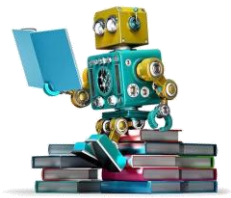
SDML
ML Paper Review

February 2025



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

Keyu Tian et al., ByteDance+

https://arxiv.org/abs/2404.02905

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

Keyu Tian[1,2], **Yi Jiang**[2,†], **Zehuan Yuan**[2,*], **Bingyue Peng**[2], **Liwei Wang**[1,*]
[1]Peking University [2]Bytedance Inc
keyutian@stu.pku.edu.cn, jiangyi.enjoy@bytedance.com,
yuanzehuan@bytedance.com, bingyue.peng@bytedance.com, wanglw@pku.edu.cn

Try and explore our online demo at: https://var.vision
Codes and models: https://github.com/FoundationVision/VAR

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).

Abstract

We present Visual AutoRegressive modeling (VAR), a new generation paradigm that redefines the autoregressive learning on images as coarse-to-fine "next-scale prediction" or "next-resolution prediction", diverging from the standard raster-scan "next-token prediction". This simple, intuitive methodology allows autoregressive (AR) transformers to learn visual distributions fast and can generalize well: VAR, for the *first time*, makes GPT-style AR models surpass diffusion transformers in image generation. On ImageNet 256×256 benchmark, VAR significantly improve AR baseline by improving Fréchet inception distance (FID) from 18.65 to 1.73, inception score (IS) from 80.4 to 350.2, with 20× faster inference speed. It is also empirically verified that VAR outperforms the Diffusion Transformer (DiT) in multiple dimensions including image quality, inference speed, data efficiency, and scalability. Scaling up VAR models exhibits clear power-law scaling laws similar to those observed in LLMs, with linear correlation coefficients near −0.998 as solid evidence. VAR further showcases zero-shot generalization ability in downstream tasks including image in-painting, out-painting, and editing. These results suggest VAR has initially emulated the two important properties of LLMs: **Scaling Laws** and **zero-shot** generalization. We have released all models and codes to promote the exploration of AR/VAR models for visual generation and unified learning.

*Corresponding authors: wanglw@pku.edu.cn, yuanzehuan@bytedance.com; †: project lead

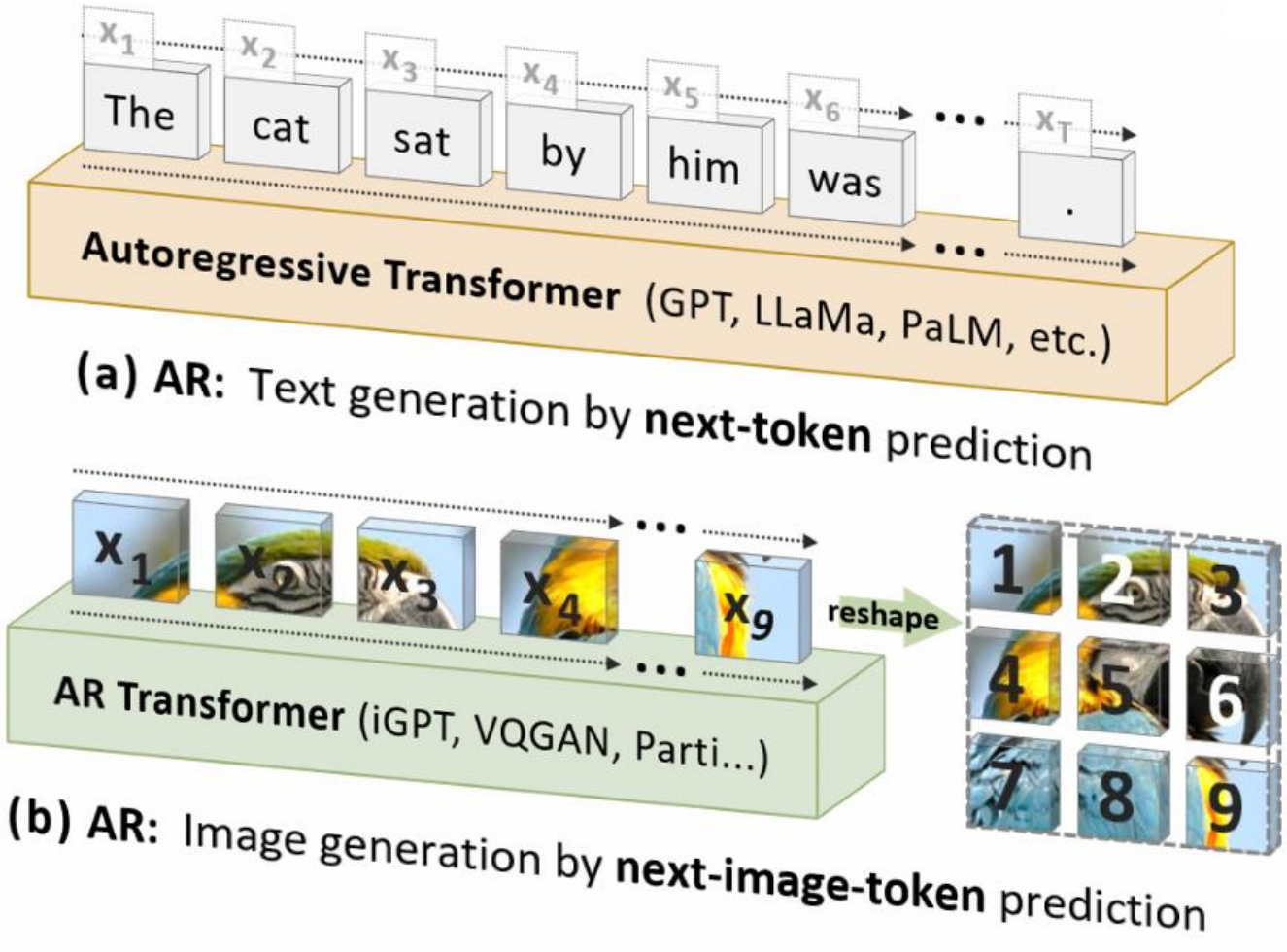
Preprint. Under review.

