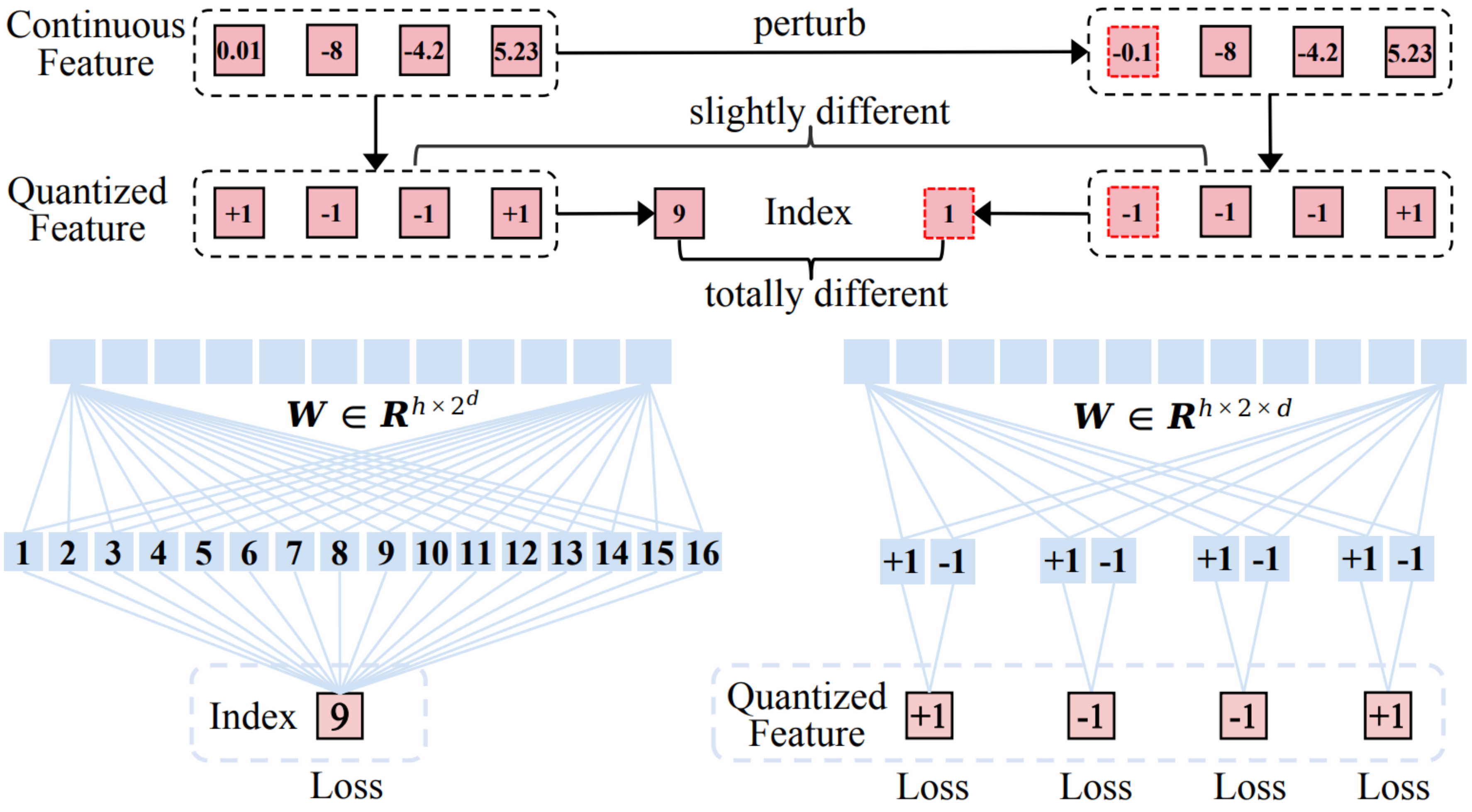
Infinity∞: Scaling Bitwise AutoRegressive Modeling for High-Resolution Image Synthesis

