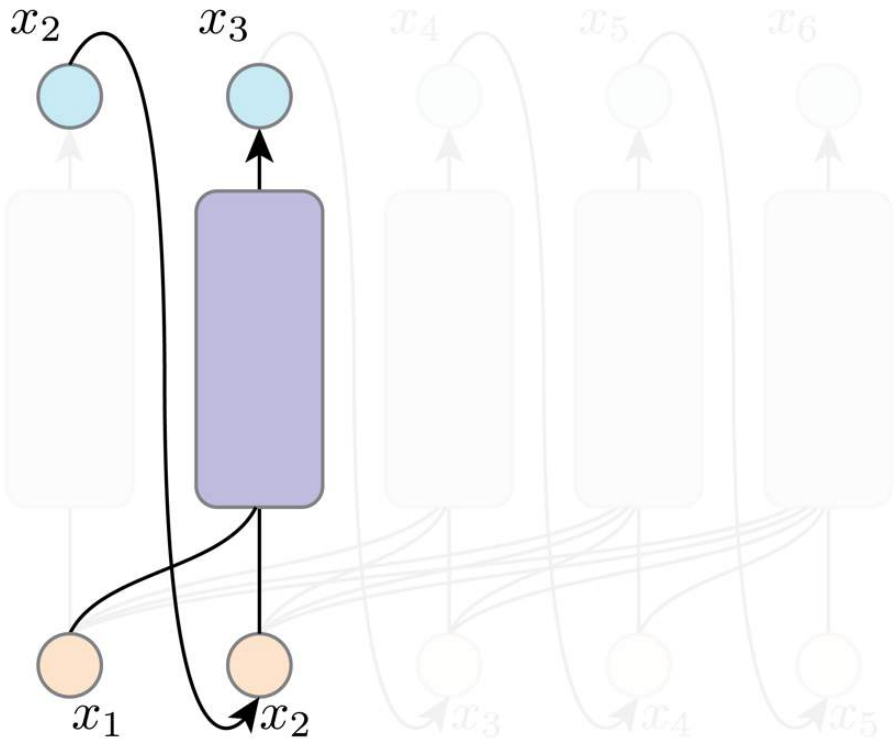


Image credit: Kaiming He

Autoregressive Inference

- This net models $p(x_6 \mid x_{1,2,3,4,5})$
- **5 inputs**
- 1 output

- inputs: outputs from previous steps

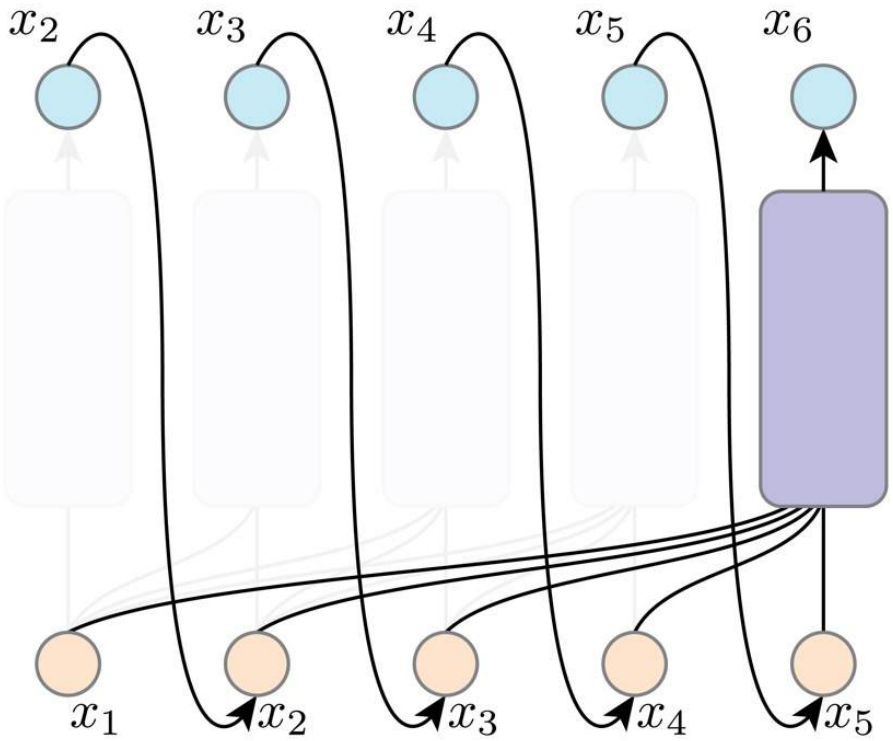


Image credit: Kaiming He

Training

- For training phase, performing the forward process in an auto-regressive way is impractical.
- Computing the gradient of x_6 should go through
  - all previous outputs
  - all previous sampling
  - all previous networks

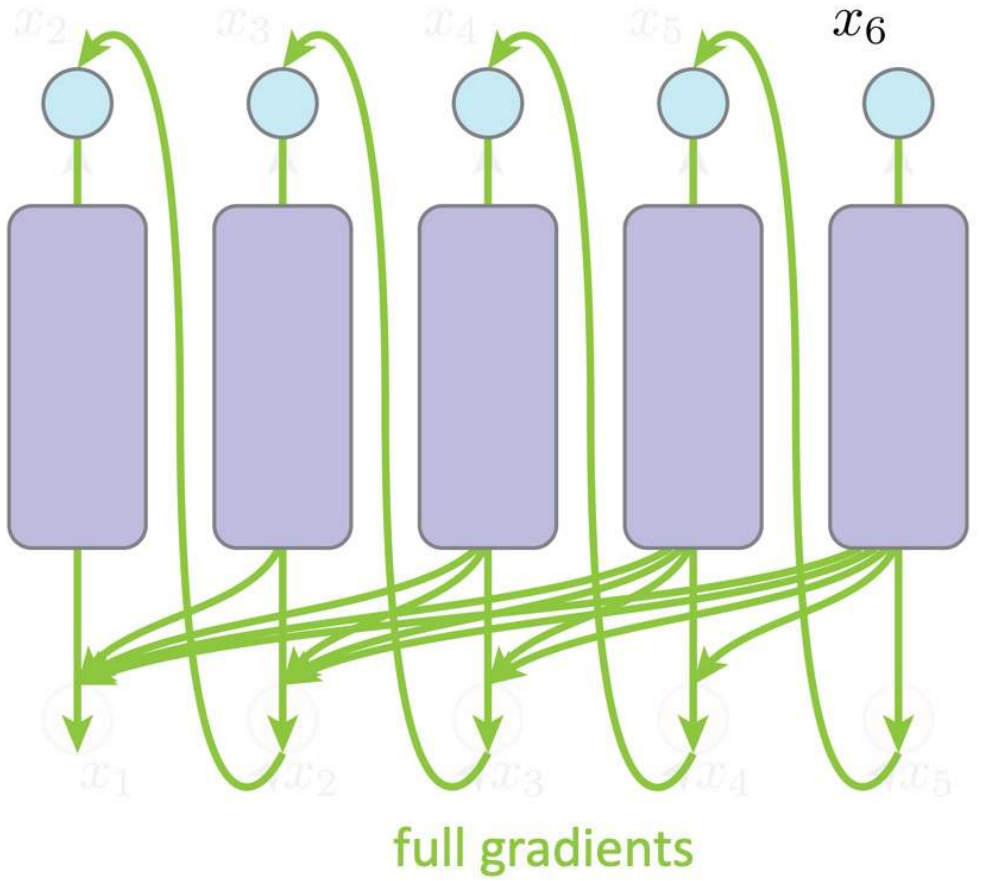


Image credit: Kaiming He

Teacher Forcing

- During **training**, we always feed the correct x_{t-1}, not the model's own prediction.
- Pros:
  - Allow parallel computation over all timesteps
  - Avoid error accumulation during training
  - Efficiently compute gradients
- Cons: train-test mismatch

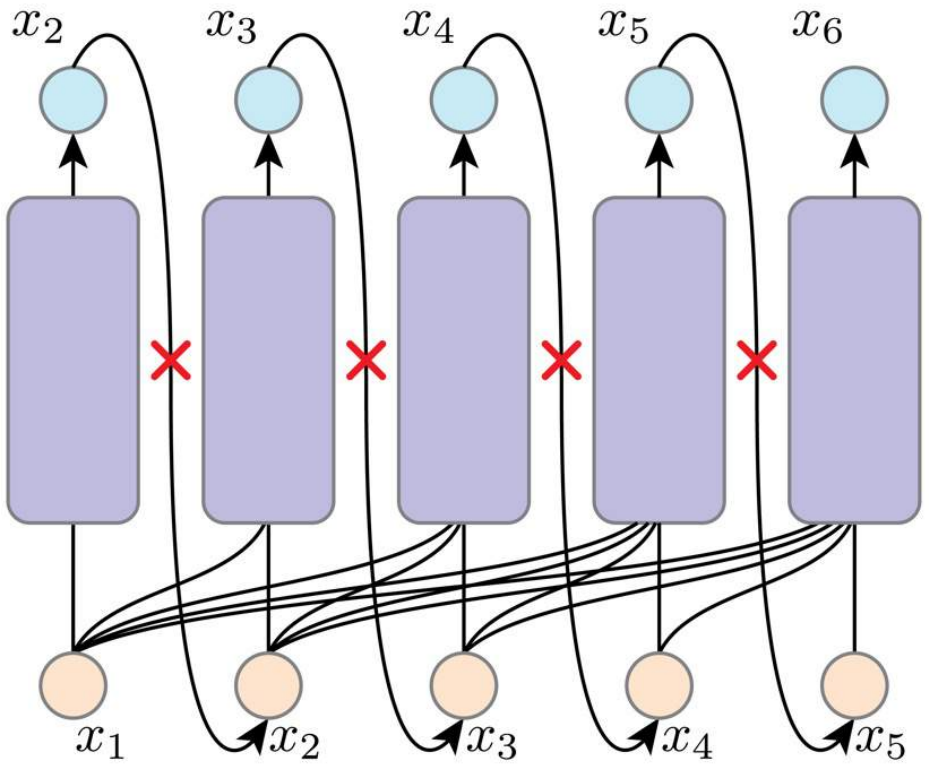
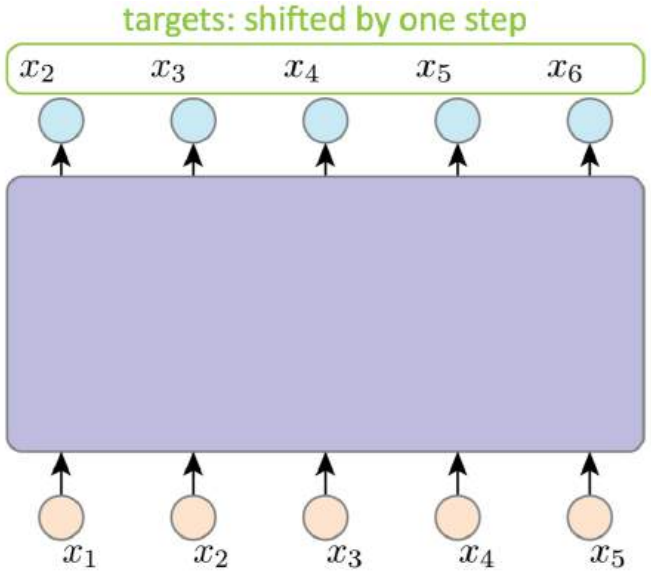
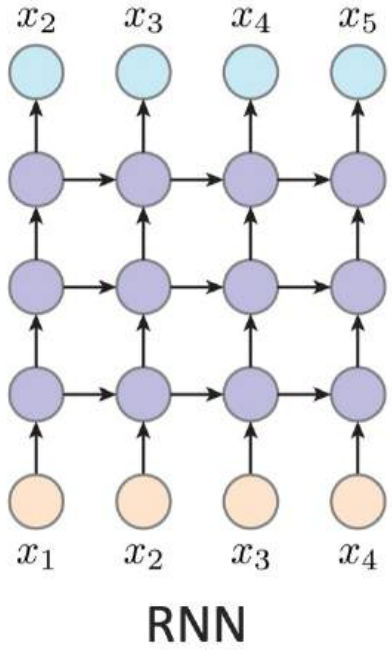