VAR overview

- Previous attempts to leverage the power of LLMs for generating images struggled how to decompose image into a sequence of parts
  - First try was predicting image patches in raster order left-right, top-down
  - This is the way we read English text on a page, but is unnatural for images
- VAR suggests low resolution to high resolution is a natural sequence
- Implementing this idea utilizes a few different older concepts
  - A variational autoencoder (VAE) allows encoding and decoding images
  - For tokenization, ViT models use infinite possible continuous-valued tokens
    - LLM prediction needs a finite vocabulary, so vector quantization used (VQ-VAE)
  - A GPT-2 clone LLM architecture used to predict tokens in the VAE latent space
  - Need special masking to predict multiple tokens for the next image at once

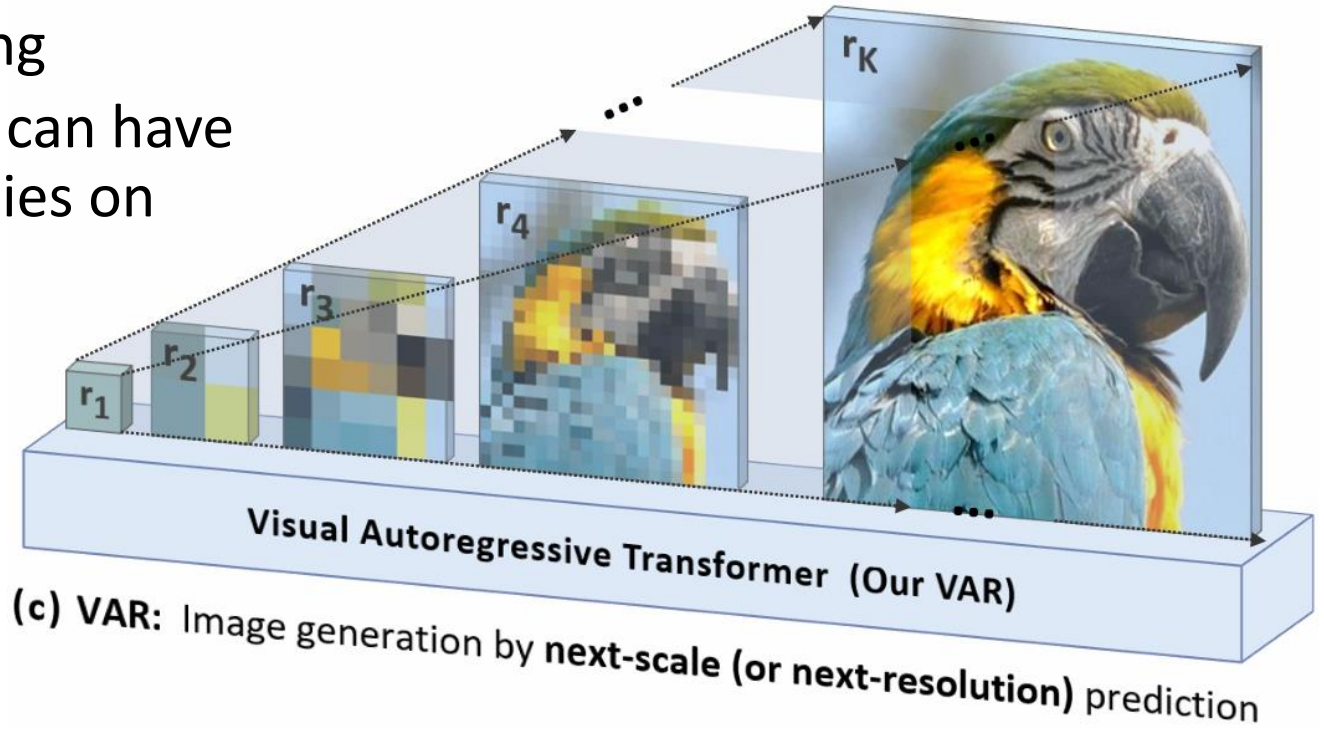
Autoregressive raster scan

- LLMs work on 1D input sequences
  - Models the next token's conditional probability given the previous context
- To copy LLM pattern, break the image into patches, then create a 1D sequence
  - But what's the right ordering?