Jian Han*, **Jinlai Liu***, **Yi Jiang***, **Bin Yan**
Yuqi Zhang, **Zehuan Yuan**†, **Bingyue Peng**, **Xiaobing Liu**
ByteDance
{hanjian.thu123,liujinlai.licio,jiangyi.enjoy,yanbin.master}@bytedance.com,
{zhangyuqi.hi,yuanzehuan,bingyue.peng,will.liu}@bytedance.com,

Codes and models: https://github.com/FoundationVision/Infinity

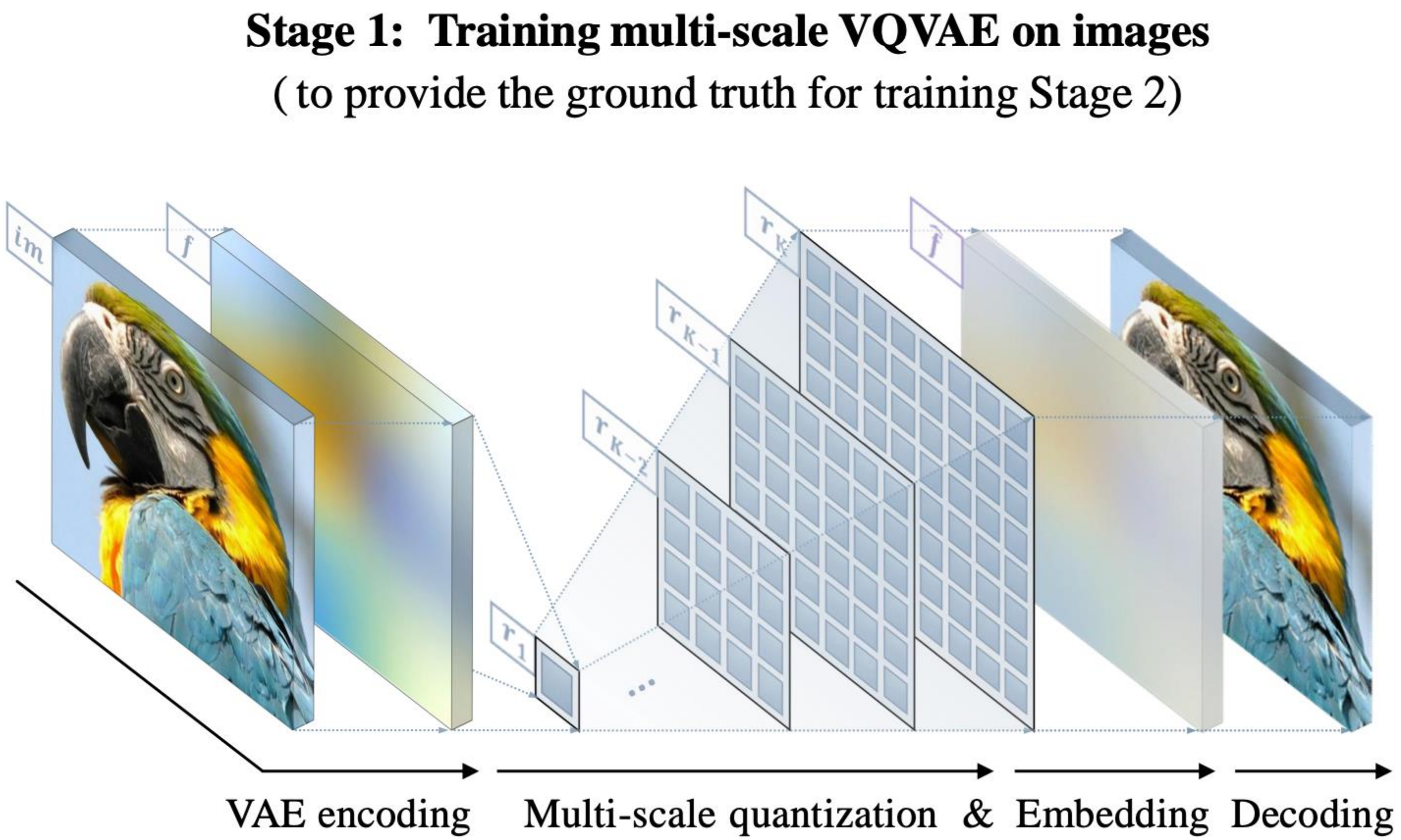
Index-wise token VS Bit-wise token



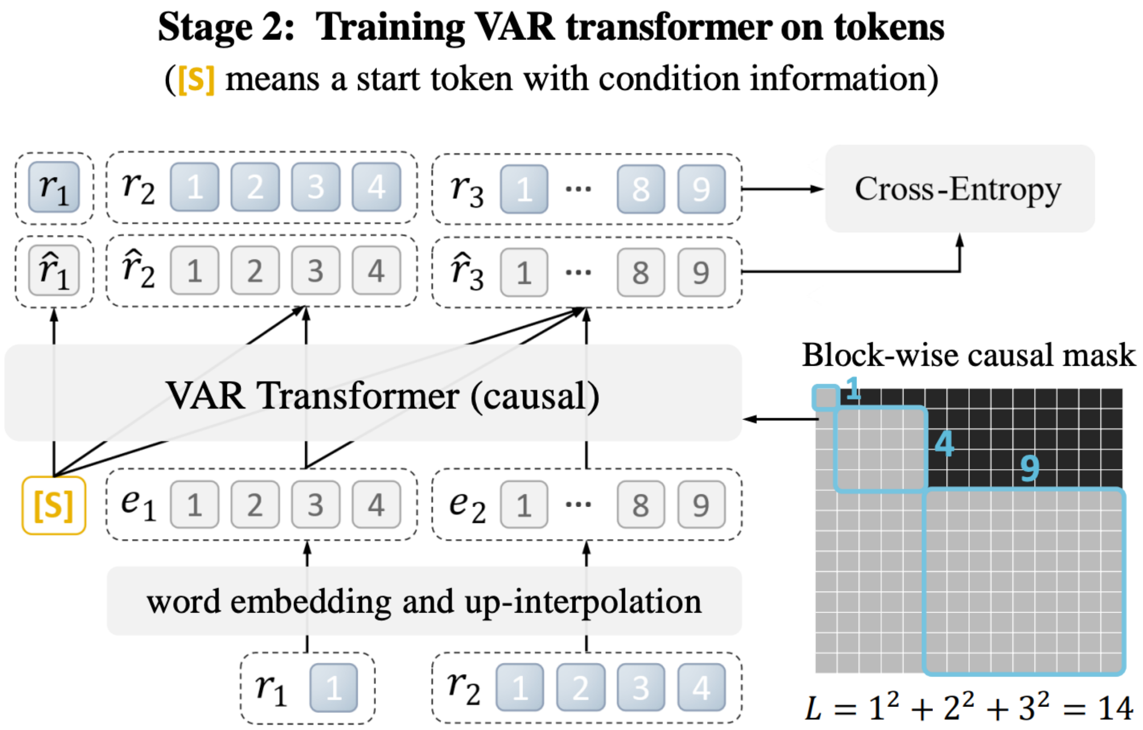
Bitwise modeling framework

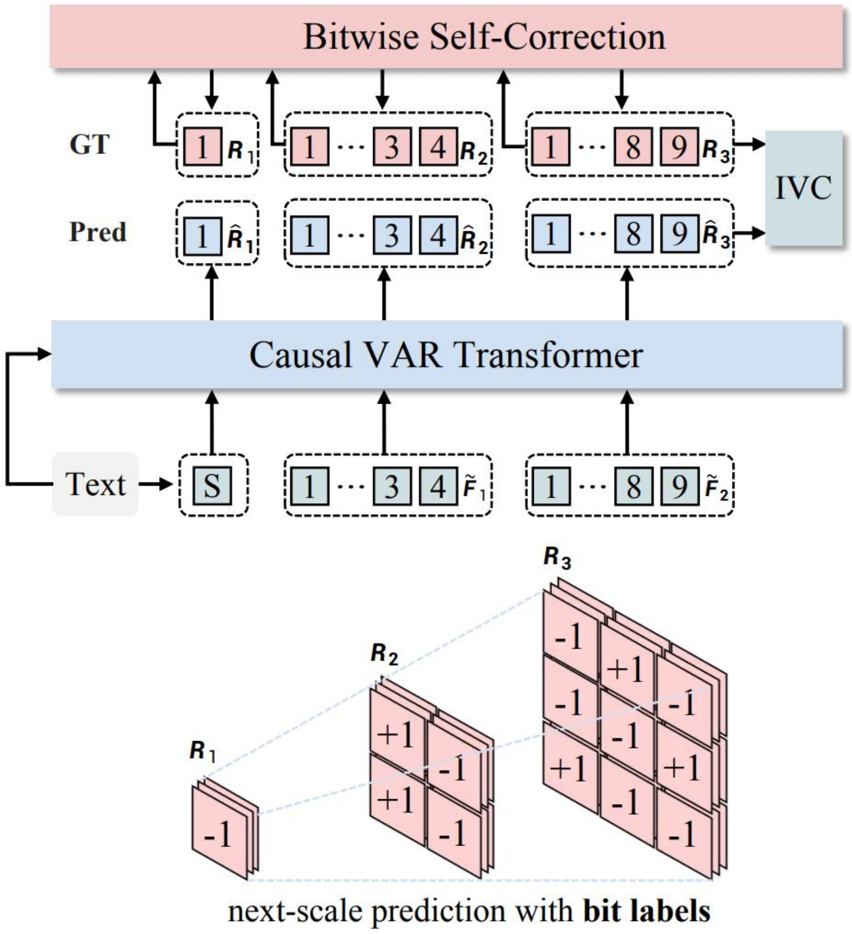
- Our bitwise modeling framework consists of 3 primary modules:
  - bitwise visual tokenizer,
  - bitwise infinite-vocabulary classifier, and
  - bitwise self-correction