Image credit: Kaiming He

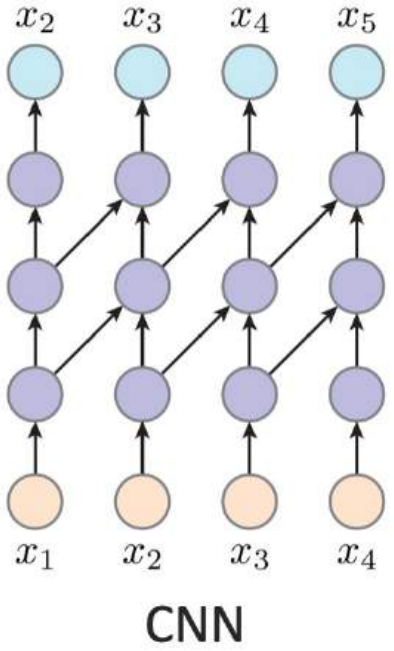
Architectures





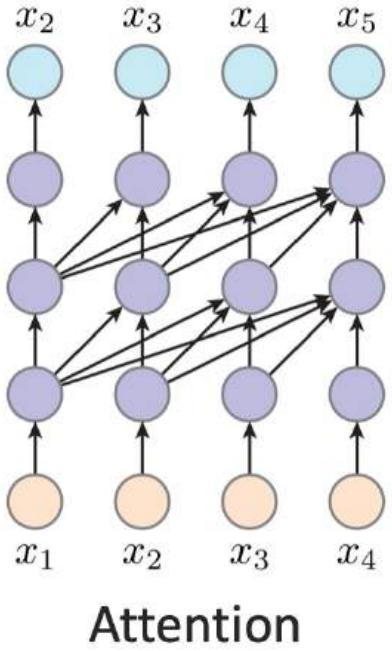
RNN

not parallelable



CNN

short context

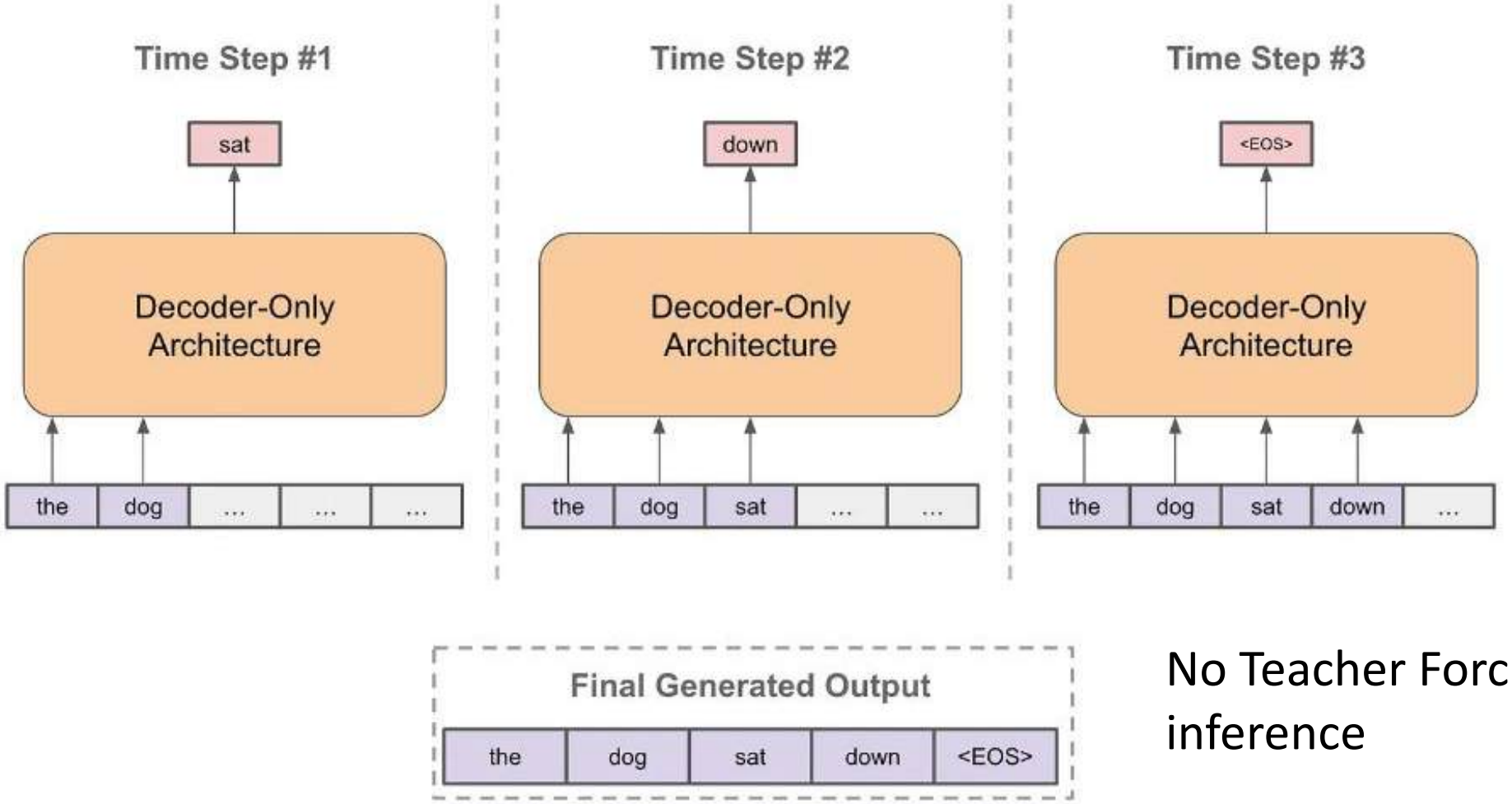


Attention

- parallelable
- long context

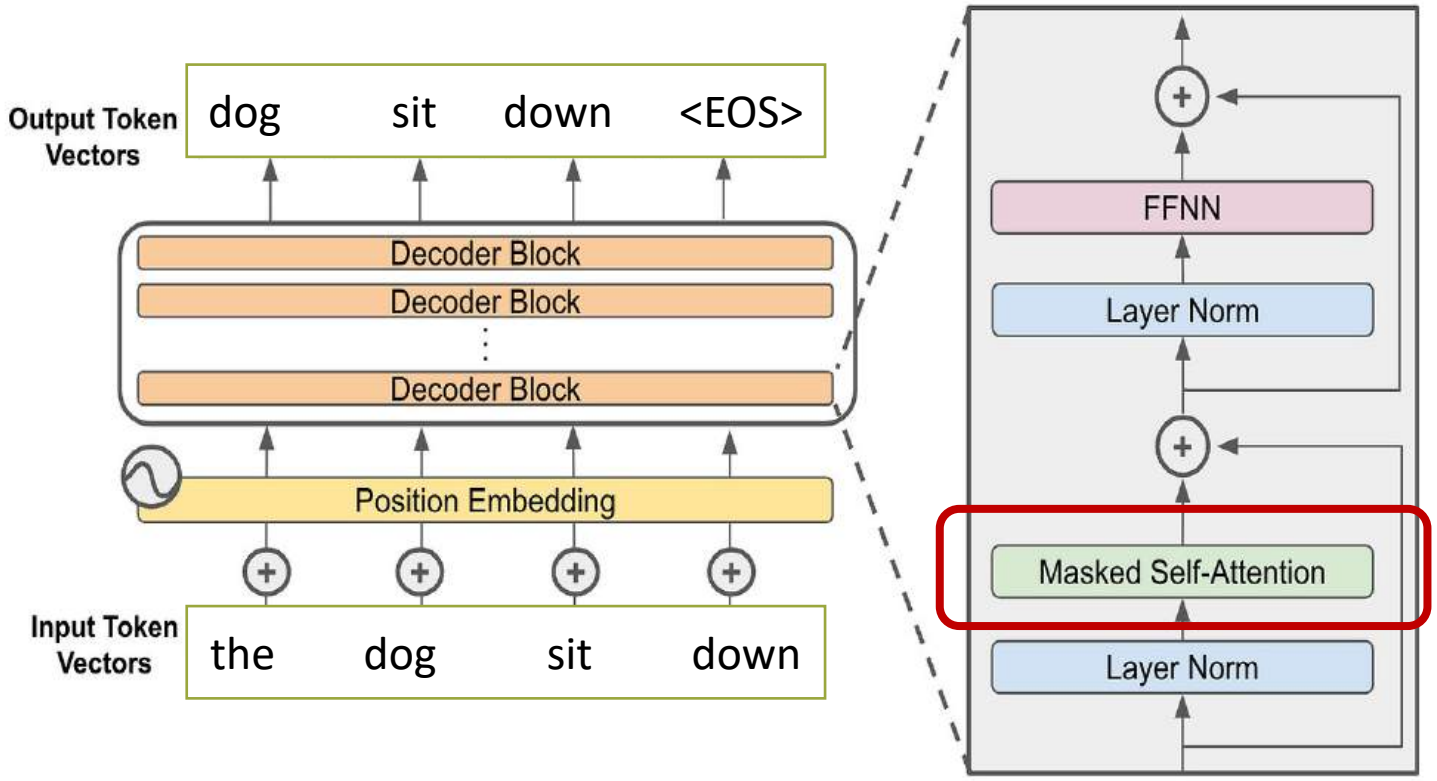
Image credit: Kaiming He

Autoregressive Generation



No Teacher Forcing during inference

Causal Self-Attention



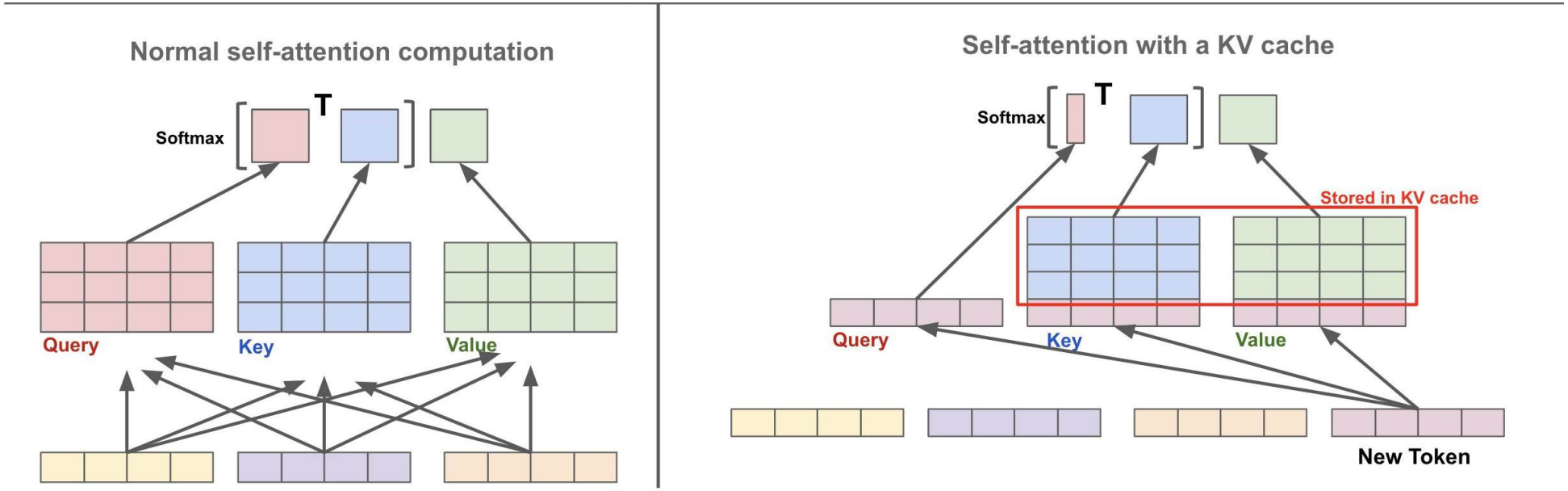
GPT: Generative Pre-Training

- **Causal Self-Attention**: Each token can only attend to itself and previous tokens — not any future tokens.