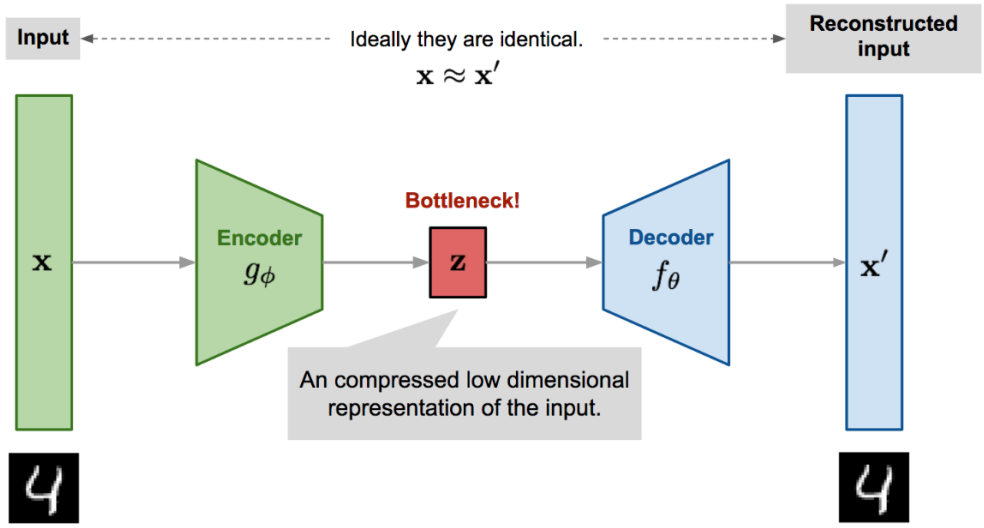
VAR autoregressive modeling

- Here, we model, for the next full image at a slightly higher resolution, the conditional probability given the previous context of all lower resolution versions of the full image
  - This has a logical 1D ordering
  - Every region of each image can have local and global dependencies on any region of the lower res images in the context
- How to implement?
  - Image dimensionality is too high, so we want to work in lower dims



Autoencoder

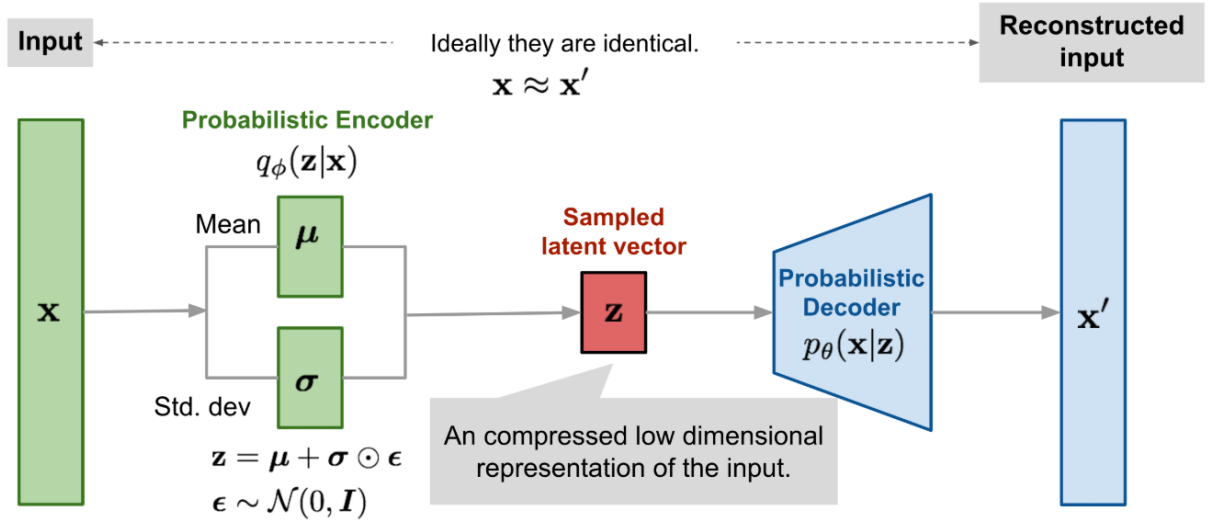
- A simple autoencoder compresses the input down to a bottleneck (encoder), then expands it back (decoder)
  - The bottleneck layer contains the reduced dimensionality embedding
- A simple autoencoder has a fixed embedding, so it's decoder can't generate multiple examples for the same request