结构-VAR与infinity的对比

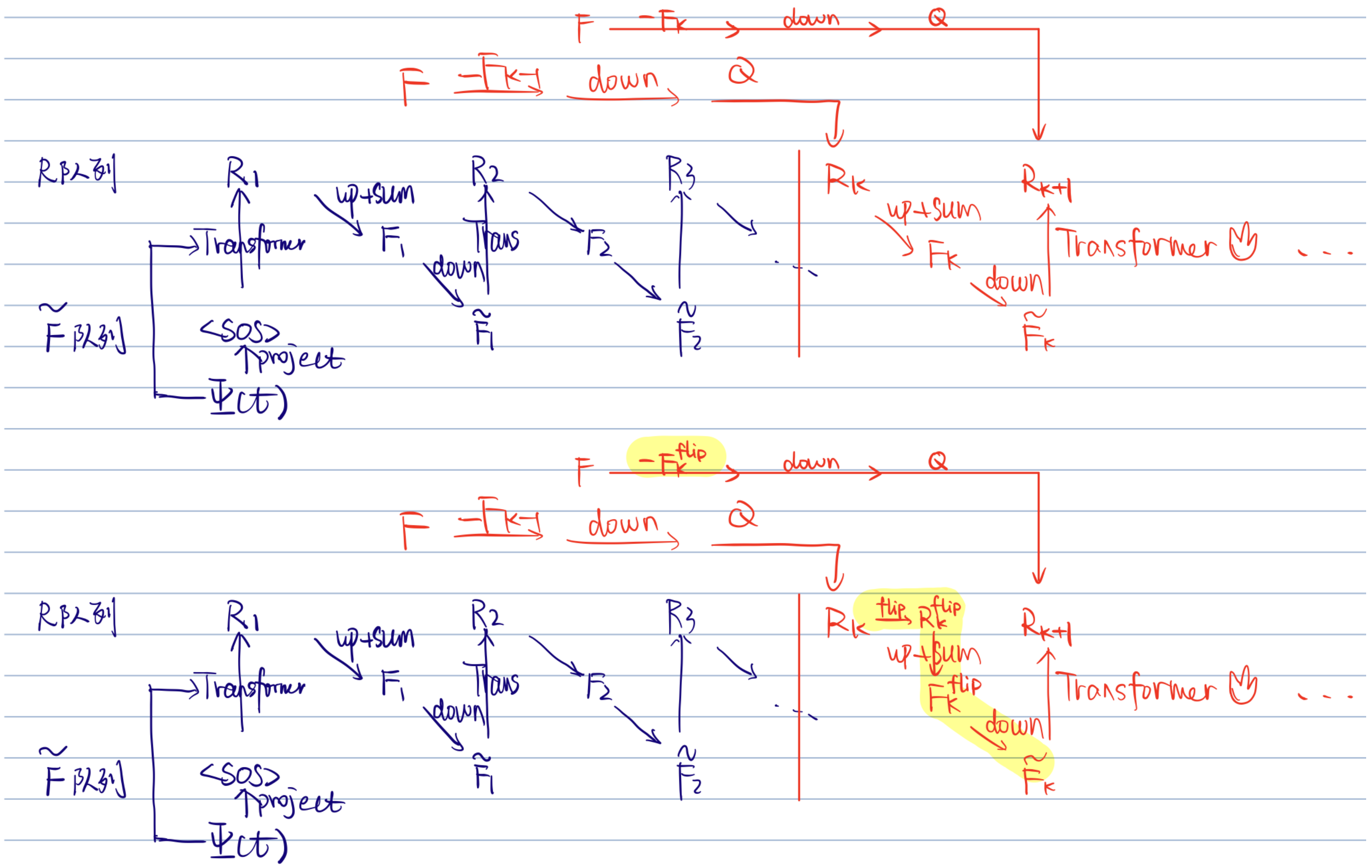


结构-VAR与infinity的对比





结构



F
-Fk
down
Q
F
-Fk-1
down
Q
R队列
R1
up+sum
R2
R3
Rk
Rk+1
Transformer
F1
Trans
F2
Fk
down
up+sum
Transformer
...
F̃队列
<SOS>
project
Ψ(t)
F̃1
F̃2
F̃k
F
-Fk flip
down
Q
F
-Fk-1
down
Q
R队列
R1
up+sum
R2
R3
Rk
flip
Rk flip
up+sum
Fk flip
down
Rk+1
Transformer
...
F1
Trans
F2
F̃队列
<SOS>
project
Ψ(t)
F̃1
F̃2
F̃k

结构

$$\boldsymbol{F}_k = \sum_{i=1}^{k} \mathrm{up}(\boldsymbol{R}_i, (h, w))$$