	X_1	X_2	X_3	X_4
X_1	✅			
X_2	✅	✅		
X_3	✅	✅	✅	
X_4	✅	✅	✅	✅

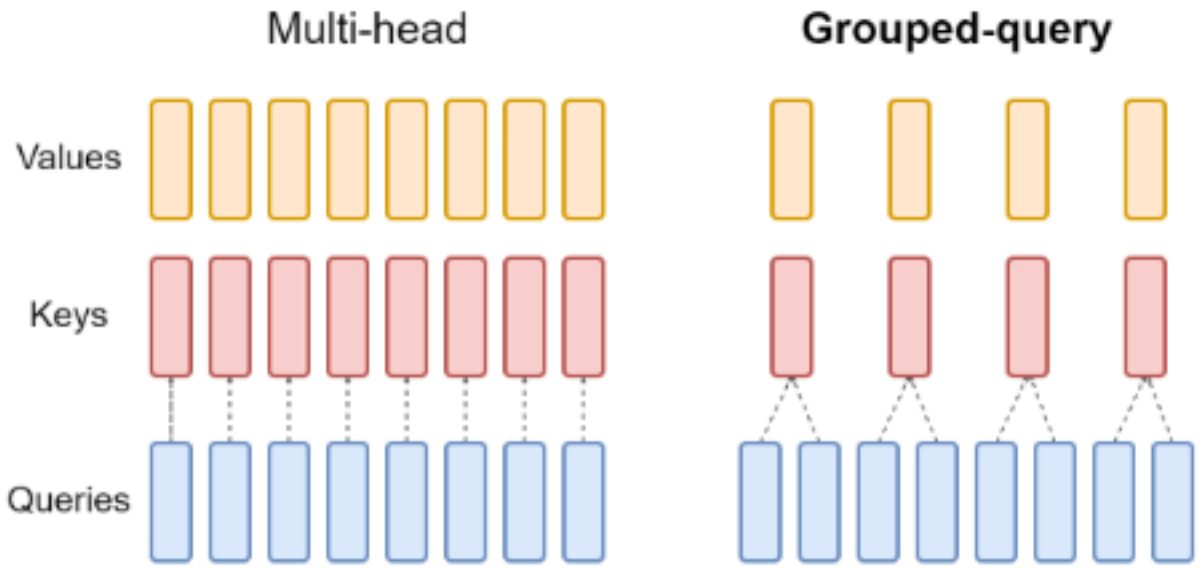
KV cache

- We only need to compute the representation for each new token, but other tokens remain fixed. KV cache size: about 2* head_dim * n_heads * n_layers * seq_length * batch_size



Grouped-query Attention

- For the query vectors, the original number of attention heads is maintained.
- The key and value vector pairs are then shared across multiple query heads.



"GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints"

Sliding Window Attention

- SWA makes the seq_length term a constant, allowing us to scale to larger context windows while keeping the KV cache size fixed.

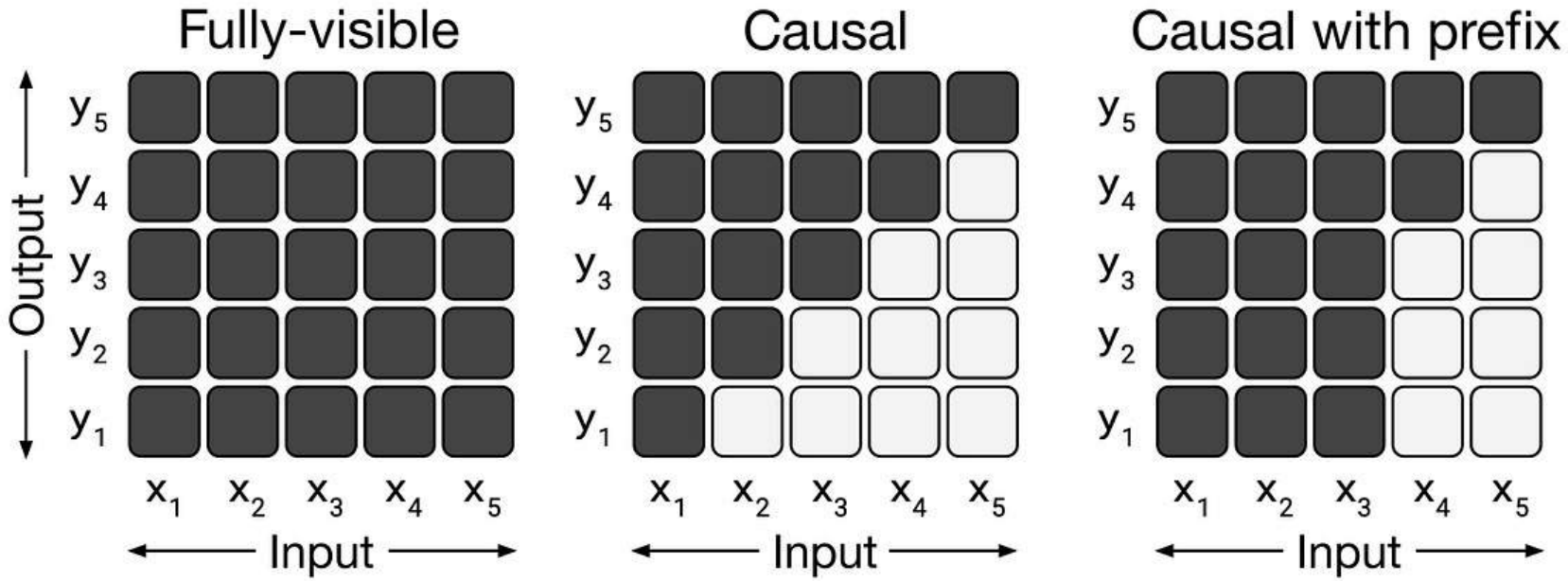
Causal Mask in Self-Attention

	The	baby	was	asleep	by	ten
The	1.00	-	-	-	-	-
baby	0.40	0.60	-	-	-	-
was	0.10	0.50	0.40	-	-	-
asleep	0.17	0.23	0.55	0.05	-	-
by	0.05	0.05	0.12	0.73	0.05	-
ten	0.04	0.08	0.03	0.15	0.17	0.53

Sliding Window Attention (w = 3)

	The	baby	was	asleep	by	ten
The	1.00	-	-	-	-	-
baby	0.40	0.60	-	-	-	-
was	0.10	0.50	0.40	-	-	-
asleep	-	0.23	0.55	0.05	-	-
by	-	-	0.12	0.73	0.05	-
ten	-	-	-	0.15	0.17	0.53

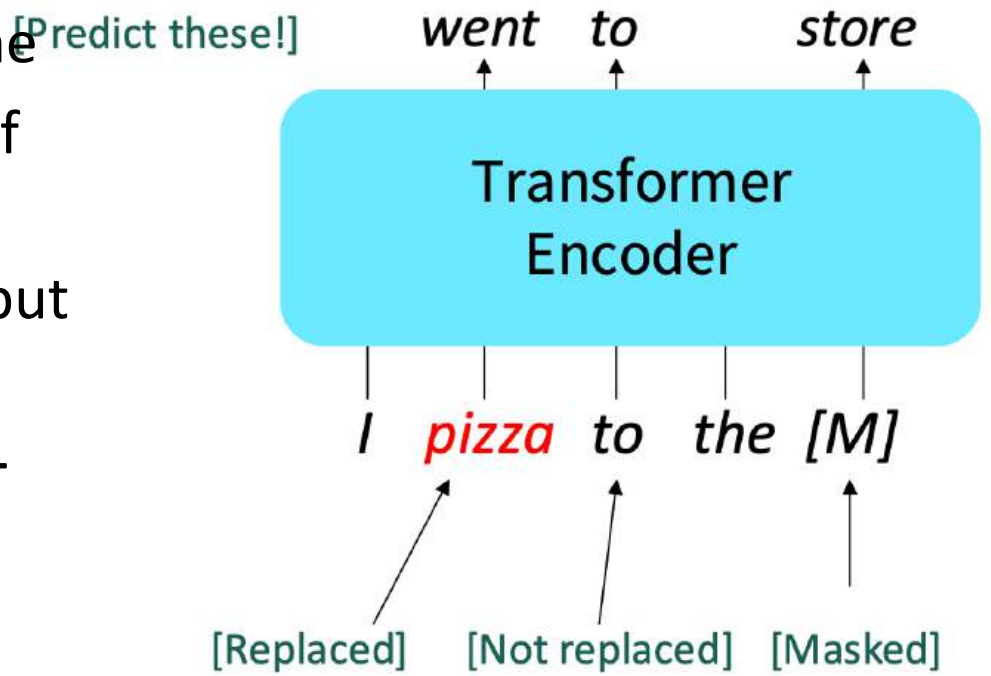
Attention Mask Patterns



- Encoder-only models leverage bidirectional (or fully-visible) self-attention, which considers all tokens within the entire sequence during self-attention.
- Decoder-only models use causal self-attention, where each token only considers tokens that come before it in the sequence.

BERT: Encoder-only

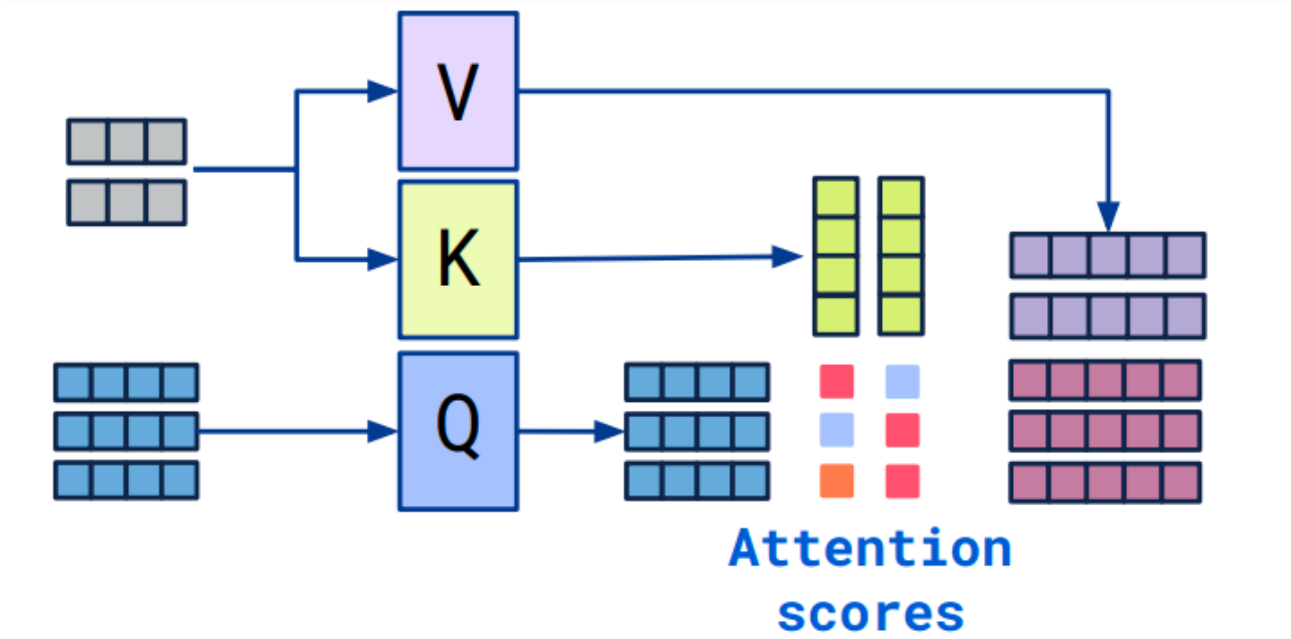
- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (but still predict it!)
- The performance of BERT is better than GPT on lots of tasks. But it is not suitable for generation!