https://lilianweng.github.io/posts/2018-08-12-vae/

Variational autoencoder

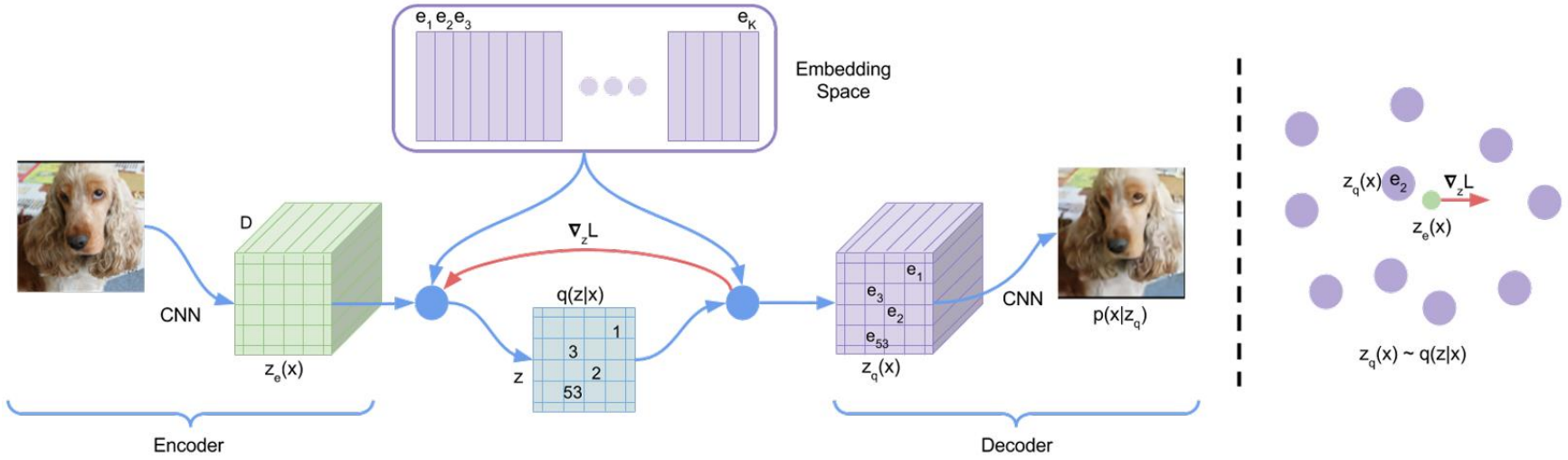
- A variational autoencoder compresses input down to a latent space, expressed as a probability distribution, then samples and expands
  - The training is more complex and outside the scope of this discussion
- Because the latent space is a probability distribution, its decoder can sample to generate multiple examples for the same request



https://lilianweng.github.io/posts/2018-08-12-vae/

Vector quantization and VQ-VAE

- Vector quantization is a generalization of rounding/quantization from scalars to multidimensional vectors
- The VAE is modified so that latents are assigned/rounded to the nearest embedding vector. Only the embedding ID need be stored.
  - These embedding vectors are stored in a *codebook* and are learned

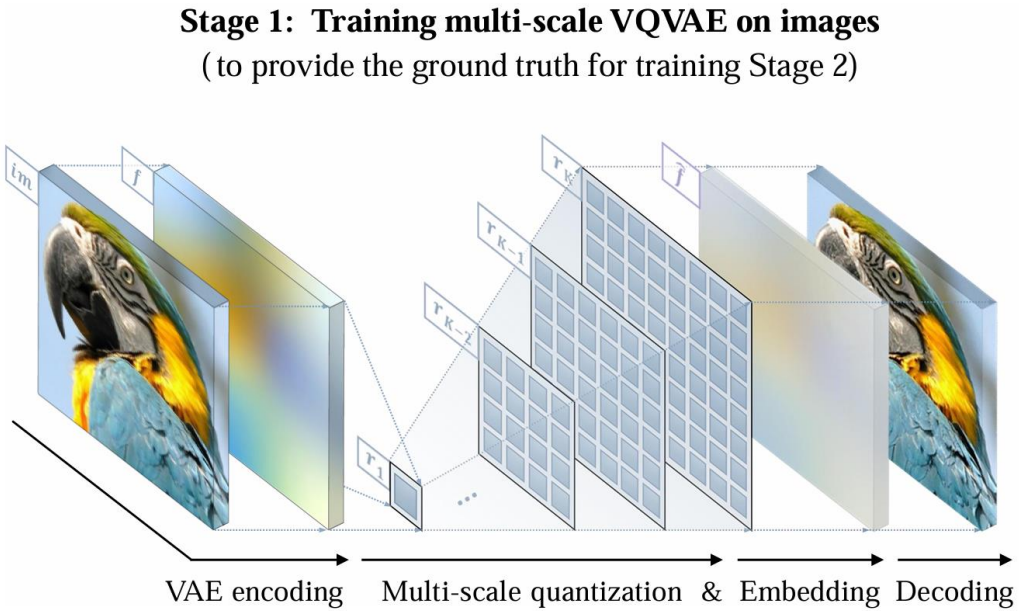


How to fit images into an LLM?