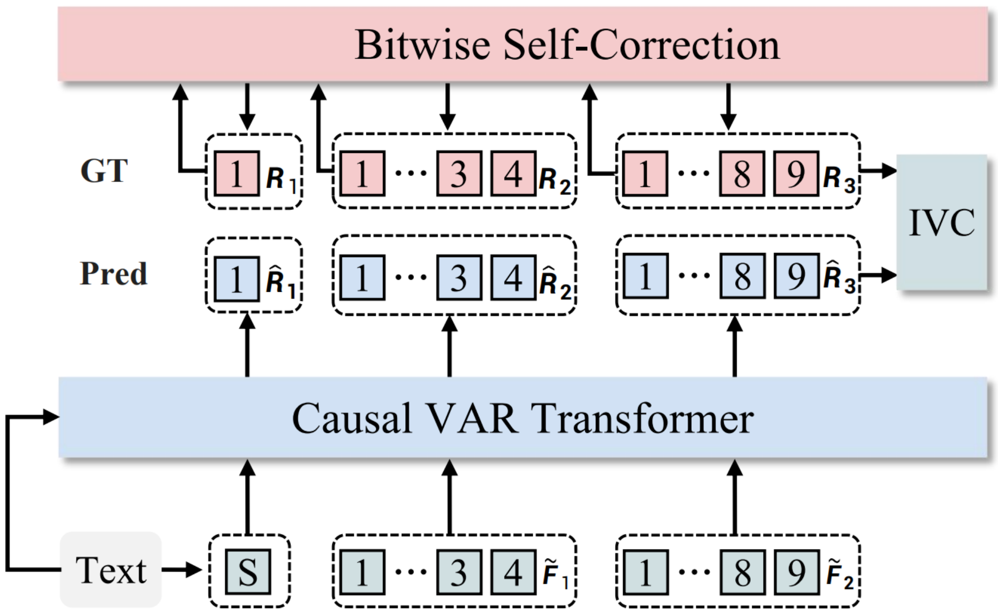
$$\underset{L\times C}{\Psi(t)} \xrightarrow{\text{proj}} \underset{1\times 1\times h}{\text{<SOS>}} \xrightarrow{\text{trans}} \underset{h_1\times w_1\times d}{R_1} \xrightarrow[\text{sum}]{\text{up}+} \underset{h\times w\times d}{F_1} \xrightarrow{\text{down}} \underset{h_2\times w_2\times d}{\tilde{F}_1} \xrightarrow{\text{trans}} \underset{h_2\times w_2\times d}{R_2} \rightarrow$$

......