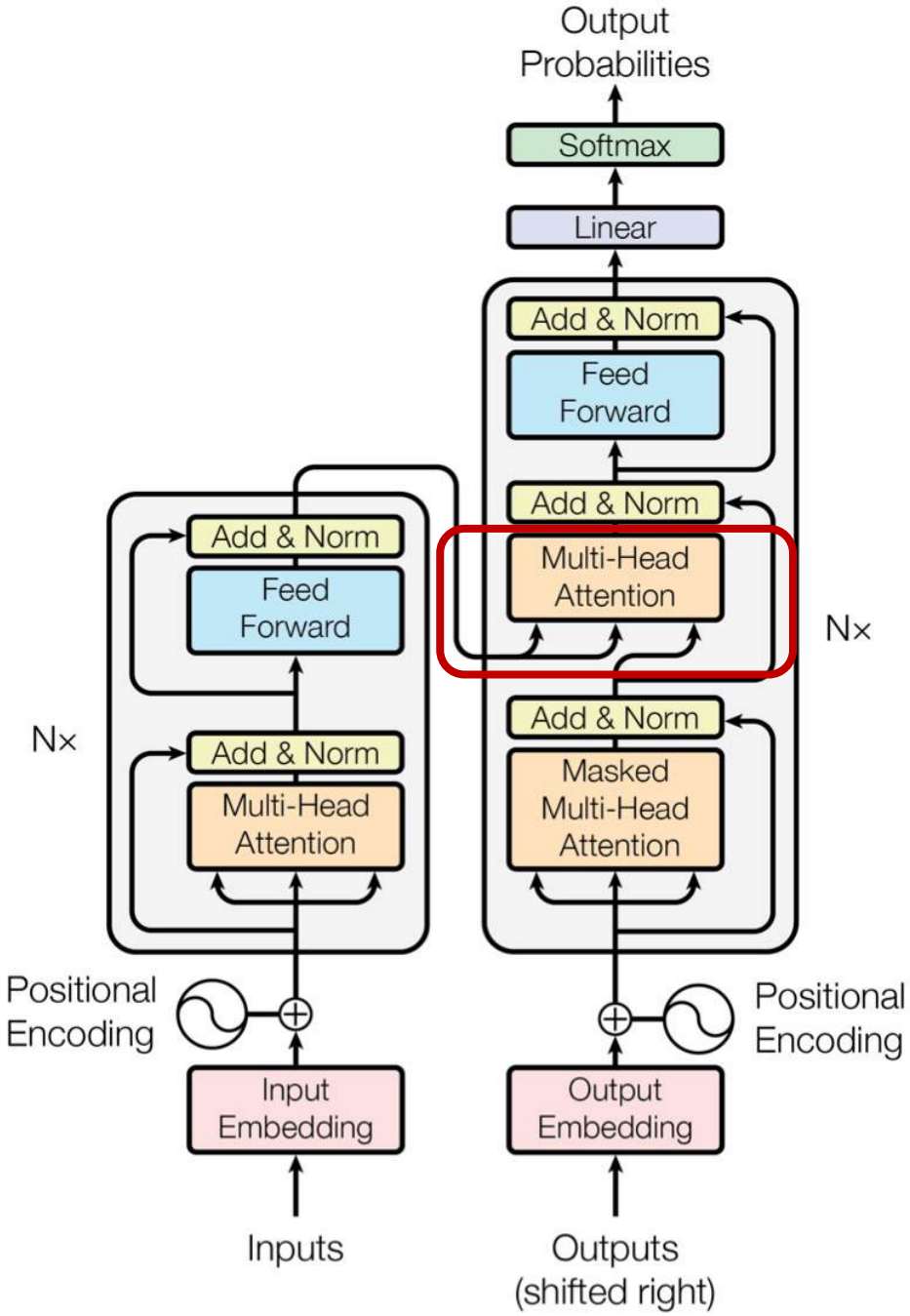


BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding ,2019

Encoder-decoder

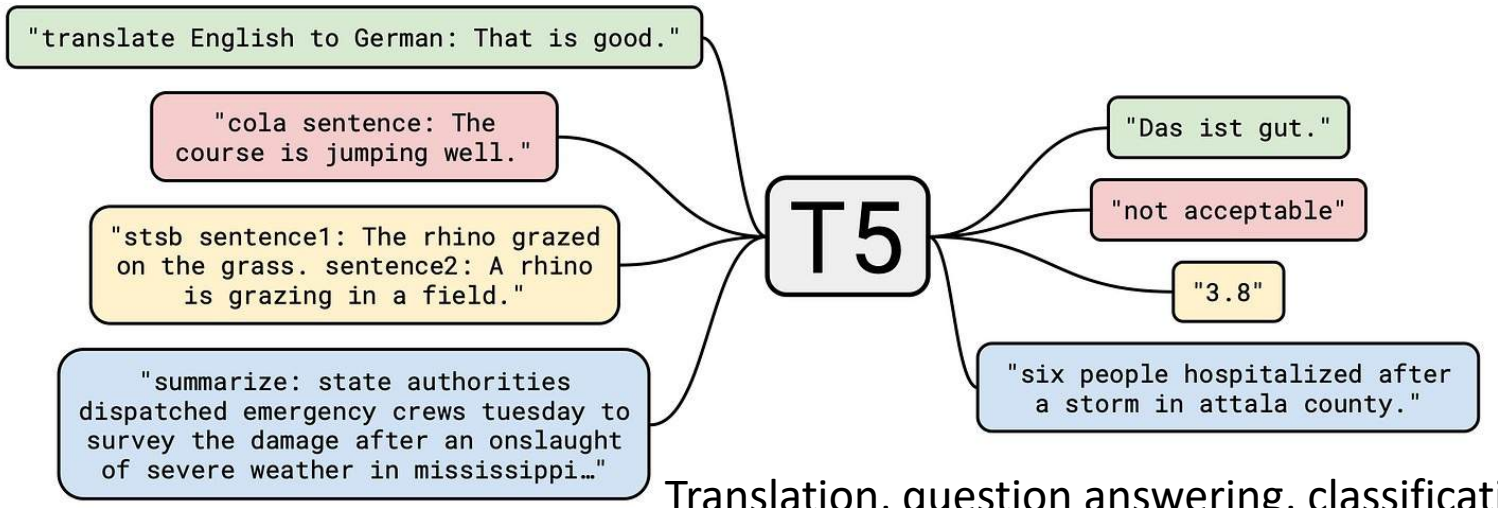
- Suitable for seq-to-seq tasks
- Cross attention: embed encoder outputs into decoder



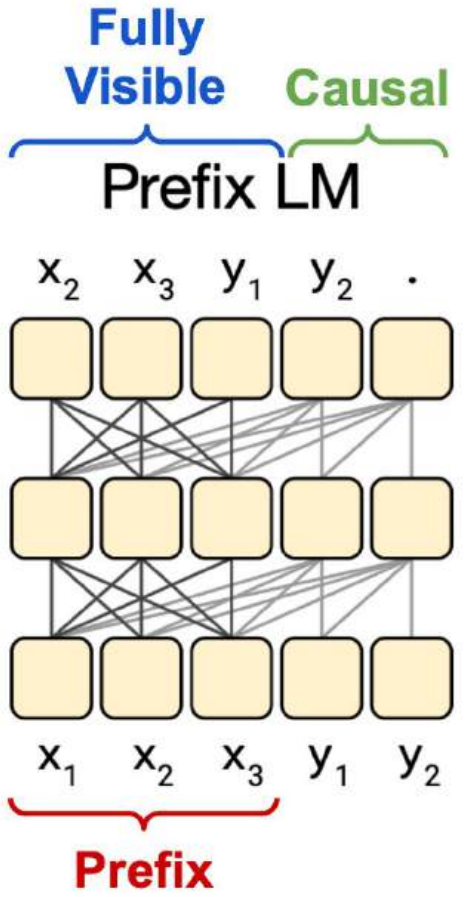


Text-to-Text Framework

- **T5 (The Unified Text-to-Text Transformer):** model and solve a variety of different tasks with a shared approach: *the same model, objective, training procedure and decoding process*
- Encoder–decoder models are excellent for conditional generation ,but less efficient for free-form generation.



Translation, question answering, classification….



Post-training

1. Supervised Fine-tuning (GPT1)
2. Emergent Zero-shot Learning via Prompting (GPT2)
3. In-context (Few-shot) Learning (GPT3)
4. Chain-of-thought Prompting
5. Reinforcement Learning from Human Feedback (covered in later class)

Supervised Fine Tuning (SFT)

- **Supervised fine-tuning:** continuing the training of a pretrained language model using **labeled data** for a **specific task.**
- Input–output pairs are often formatted as **prompts** and responses.
- Even a few thousand examples can make a big difference.

