[Neurips 2024 Best Paper]
VAR: Visual Autoregressive Modeling

Scalable Image Generation via Next-Scale Prediction

Kyungseon Lee

2026-06

Seoul National University

Image generation

Image generation overview

- GAN/VAE
- **Diffusion** (CNN-based $\rightarrow$ Transformer-based)
- Flow matching
- Autoregressive ✓: an image is generated as a product of conditional densities.

$$p(x) = \prod_{t=1}^{T} p(x_t \mid x_{<t})$$

x_t : the t-th unit, $x_{<t}$: previously generated units

Diffusion is already successful, so why do we use autoregressive methods for image generation?

Image generation: Why do we use AR?

Reproducing **the capabilities of LLMs** in image generation. (In the future, this direction can be extended to integrate text and image models.)

1. scaling law
   - When the model size increases, the test loss decreases.

     $$L(X) \approx aX^{-\alpha}$$

     where L is test loss, X is model size/data/compute, $a > 0$ is constant, and $\alpha > 0$ is the scaling exponent.
2. task generalization
   - The model can perform tasks that it was not explicitly trained for.
   - in-painting(filling masked regions), out-painting(filling padded regions) $\star$

Are the capabilities of LLMs due only to autoregressive modeling? $\rightarrow$ No.
To reproduce the capabilities of LLMs, we need to explore AR modeling, which is one key component of LLMs.

Background: Existing autoregressive image generation

- PixelCNN
  - PixelCNN generates pixels sequentially from top-left to bottom-right.
- VQVAE $\star$
  - VQVAE introduces image tokenization via autoencoder, generating tokens instead of raw pixels. Let $x \in \mathbb{R}^{H \times W \times 3}$, where H, W are height and width.

$$x \in \mathbb{R}^{H \times W \times 3} \xrightarrow{\text{CNN Encoder}} f \in \mathbb{R}^{h \times w \times C} \xrightarrow{\text{Tokenizer}} r \in [Q]^{h \times w}$$

    Q is the number of visual tokens in the dictionary, f is the feature tensor, r is the token map and h, w and C denote its height, width, and channel dimension.
- VAR

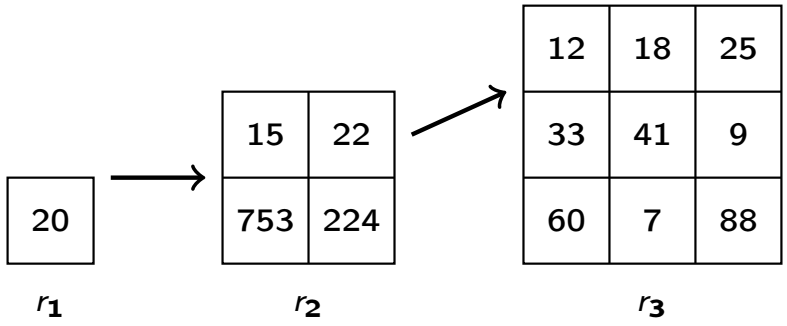
Background: Existing autoregressive image generation

Tokenize the image using VQ-VAE.

- The CNN encoder $\mathcal{E} : \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^{h\times w\times C}$ produces a continuous latent feature map:
$$f = \mathcal{E}(x) \in \mathbb{R}^{h\times w\times C}.$$
- VQ-VAE has a learnable dictionary $E = \{e_1, \ldots, e_Q\}$, $e_q \in \mathbb{R}^C$. For each $i \in [h] \times [w]$, feature vector $f^{(i)} \in \mathbb{R}^C$ choose its nearest dictionary vector e_q:
$$r^{(i)} = \arg\min_{q\in[Q]} \left\| f^{(i)} - e_q \right\|_2^2, \qquad d^{(i)} = e_{r^{(i)}}.$$
  - $r = (r^{(j)})_{j=1}^{h\times w} \in [Q]^{h\times w}$: token map of dictionary indices.
  - $d = (d^{(j)})_{j=1}^{h\times w} \in \mathbb{R}^{h\times w\times C}$: feature tensor obtained by replacing each feature vector with its corresponding dictionary vector.
  - $\mathrm{Dict}(f) = r, d$

VAR Overview

VAR



Generating token maps sequentially.

- VAR introduces next-scale generation, the natural way people paint. So it splits the token map r into $r_1, r_2, \cdots, r_K$ to capture the change from silhouette to detail. $r_k \in [Q]^{h_k \times w_k}$ is the token map at scale and $[s]$ is a start token.
- Train the transformer.

$$f \in \mathbb{R}^{h \times w \times C} \xrightarrow{\text{Multi-scale Tokenizer}} \{r_i\}_{i=1}^{K} \xrightarrow{\text{Transformer}} \{p(r_j \mid [s], r_1, \cdots, r_{j-1})\}_{j=1}^{K}$$

- Inference: Generate the token maps.

$$[s] \xrightarrow{\text{Transformer}} p(r_1 \mid [s]) \xrightarrow{\text{Transformer}} \cdots \rightarrow p(r_K \mid [s], r_1, \cdots, r_{K-1})$$

Step 1 - Multi-scale Tokenizer

VAR copies how people paint, the rough shape first and the details later. So instead of a single token map r, it builds token maps $r_1, r_2, \ldots, r_K$ that run from silhouette to fine detail.