- A VQ-VAE will give us embeddings of images, but how do we use these embeddings as inputs and outputs of an LLM architecture?
  - Solution: break the image into patches, and give low resolution images fewer patches, and give higher resolution images more patches
- Now we need a special VQ-VAE that knows how to generate different resolution embeddings of the input image
- If our LLM architecture is supposed to predict the next higher resolution image, now that image is multiple patch tokens
  - GPT-2 predicts one token at a time, so input always grows one token at a time
  - VAR predicts one image at a time, so input can grow by multiple tokens
  - Solution: change the shape of the AR mask used with queries and keys

Multi-scale VQVAE

- One VQVAE handles all of the different scales/resolutions
  - All scales of embeddings are generated, using the same codebook
  - Refinement: higher scales are deltas from previous step (like xgboost, RVQ)
  - Decoder predicts pixels based on the embeddings



Multi-scale VQVAE iterative operation

- Because deltas from the previous scale are used, both the encoding and decoding have to start with the lowest resolution and iterate through higher scales

Algorithm 1: Multi-scale VQVAE Encoding

Inputs: raw image im;
Hyperparameters: steps K, resolutions $(h_k, w_k)_{k=1}^{K}$;
$f = \mathcal{E}(im)$, $R = []$;
for $k = 1, \cdots, K$ **do**
    $r_k = \mathcal{Q}(\text{interpolate}(f, h_k, w_k))$;
    $R = \text{queue\_push}(R, r_k)$;
    $z_k = \text{lookup}(Z, r_k)$;
    $z_k = \text{interpolate}(z_k, h_K, w_K)$;
    $f = f - \phi_k(z_k)$;
Return: multi-scale tokens R;

Algorithm 2: Multi-scale VQVAE Reconstruction

Inputs: multi-scale token maps R;
Hyperparameters: steps K, resolutions $(h_k, w_k)_{k=1}^{K}$;
$\hat{f} = 0$;
for $k = 1, \cdots, K$ **do**
    $r_k = \text{queue\_pop}(R)$;
    $z_k = \text{lookup}(Z, r_k)$;
    $z_k = \text{interpolate}(z_k, h_K, w_K)$;
    $\hat{f} = \hat{f} + \phi_k(z_k)$;
$\hat{im} = \mathcal{D}(\hat{f})$;
Return: reconstructed image $\hat{im}$;

Training Stages

- First, the multi-scale VQVAE is trained by itself on images
  - The VQVAE is then frozen
- Second, the transformer is trained to predict next resolution images

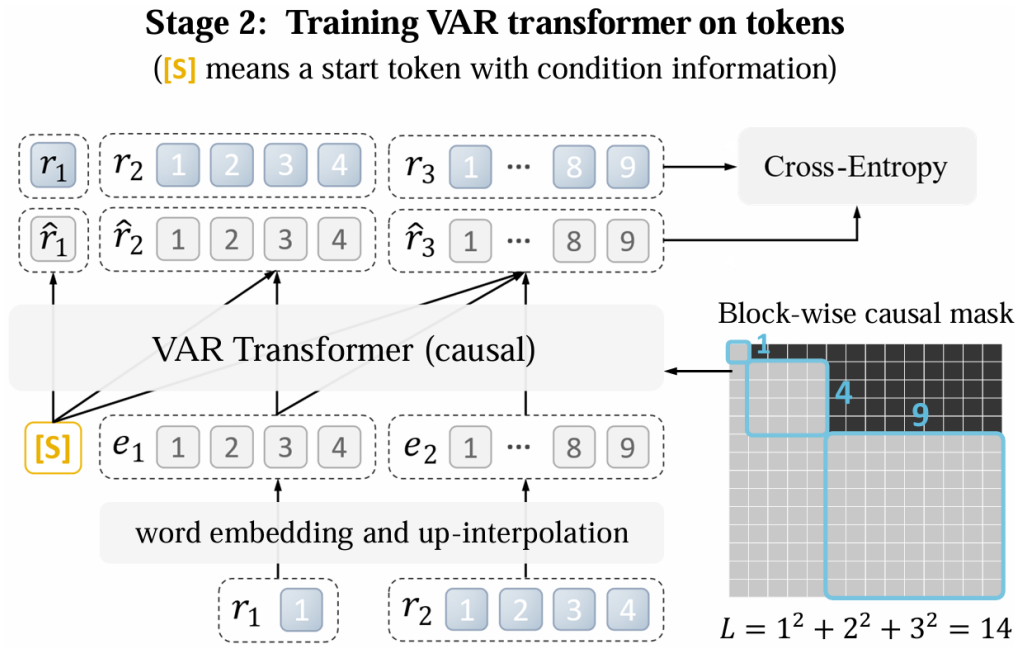


Image generation

- VAR was trained on ImageNet, so you give it the class you want
  - The obvious next step is to build a text-to-image version
- Starting with the token for that class, VAR predicts the 1x1 embedding, which is only 1 token
- Next, VAR predicts the next 4 tokens for the 2x2 image
- This repeats until it predicts the 16x16 image
- The sum of all the predictions is then sent to the decoder to generate the pixel output values