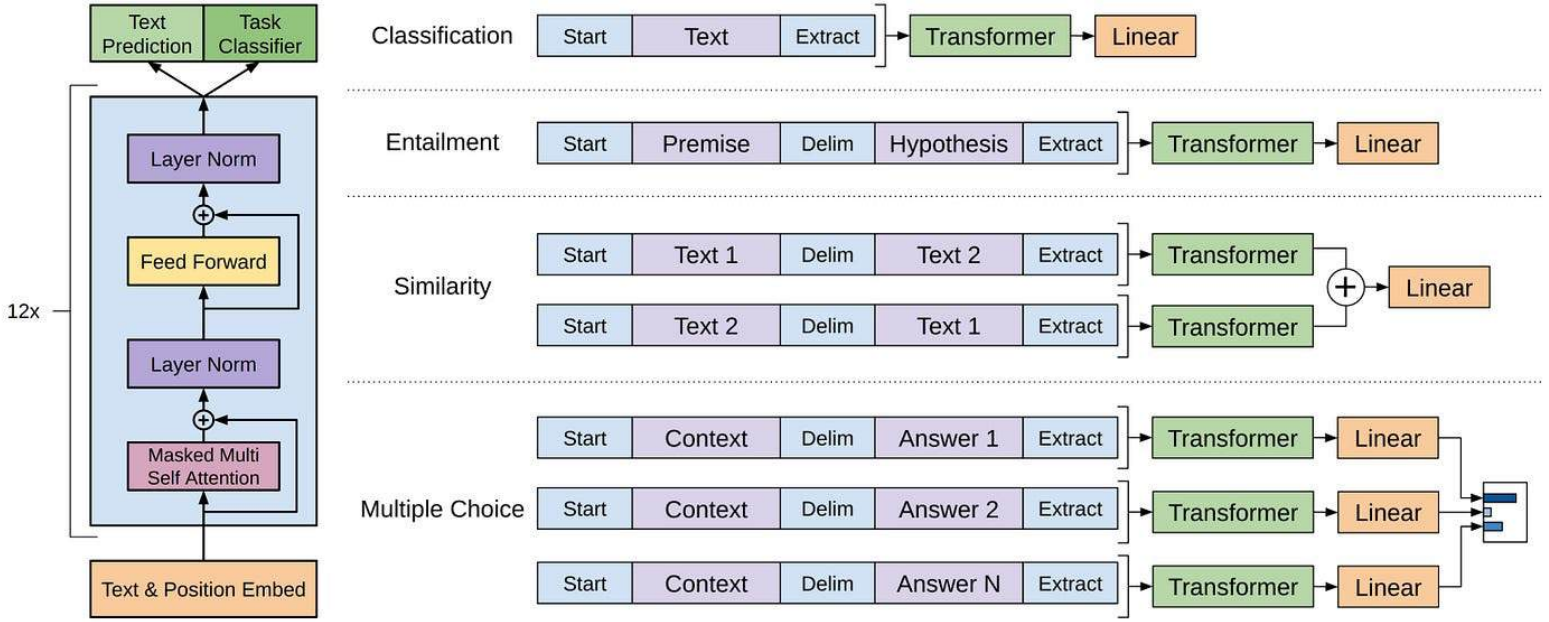


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

GPT: Improving Language Understanding by Generative Pre-Training, 2018

Zero-shot Learning

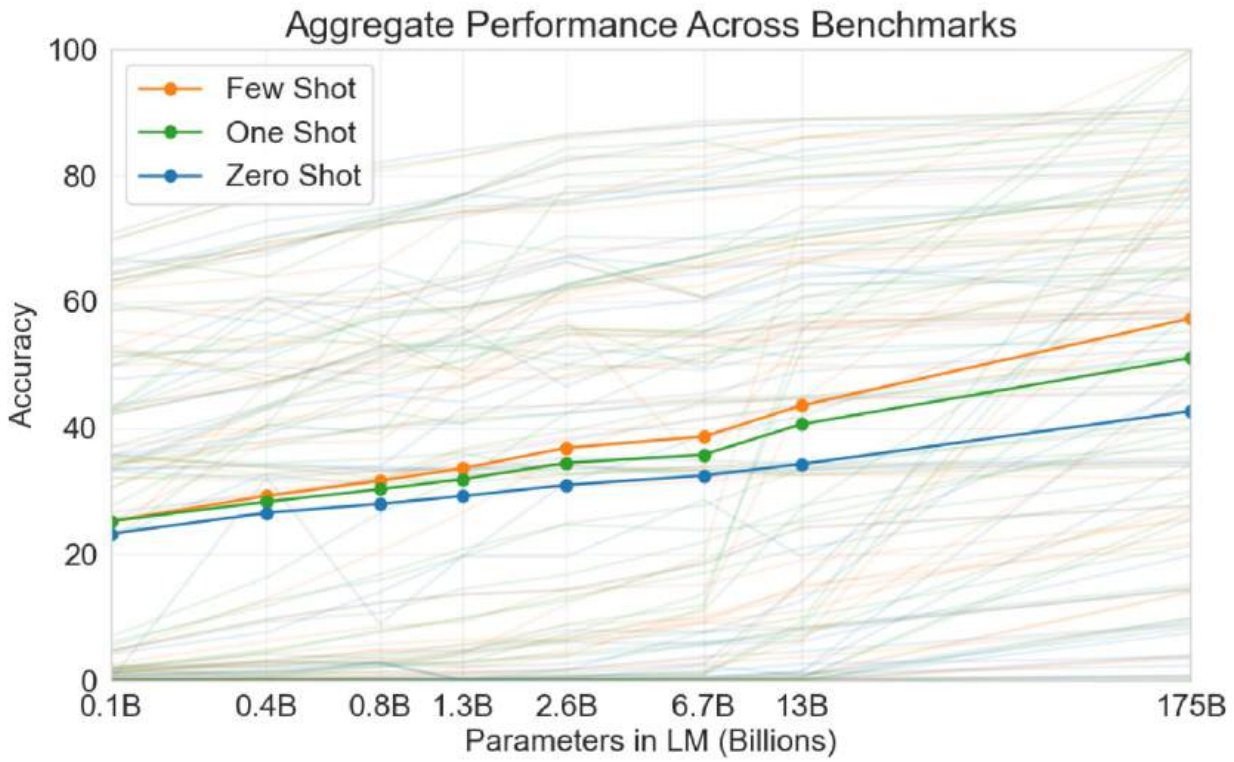
- GPT2: Language Models are Unsupervised Multitask Learners, 2019
  - Same Transformer decoder architecture as GPT-1, just larger 117 M → 1.5 B parameters
  - Trained on much more diverse web data: 4 GB → 40 GB of internet text (WebText)
- GPT-2 could **perform unseen tasks** directly from a prompt

Translate English to French: cat → ?

- GPT2 beats SOTA on some tasks without fine-tuning.

Few-shot Learning

- GPT3:Language Models are Few-Shot Learners



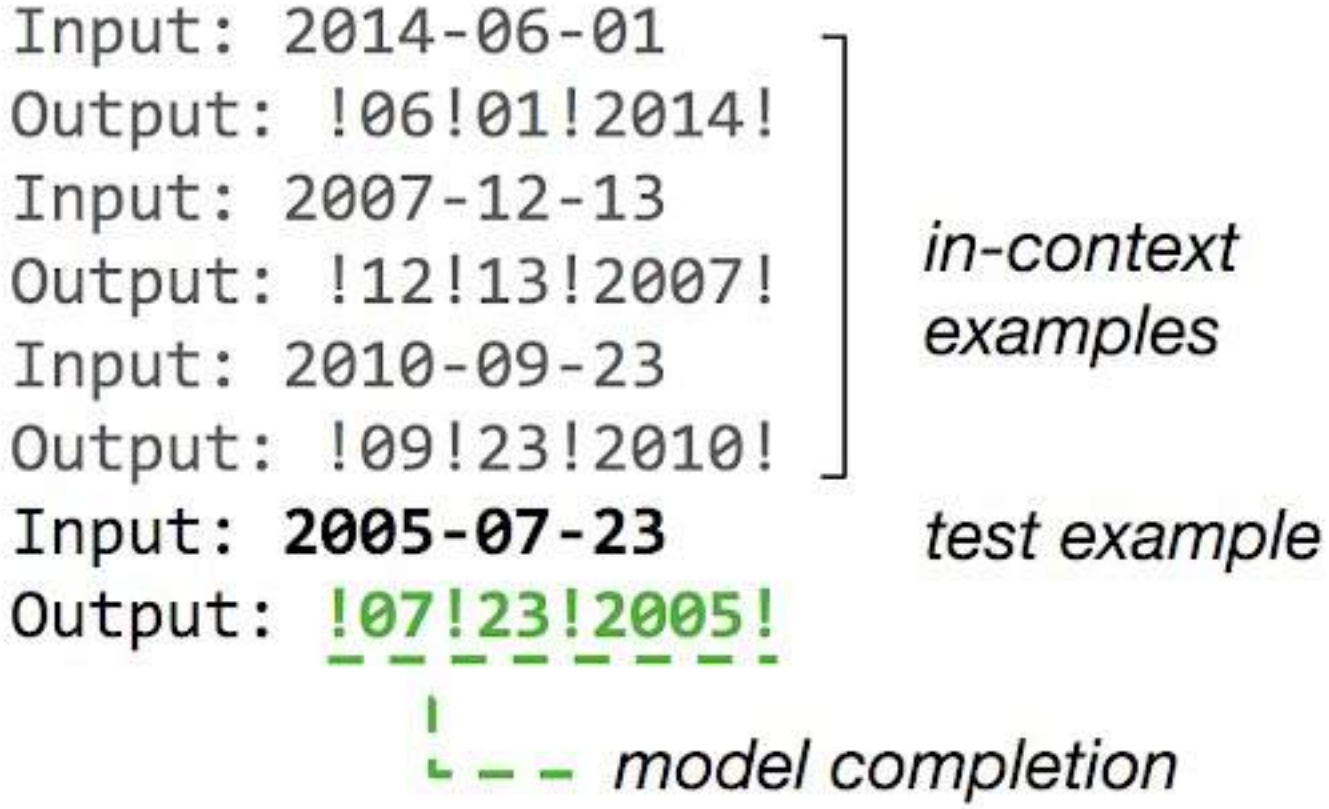
Translate English to French:
cat → ?

Translate English to French:
cat → chat
house → ?

Translate English to French:
cat → chat
dog → chien
house → ?

In-Context Learning

- In-Context Learning (ICL): learning from a few examples in the context



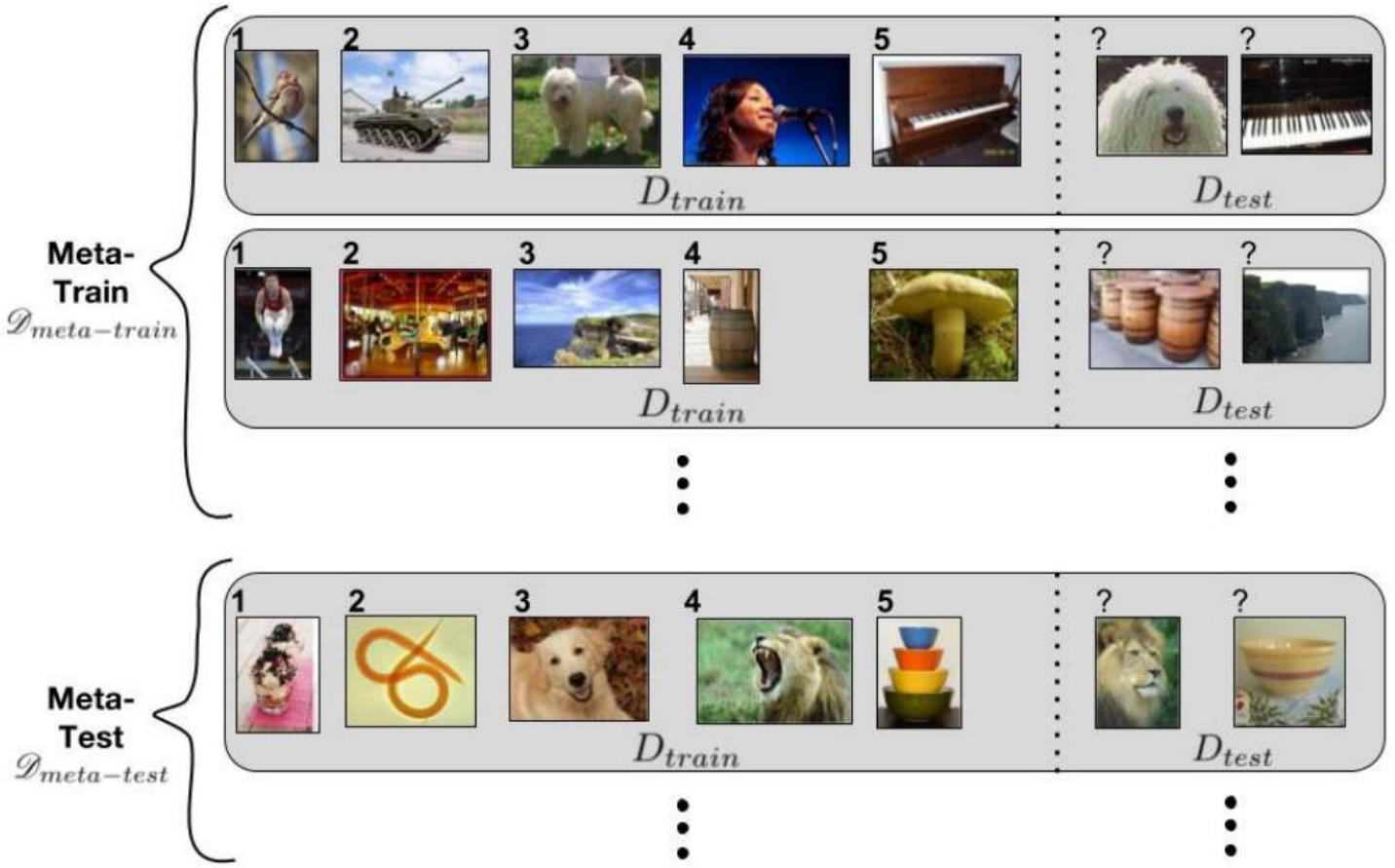
It allows models to **learn a task on the fly**, simply by being shown examples **within the prompt**, without any weight updates.