- The tokenizer is a loop that keeps a leftover feature f_i. Start with $f_1 = f$, with sizes $h_1 \leq h_2 \leq \cdots \leq h_K$.
  - for $i = 1, \ldots, K$:
  - $g_i = \pi_{K,i}(f_i)$, shrink the leftover to scale i.
  - r_i, $d_i = \text{Dict}(g_i)$, read off the tokens and their vectors.
  - $f_{i+1} = f_i - \pi_{i,K}(d_i)$, remove what scale i already caught.
- $\pi_{i,k}$ resizes from scale $h_i \times w_i$ to scale $h_k \times w_k$.

Step 2 - Training with decoder-based Transformer

- Step 1 already gives the ground-truth token maps $r_1, r_2, \cdots, r_K$. We feed them all in at once and train the transformer by teacher forcing.
- Input is $[s], r_1, \cdots, r_{K-1}$, and the target is $r_1, r_2, \cdots, r_K$. Each scale is predicted from the scales before it:

$$p(r_{i+1} \mid [s], \pi_{1,2}(r_1) + p_2, \cdots, \pi_{i,i+1}(r_i) + p_{i+1})$$

where $p_i \in \mathbb{R}^{h_i \times w_i}$ is the position embedding matrix.

Step 3 - Inference with decoder-based Transformer

- Now there are no ground-truth maps. Starting from $[s]$, we sample r_1, feed it back to sample r_2, and keep going up to r_K:

$$r_{i+1} \sim p(r_{i+1} \mid [s], \pi_{1,2}(r_1) + p_2, \cdots, \pi_{i,i+1}(r_i) + p_{i+1})$$

- Once we have $r_1, r_2, \cdots, r_K$, we look up their dictionary vectors $d_1, d_2, \cdots, d_K$ and rebuild the feature tensor and image with the CNN decoder $\mathcal{D} : \mathbb{R}^{h \times w \times C} \to \mathbb{R}^{H \times W \times 3}$:

$$\hat{f} = \sum_{i=1}^{K} \pi_{i,K}(d_i), \quad \hat{X} = \mathcal{D}(\hat{f})$$

Results

Model	FID↓	IS↑	#step	rel. time
DiT-XL/2 (diffusion)	2.27	278.2	250	45
VQGAN (raster AR)	18.65	80.4	256	slow
VAR-d30	1.73	350.2	10	1

- On ImageNet 256×256, AR beats the diffusion transformer for the first time.
- Scaling law: $\log L = -\alpha \log X + c$ with Pearson correlation -0.998.
- Zero-shot in-painting, out-painting, and editing with no extra training.

Appendix: VQ-VAE gradient update

Tokenize the image using VQ-VAE

- The nearest-neighbor step is non-differentiable:

$$\text{Decoder input: } d = \mathcal{E}(x) + (d - \mathcal{E}(x))$$

$$\text{Gradient: } \mathcal{E}(x) = \mathcal{E}(x) + \text{detach}(d - \mathcal{E}(x))$$

Appendix: VAR step 1

Step 1 - Multi-scale Tokenizer

Introduces next-scale generation which is natural method in human painting. so they make the token map r to $r_1, r_2, \cdots, r_K$ for presenting the change in silhouette to detail.

- For getting the token maps $r_1, r_2, \cdots, r_K$ they introduce an iterative update Tokenizer where $f_k \in \mathbb{R}^{h_k \times w_k \times C}$ and $h_k \leq h_{k+1},\ w_k \leq w_{k+1}$

$$f_0 = f, \quad f_{i+1} = f_i - \phi_i(\mathcal{I}(\mathcal{V}(\mathcal{I}(f_i, h_i, w_i)), h_K, w_K))$$

- make a r to $r_1, r_2, \cdots, r_K$, where $r_k \in [Q]^{h_k \times w_k}$.

$$r_i = \mathcal{Q}(\mathcal{I}(f_i, h_i, w_i))$$

  - $\mathcal{I}(\cdot, h, w)$: resizes a feature map to scale $h \times w$.
  - $\mathcal{Q} : \mathbb{R}^{h_k \times w_k \times C} \to \mathbb{R}^{h_k \times w_k}$,
    $r_j = \mathcal{Q}(f_j) = \{\arg\min_{q \in [Q]} \left\| f^{(i)} - e_q \right\|_2^2\}_{i=1}^{h \times w}$, vector quantization.
  - $\mathcal{V} : \mathbb{R}^{h_k \times w_k \times C} \to \mathbb{R}^{h_k \times w_k \times C}$,
    $d = \mathcal{V}(f) = \{\arg\min_{e_q \in E} \left\| f^{(i)} - e_q \right\|_2^2\}_{i=1}^{h \times w}$, codebook lookup.
  - $\phi_k : \mathbb{R}^{h_k \times w_k \times C} \to \mathbb{R}^{h_k \times w_k \times C}$, scale-specific convolution layer.

References

- Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, Liwei Wang. *Visual Autoregressive Modeling: Scalable Image Generation via Next-Scale Prediction*. NeurIPS, 2024.
- Esser, Rombach, Ommer. *Taming Transformers for High-Resolution Image Synthesis*. CVPR, 2021.
- Peebles, Xie. *Scalable Diffusion Models with Transformers*. ICCV, 2023.