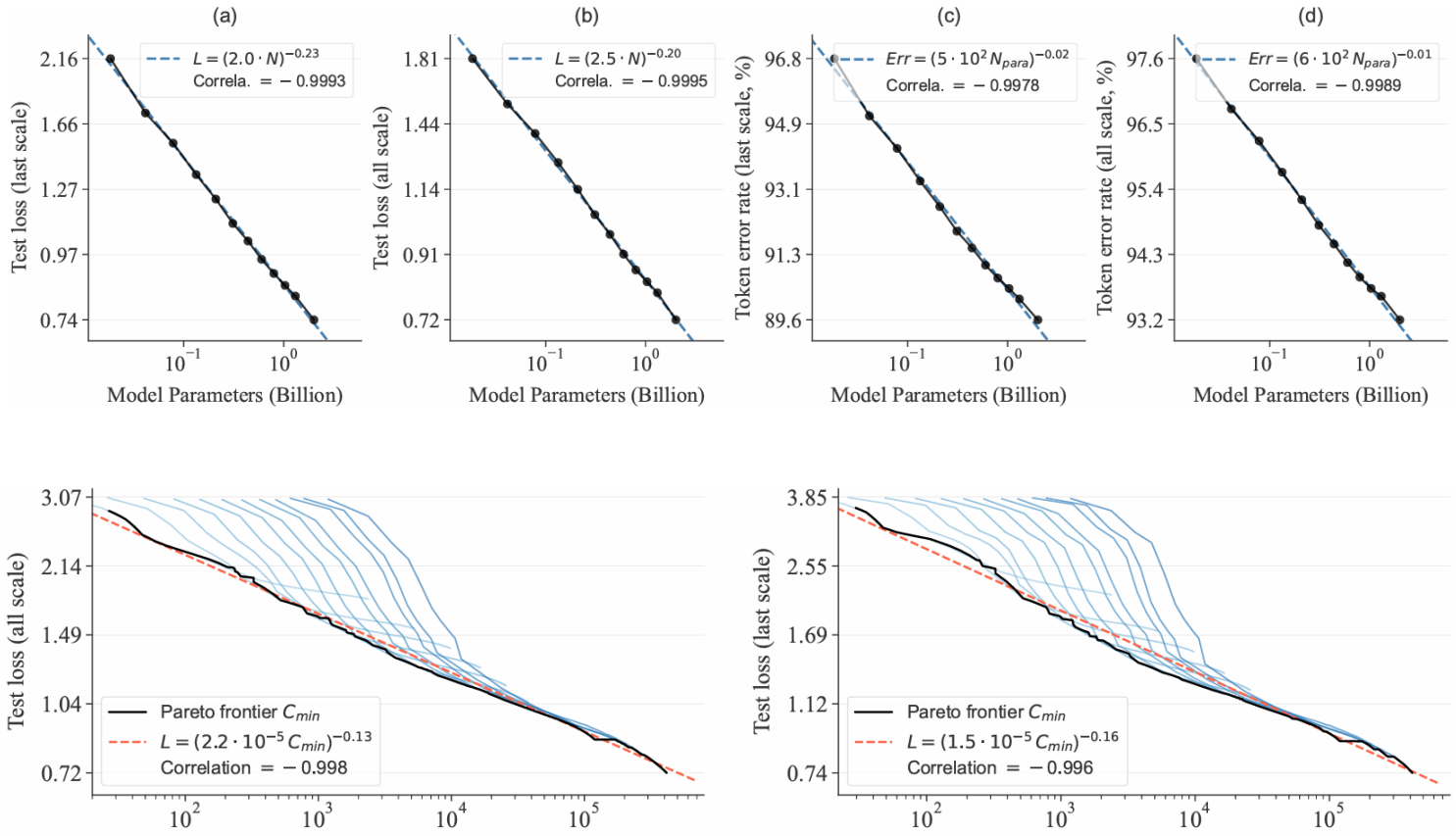
Results

- VAR uses an old GPT-2 architecture, so that good results can be attributed to the multi-scale AR, not transformer improvements
- Not only does it score well on image quality, but with only 10 image resolution steps, it is much faster than most other models