In-Context Learning

- **Why it works?**
  - It emerges in large-scale pretraining **(Emergent Abilities)**
  - Exposure to many formats and patterns in data **(SFT)**
  - The model's implicit **meta-learning** behavior: it acts like it's learning a task from examples, even though it's just predicting text
- But no long-term memory — model forgets after each prompt

Meta Learning

- Training Data
  - Inputs: $\{\mathcal{D}_{tr}, x_{test}\}$
  - Targets: y_{test}
- Testing Data
  - Input: $\{\mathcal{D}_{tr}^{0}, x_{test}^{0}\}$
  - Expected Output: y_{test}^{0}
- Meta Goal: learn how to learn



Chain-of-thought Promoting

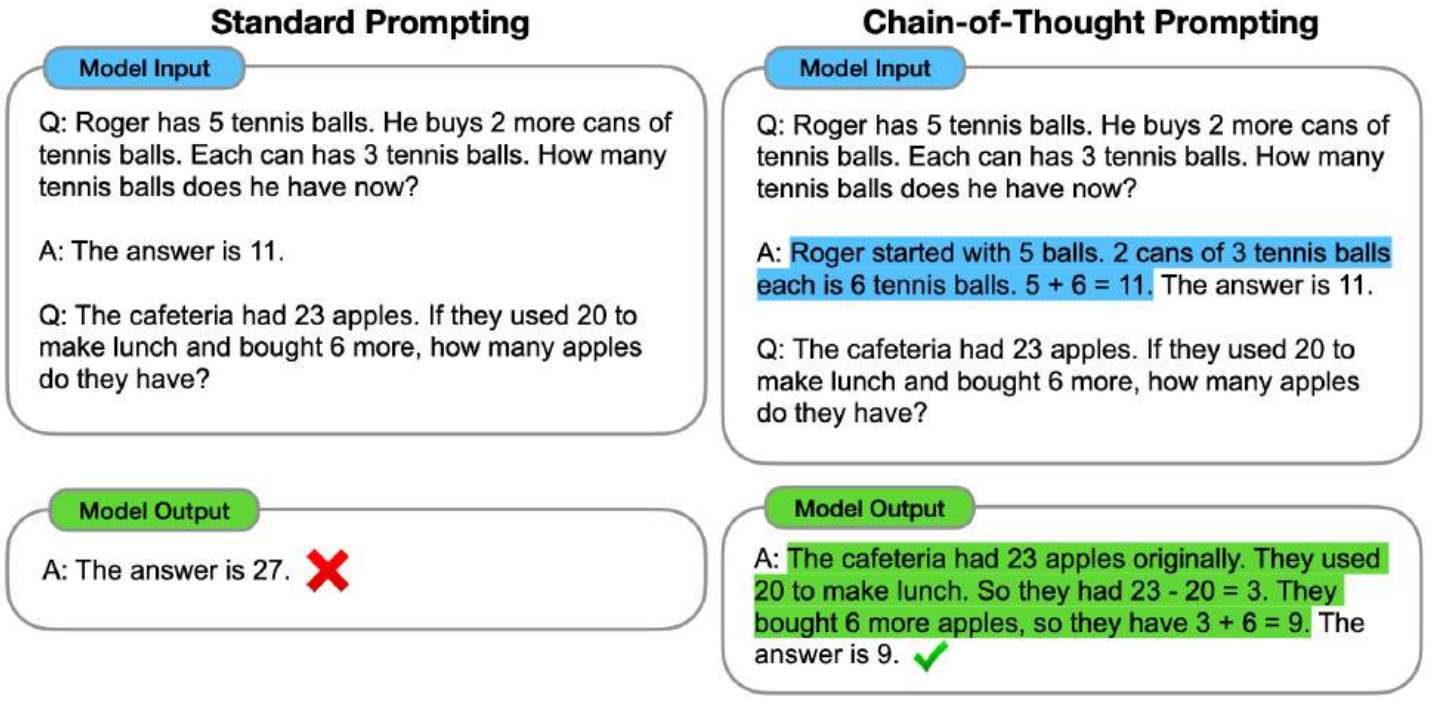
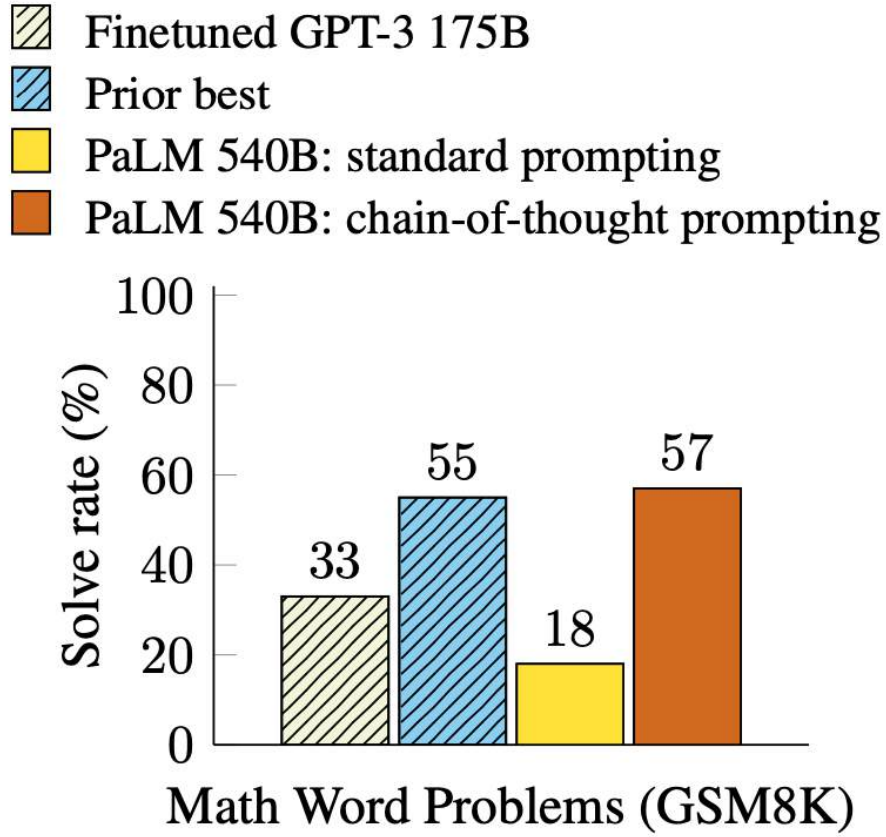
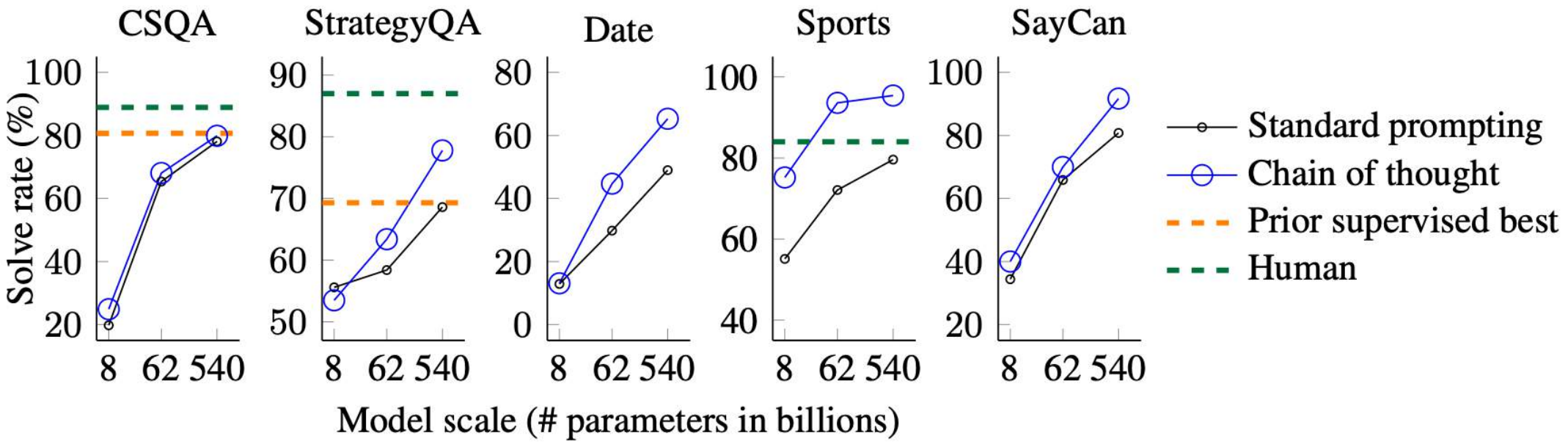


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted



Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2022

CoT: Emergency



Common sense reasoning tasks

Sampling Strategies

How can we sample from $p_\theta(x_i|x_1, \ldots, x_{i-1})$ at each step?

Option 1: Greedy Decoding

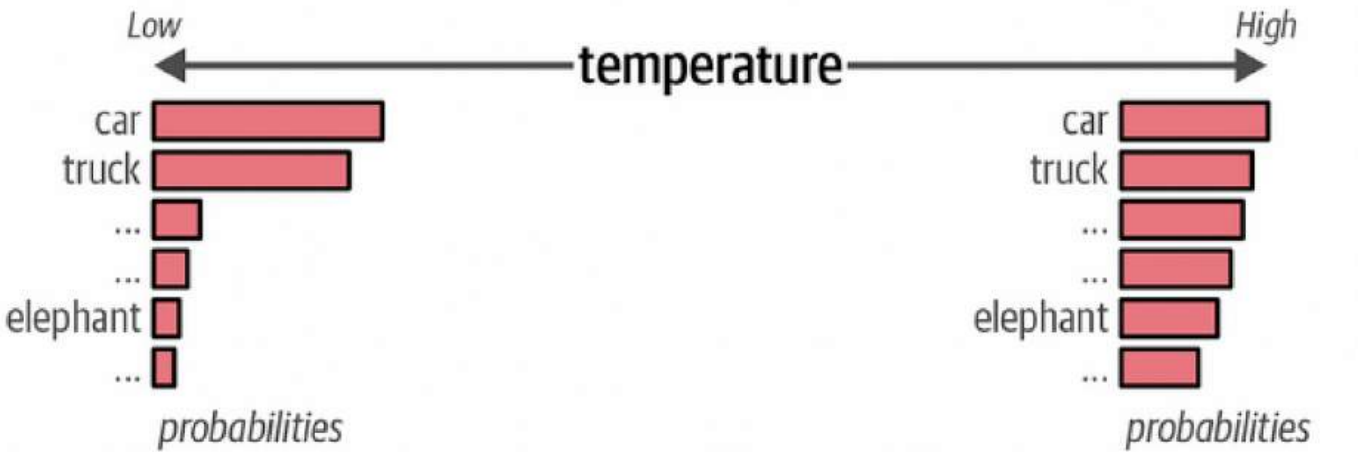
Pick the most likely token: $\hat{x}_i = \arg\max p_\theta(x_i|x_1, \ldots, x_{i-1})$

- Pros: Deterministic, fast
- Cons: Low diversity, prone to repetition, gets stuck in loops

Sampling Strategies

Option 2: Temperature Sampling: Adjusting the Creativity

- High Temperature (≥ 1): Creates more varied and creative outputs. Ideal for brainstorming or fiction writing.
- Low Temperature (~0): Produces more focused, predictable responses, looks like argmax, ideal for tasks like text summarization or factual Q&A.