<SOS>的h表示transformer的hidden dimension

Transformer Block:
RoPE2d –> block-wise causal self-attention
-> cross-attention(+text) -> FFN

在用transformer的时候理论上应该是把前面的token都放进去了,



Infinity to have the self-corr

结构

$F - F_{k-1} \xrightarrow{\text{down}} Q \rightarrow R_k \xrightarrow{\text{up+sum}} F_k \xrightarrow{\text{down}} \tilde{F}_k$

$h \times w \qquad h_k \times w_k \qquad h \times w \qquad h_{k+1}, w_{k+1}$

flip

flip

$R_k^{\text{flip}} \xrightarrow{\text{up+sum}}$

Algorithm 1 Visual Tokenizer Encoding

Input: raw feature $\boldsymbol{F}$, scale schedule $\{(h_1^r, w_1^r), \ldots, (h_K^r, w_K^r)\}$

$\boldsymbol{R}_{queue} = []$ ▷ *multi-scale bit labels*

$\widetilde{\boldsymbol{F}}_{queue} = []$ ▷ *inputs for transformer*

for $k = 1, 2, \cdots, K$ **do**

$\quad \boldsymbol{R}_k = \mathcal{Q}(\text{down}(\boldsymbol{F} - \boldsymbol{F}_{k-1}, (h_k, w_k)))$

$\quad \text{Queue\_Push}(\boldsymbol{R}_{queue}, \boldsymbol{R}_k)$

$\quad \boldsymbol{F}_k = \sum_{i=1}^{k} \text{up}(\boldsymbol{R}_i, (h, w))$

$\quad \widetilde{\boldsymbol{F}}_k = \text{down}(\boldsymbol{F}_k, (h_{k+1}, w_{k+1}))$

$\quad \text{Queue\_Push}(\widetilde{\boldsymbol{F}}_{queue}, \widetilde{\boldsymbol{F}}_k)$

end for

Output: $\boldsymbol{R}_{queue}, \widetilde{\boldsymbol{F}}_{queue}$

Algorithm 2 Encoding with BSC

Input: raw feature $\boldsymbol{F}$, random flip ratio p, scale schedule $\{(h_1^r, w_1^r), \ldots, (h_K^r, w_K^r)\}$,

$\boldsymbol{R}_{queue} = [], \widetilde{\boldsymbol{F}}_{queue} = []$

for $k = 1, 2, \cdots, K$ **do**

$\quad \boldsymbol{R}_k = \mathcal{Q}(\text{down}(\boldsymbol{F} - \boldsymbol{F}_{k-1}^{flip}, (h_k, w_k)))$

$\quad \text{Queue\_Push}(\boldsymbol{R}_{queue}, \boldsymbol{R}_k)$

$\quad \boldsymbol{R}_k^{flip} = \text{Random\_Flip}(\boldsymbol{R}_k, p)$

$\quad \boldsymbol{F}_k^{flip} = \sum_{i=1}^{k} \text{up}(\boldsymbol{R}_i^{flip}, (h, w))$

$\quad \widetilde{\boldsymbol{F}}_k = \text{down}(\boldsymbol{F}_k^{flip}, (h_{k+1}, w_{k+1}))$