Type	Model	FID↓	IS↑	Pre↑	Rec↑	#Para	#Step	Time
GAN	BigGAN [13]	6.95	224.5	**0.89**	0.38	112M	1	–
GAN	GigaGAN [42]	3.45	225.5	0.84	**0.61**	569M	1	–
GAN	StyleGan-XL [74]	2.30	265.1	0.78	0.53	166M	1	0.3 [74]
Diff.	ADM [26]	10.94	101.0	0.69	0.63	554M	250	168 [74]
Diff.	CDM [36]	4.88	158.7	–	–	–	8100	–
Diff.	LDM-4-G [70]	3.60	247.7	–	–	400M	250	–
Diff.	DiT-L/2 [63]	5.02	167.2	0.75	0.57	458M	250	31
Diff.	DiT-XL/2 [63]	2.27	278.2	0.83	0.57	675M	250	45
Diff.	L-DiT-3B [3]	2.10	304.4	0.82	0.60	3.0B	250	>45
Diff.	L-DiT-7B [3]	2.28	316.2	0.83	0.58	7.0B	250	>45
Mask.	MaskGIT [17]	6.18	182.1	0.80	0.51	227M	8	0.5 [17]
Mask.	RCG (cond.) [51]	3.49	215.5	–	–	502M	20	1.9 [51]
AR	VQVAE-2† [68]	31.11	∼45	0.36	0.57	13.5B	5120	–
AR	VQGAN† [30]	18.65	80.4	0.78	0.26	227M	256	19 [17]
AR	VQGAN [30]	15.78	74.3	–	–	1.4B	256	24
AR	VQGAN-re [30]	5.20	280.3	–	–	1.4B	256	24
AR	ViTVQ [92]	4.17	175.1	–	–	1.7B	1024	>24
AR	ViTVQ-re [92]	3.04	227.4	–	–	1.7B	1024	>24
AR	RQTran. [50]	7.55	134.0	–	–	3.8B	68	21
AR	RQTran.-re [50]	3.80	323.7	–	–	3.8B	68	21
VAR	VAR-$d16$	3.30	274.4	0.84	0.51	310M	10	0.4
VAR	VAR-$d20$	2.57	302.6	0.83	0.56	600M	10	0.5
VAR	VAR-$d24$	2.09	312.9	0.82	0.59	1.0B	10	0.6
VAR	VAR-$d30$	1.92	323.1	0.82	0.59	2.0B	10	1
VAR	VAR-$d30$-re	**1.73**	**350.2**	0.82	0.60	2.0B	10	1
	(validation data)	1.78	236.9	0.75	0.67			

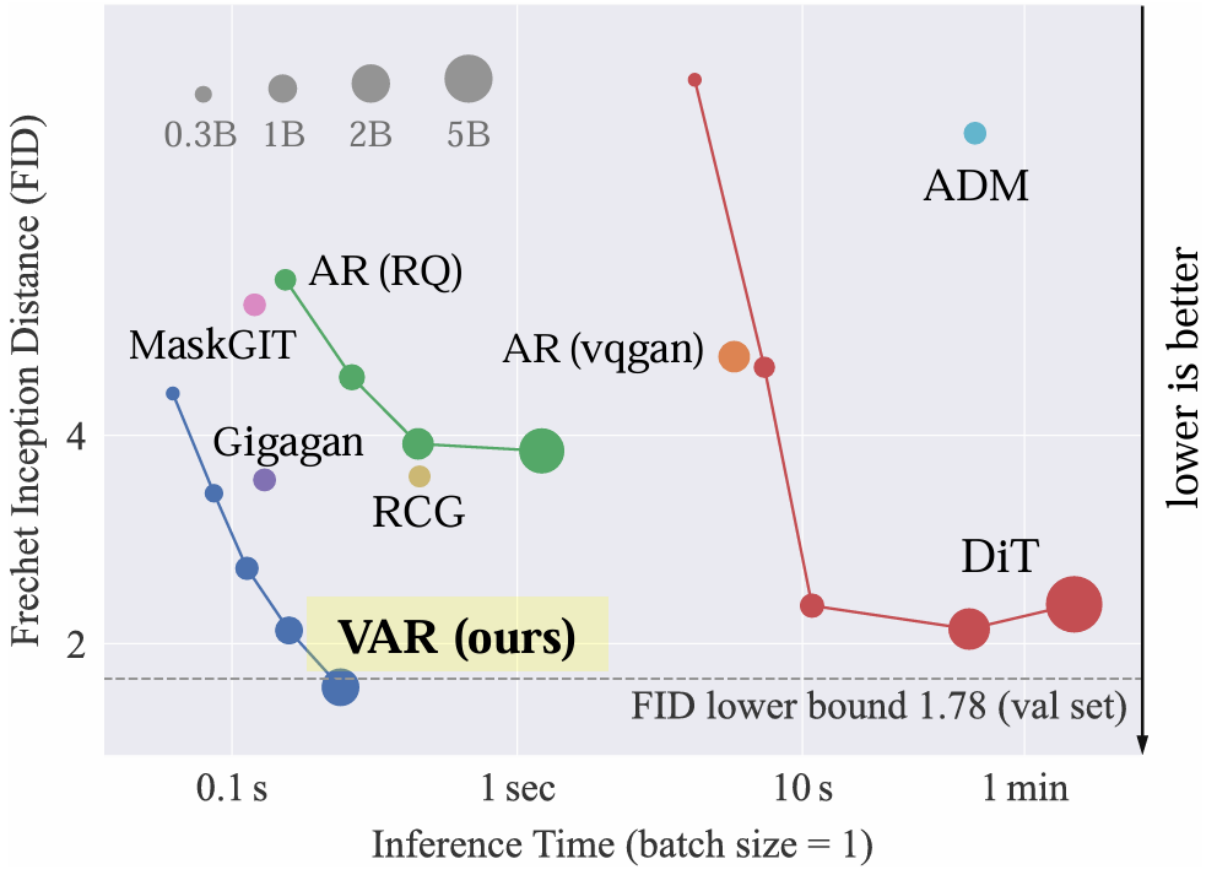
Scaling up [1]