v_k: the k-th vocabulary token,
l_k: the token's logit

$$p(v_k) = \frac{e^{\frac{l_k}{T}}}{\sum_k e^{\frac{l_k}{T}}}$$

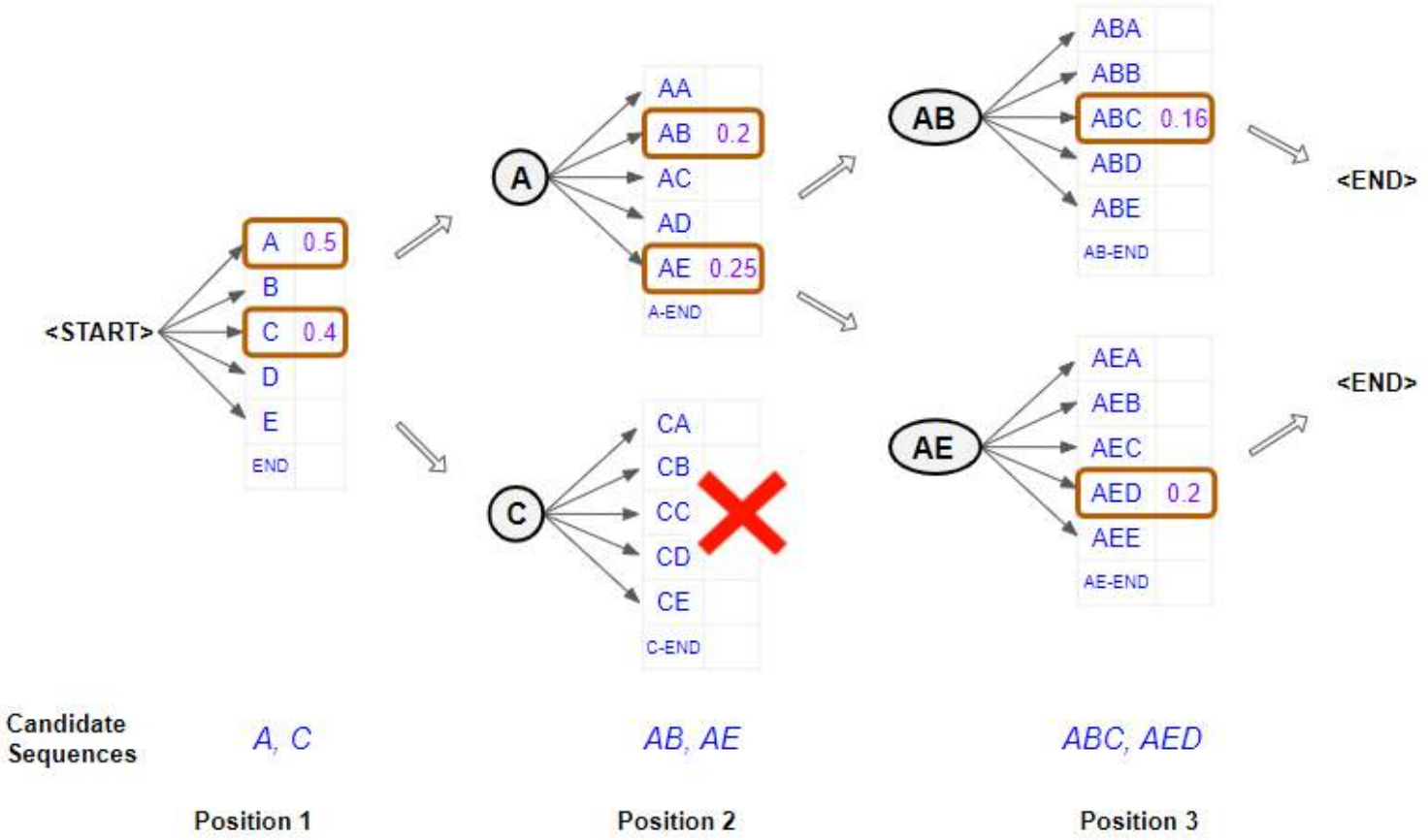
Sampling Strategies

Option 3: Top-K sampling limits the model's options to the top K most likely words

Option 4: Nucleus Sampling introduces sparsity, choose smallest set of tokens whose cumulative probability ≥ p

Option 5: Beam search maintains the top B most likely partial sequences (called beams) at each decoding step.

Beam Search



- Path probability

$$p_\theta(x_1, \dots, x_l) = \prod_{i=1}^{l} p_\theta(x_l | x_1, \dots, x_{l-1})$$

- Longer sequences naturally have smaller probabilities due to multiplying more terms.
- Fix: use length normalization:

$$\text{score} = \frac{1}{(5 + |\text{seq}|)^a / (5+1)^a} \cdot \log P(\text{seq})$$

PixelRNN

- Decompose an image to pixels

$$p_{model}(x) = \prod_{i=1}^{N} p_{model}(x_i | x_1, \dots, x_{i-1})$$

- Loss:

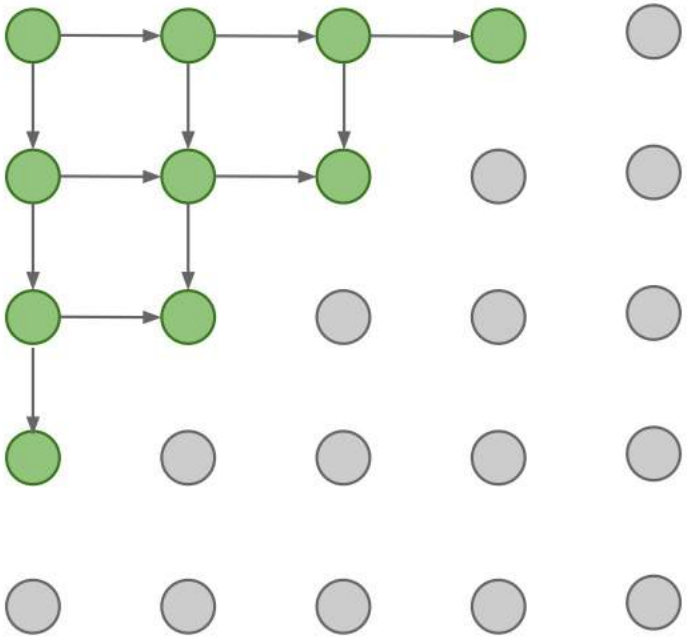
$$\sum_{x \in D} \log p_{model}(x|\theta) = \sum_{x \in D} \sum_{i}^{N} \log p_{model}(x_i | x_1, \dots, x_{i-1})$$

- $p_{model}(x_i | x_1, \dots, x_{i-1})$: discrete probability on [0,1,…,255]

"Pixel Recurrent Neural Networks"

PixelRNN

- Generate image pixels starting from corner



Generation: compute $p_{model}(x_i|x_1, \dots, x_{i-1}) \in \mathbb{R}^{256}$, then take x_i as the value with the highest probability.

The sample generation process is extremely slow for high-resolution images!

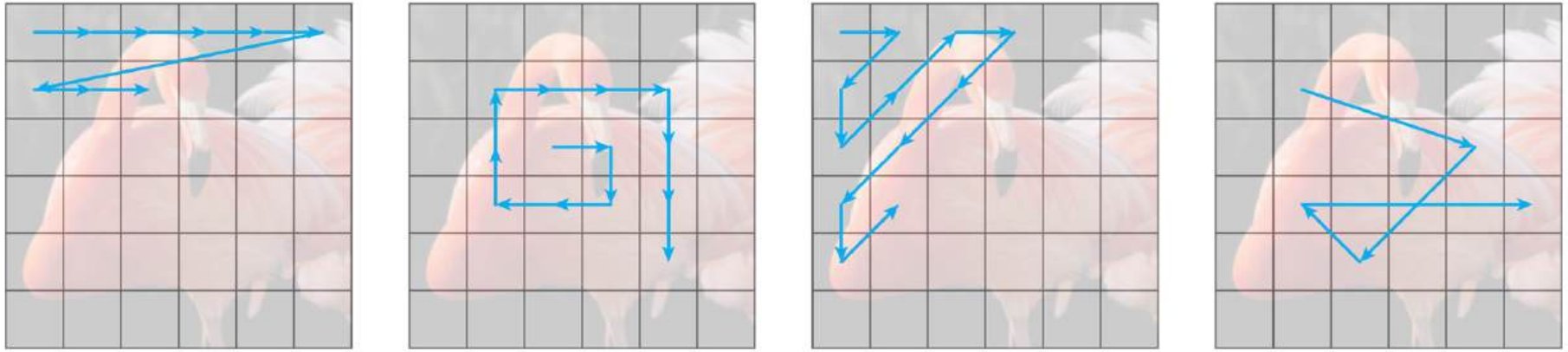
Recaps: Joint Distribution Decomposition

- A joint distribution can be written as a product of conditionals **in any order**

$$
\begin{aligned}
p(A, B, C) &= p(A)p(B \mid A)p(C \mid A, B) \\
&= p(A)p(C \mid A)p(B \mid A, C) \\
&= p(B)p(A \mid B)p(C \mid A, B) \\
&= p(B)p(C \mid B)p(A \mid B, C) \\
&= p(C)p(A \mid C)p(B \mid A, C) \\
&= p(C)p(B \mid C)p(A \mid B, C)
\end{aligned}
$$

Dependency Order

- The ordering may effect the training complexity



Joint Distribution Decomposition

- A joint distribution can be written as a product of conditionals **in any partition**

$$\begin{aligned} p(A, B, C, D) &= p(A, B)p(C, D \mid A, B) \\ &= p(C, D)p(A, B \mid C, D) \\ &= p(A, B, C)p(D \mid A, B, C) \\ &= \ldots \end{aligned}$$

Joint Distribution Decomposition

- Partitioning in the input space

$$p(x_1, x_2, ..., x_n) = p(x_1)p(x_2 \mid x_1)...p(x_n \mid x_1, x_2, ..., x_{n-1})$$

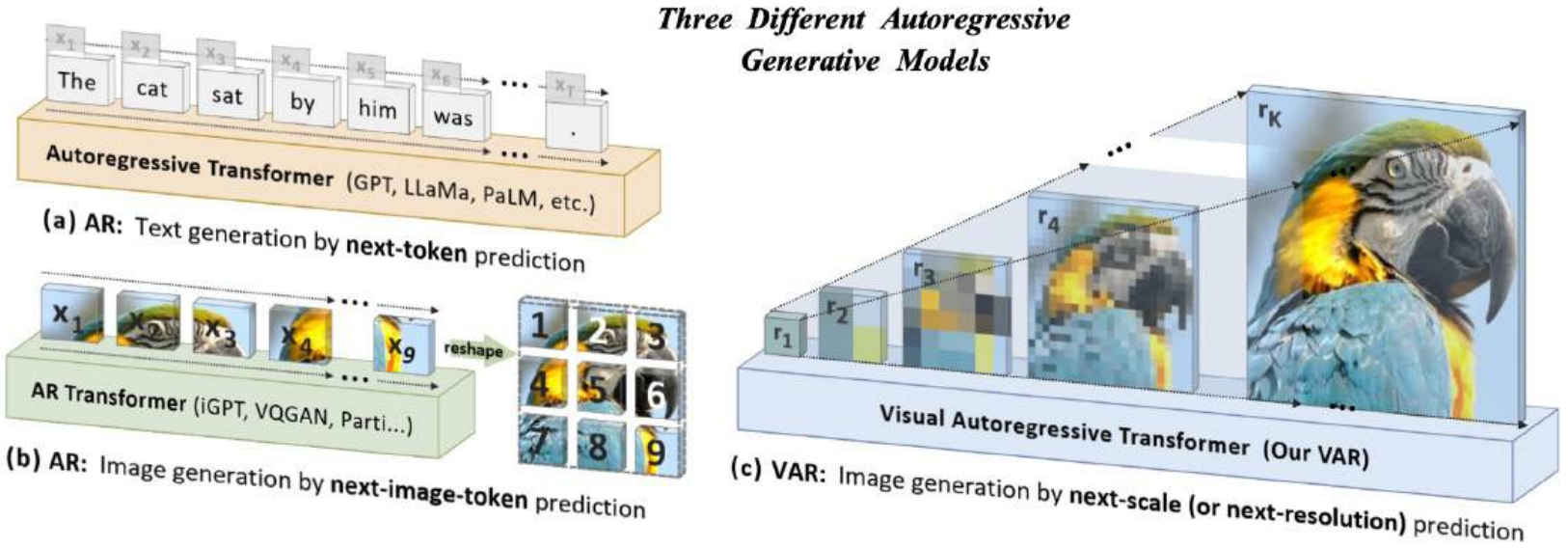
- Partition in the latent space (e.g. VAE to come next)

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z})$$
$$\text{with} \quad p(\mathbf{z}) = p(z_1)p(z_2 \mid z_1)...p(z_n \mid z_1, z_2, ..., z_{n-1})$$

Visual Autoregressive Modeling

- Next-scale Prediction: $(r_1, r_2, \ldots, r_k)$ multi scale representation

$$p(r_1, r_2, \ldots, r_K) = \prod_{k=1}^{K} p(r_k \mid r_1, r_2, \ldots, r_{k-1}),$$



"Visual Autoregressive Modeling: Scalable Image Generation via Next-Scale Prediction"

Next-token Prediction

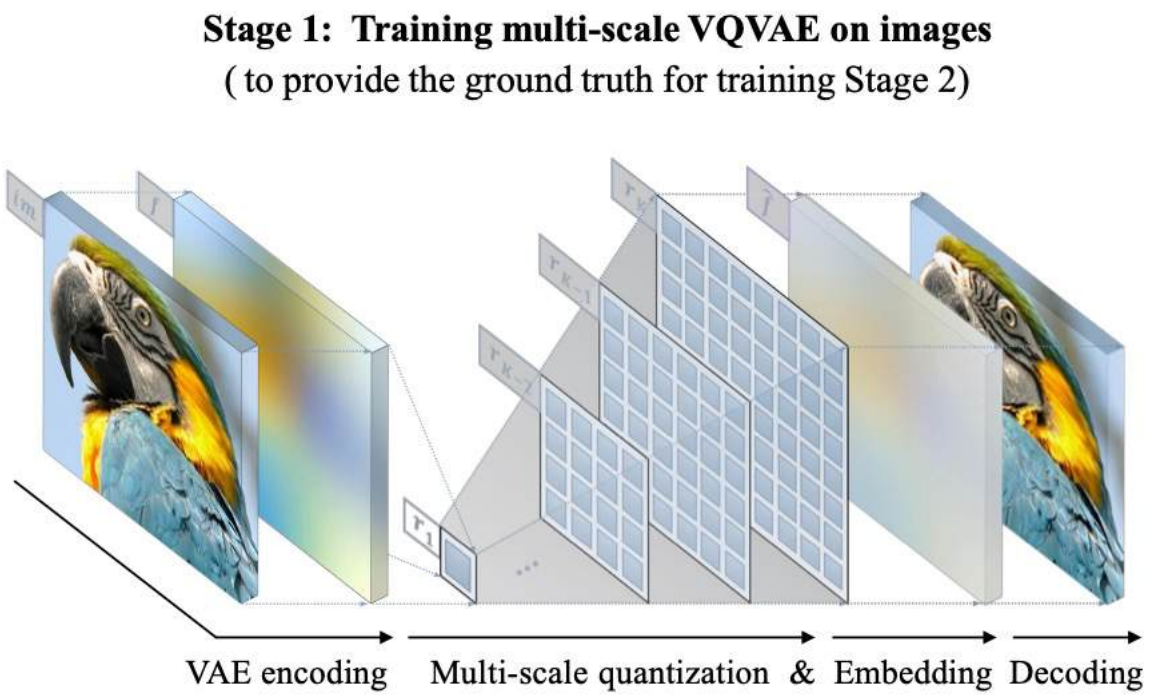
- A quantized autoencoder convert the image feature map to discrete tokens
- Encoder

$$f = \mathcal{E}(im), \qquad q = \mathcal{Q}(f),$$

$$q^{(i,j)} = \left(\underset{v \in [V]}{\arg\min} \|\mathbf{lookup}(Z, v) - f^{(i,j)}\|_2 \right) \in [V],$$

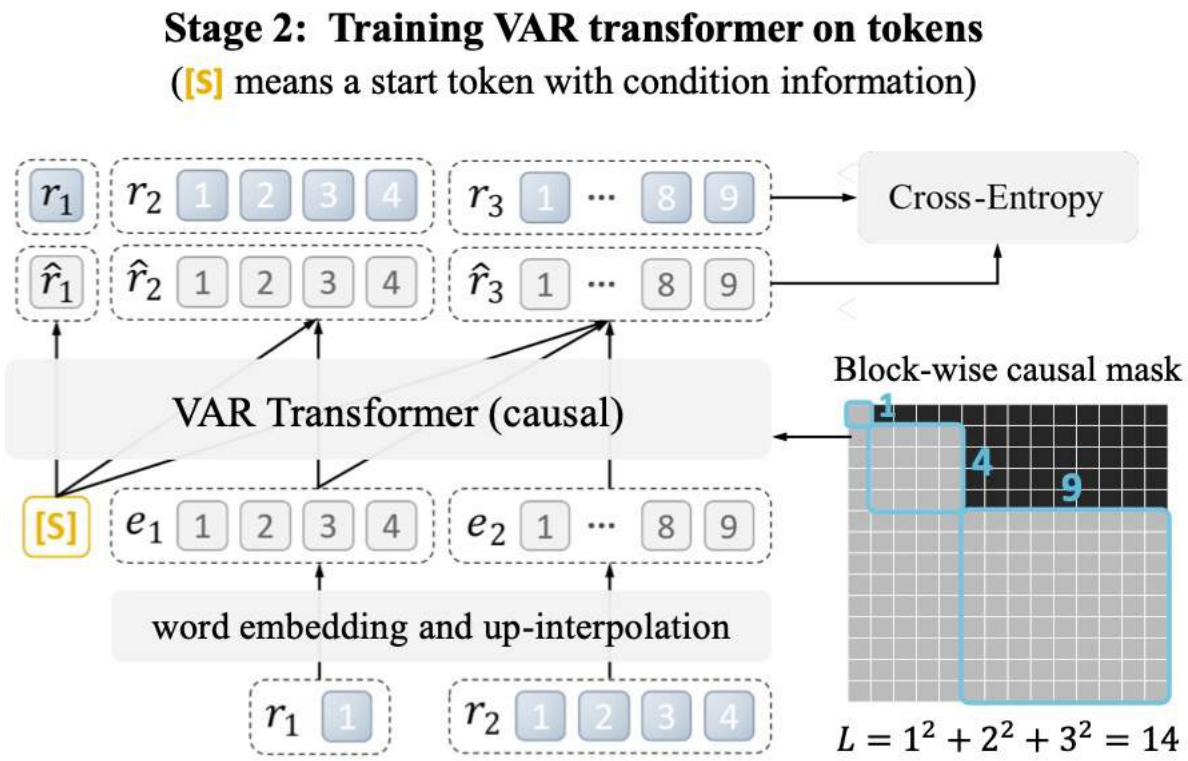
- Decoder

$$\hat{f} = \mathbf{lookup}(Z, q), \qquad \hat{im} = \mathcal{D}(\hat{f}),$$



Auto regressive learning in latent code space

Next-token Prediction



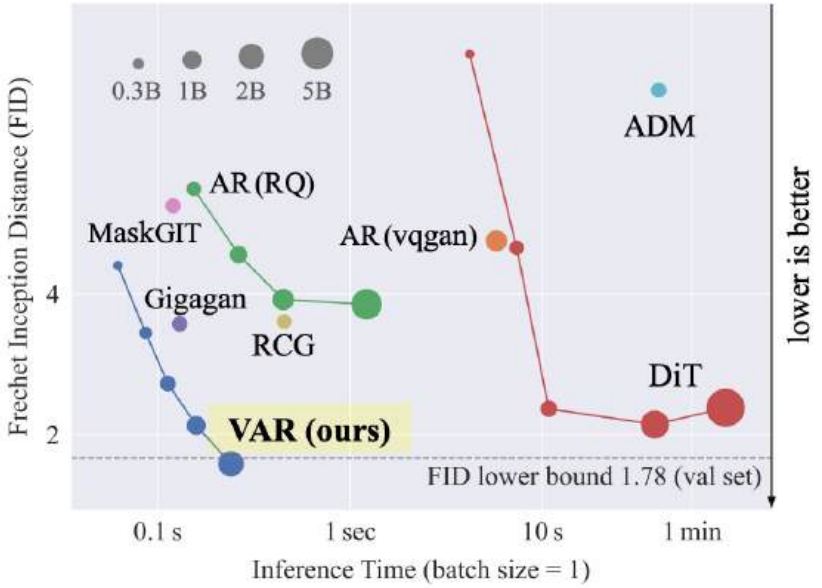
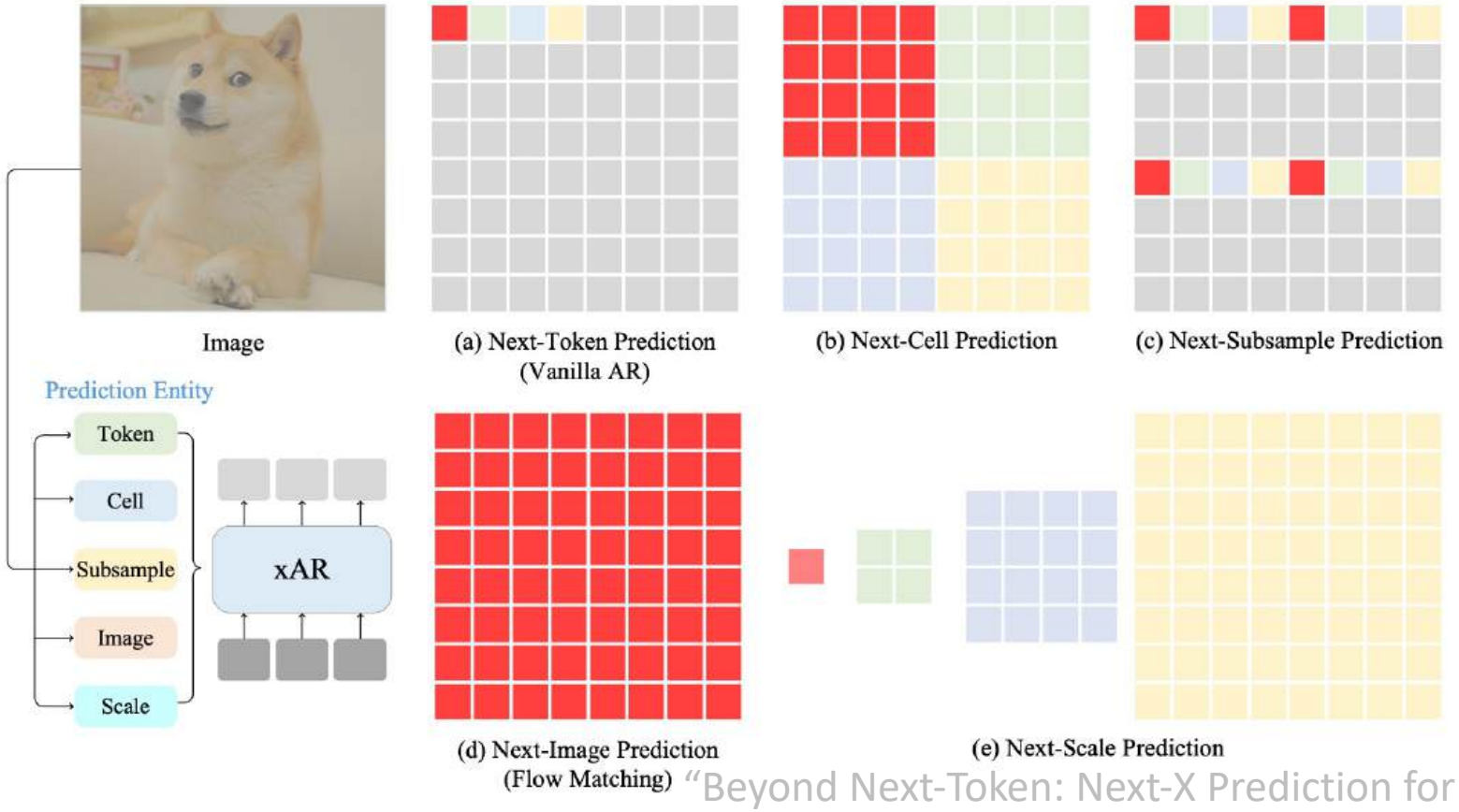


Figure 3: **Scaling behavior** of different model families on ImageNet 256×256 generation benchmark. The FID of the validation set serves as a reference lower bound (1.78). VAR with 2B parameters reaches an FID of 1.73, surpassing L-DiT with 3B or 7B parameters.

Figure 1: **Generated samples from Visual AutoRegressive (VAR) transformers trained on ImageNet**. We show 512×512 samples (top), 256×256 samples (middle), and zero-shot image editing results (bottom).

Visual Autoregressive Modeling

- Next-X Prediction: X can be any representation



"Beyond Next-Token: Next-X Prediction for Autoregressive Visual Generation "