- So far, VAR shows power law scaling patterns, similar to large LLMs

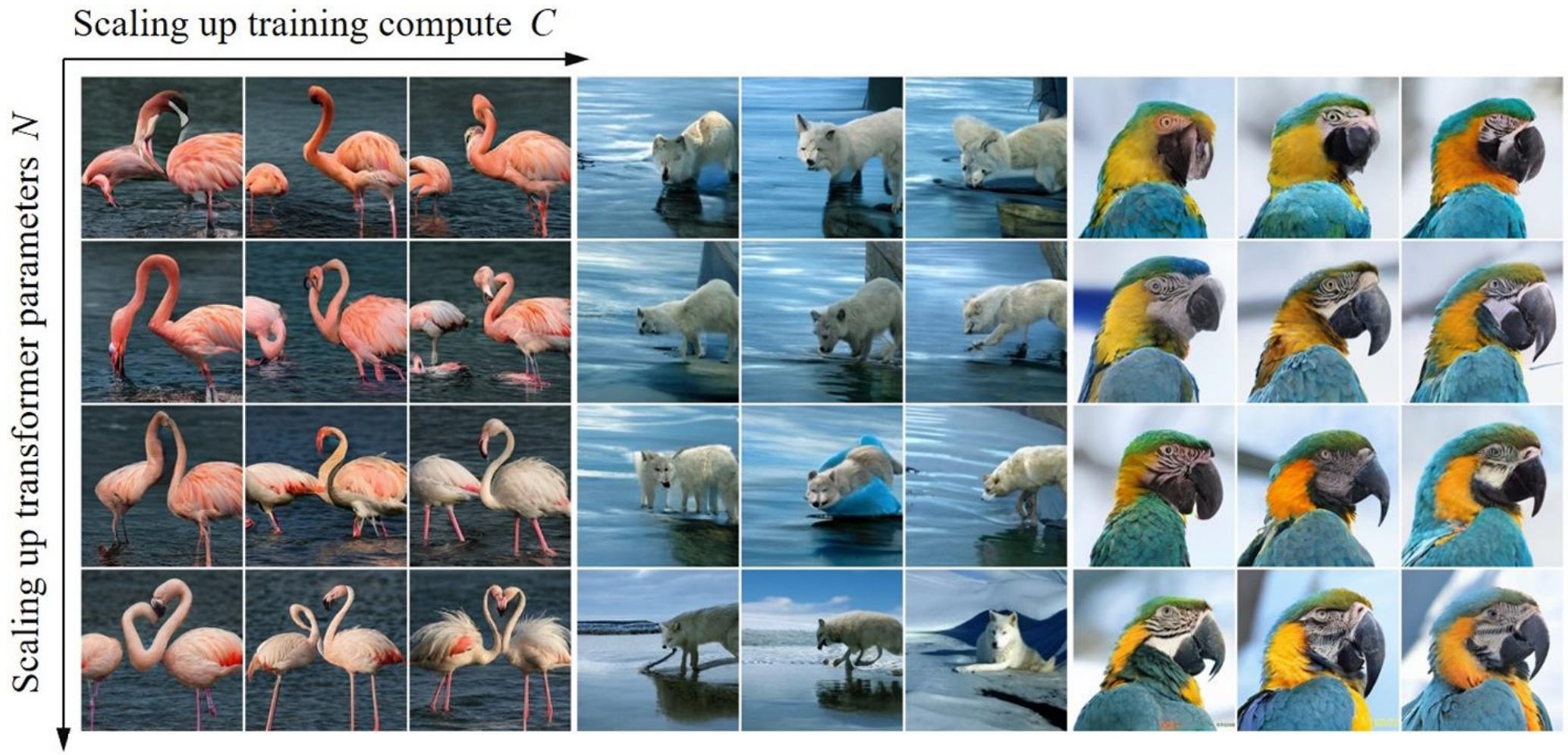


Scaling up [2]

- This chart clearly shows:
  - VAR is fast (further left)
  - Big VAR is high quality (low)
  - VAR quality does not saturate as it gets bigger, the way DiT and RQ do



Sample images



VAR conclusion

- VAR is fast and generates high quality images
- The low- to high-resolution inductive bias is logical
  - Intuitive 1D ordering
  - Proper conditional probability with respect to pixels in all directions
  - Allows tasks like in-painting or out-painting in any direction
- Code is available at https://github.com/FoundationVision/VAR
- The ByteDance team has already extended this technique to text-to-image generation in a paper called Infinity
- There are more details worth reading in the paper, but note that there is a lot of notation to learn (see appendix for a cheat sheet)

References

- From Autoencoder to Beta-VAE
  https://lilianweng.github.io/posts/2018-08-12-vae/
- Neural Discrete Representation Learning
  Aaron van den Oord, et al. (2017)
  https://arxiv.org/abs/1711.00937
- Infinity: Scaling Bitwise AutoRegressive Modeling for High-Resolution Image Synthesis
  Jian Han, et al. (2024)
  https://arxiv.org/abs/2412.04431

Appendix – VAR notation

- There is a lot of notation to capture raw images, embedded images at different resolutions, patches, tokens, and the VQVAE codebook
- Below is a summary of notation used in the VAR paper