$\quad \text{Queue\_Push}(\widetilde{\boldsymbol{F}}_{queue}, \widetilde{\boldsymbol{F}}_k)$

end for

Output: $\boldsymbol{R}_{queue}, \widetilde{\boldsymbol{F}}_{queue}$

量化方法：VQ vs FSQ

VQ-VAE

$$z = encoder(x)$$

$$z_q = z + \mathrm{sg}[e_k - z], \quad k = \underset{i \in \{1,2,\cdots,K\}}{\arg\min} \|z - e_i\|$$

$$\hat{x} = decoder(z_q)$$

$$\mathcal{L} = \|x - \hat{x}\|^2 + \beta\|e_k - \mathrm{sg}[z]\|^2 + \gamma\|z - \mathrm{sg}[e_k]\|^2$$

Codebook 坍缩

VQ-VAE-2

$$e_k^{(t)} = \alpha e_k^{(t-1)} + (1-\alpha)z$$

FSQ

$$\mathrm{FSQ}(t) \triangleq \mathrm{Round}[(L-1)\sigma(t)]$$

$$\mathrm{FSQ}(z) = \mathrm{Round}[(L-1)\sigma(z)] \in \{0, 1, \cdots, L-1\}^d$$

$$\mathrm{FSQ}(z) = (L-1)\sigma(z) + \mathrm{sg}\big[\mathrm{Round}[(L-1)\sigma(z)] - (L-1)\sigma(z)\big]$$

量化方法： Residual VQ (RVQ)

- 多级量化，固定一组codebook挨个过一遍，结果就是各个codebook量化结果相加
- 此外还有AVQ加法矢量量化，在多个codebook中联合寻找最优量化值（有点贪心算法的意思）

量化方法：LFQ与BSQ

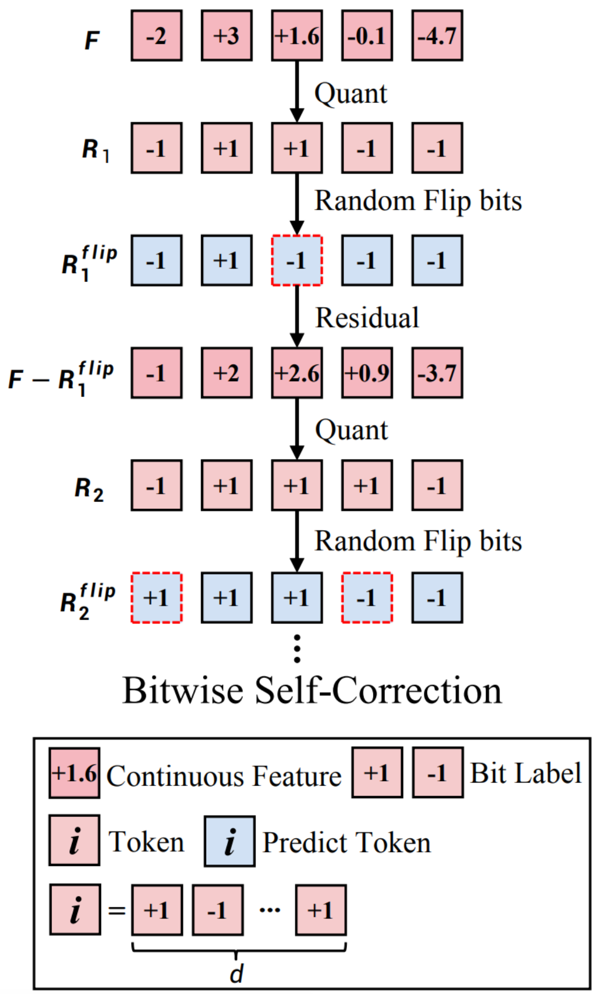
and BSQ[79]. Given K scales in the multi-scale quantizer, on the k-th scale, the input continuous residual vector $z_k \in \mathbb{R}^d$ are quantized into binary output q_k as shown below.

$$q_k = \mathcal{Q}(z_k) = \begin{cases} \text{sign}(z_k) & \text{if LFQ} \\ \frac{1}{\sqrt{d}}\text{sign}(\frac{z_k}{|z_k|}) & \text{if BSQ} \end{cases} \tag{4}$$

since both input and output in BSQ are unit vectors, BSQ[79] proposes an approximation formula for the entropy penalty, reducing the computational complexity to $O(d)$. As shown in Tab 3, there is no obvious increase in memory consumption for BSQ even when codebook size is 2^{64}. Unless otherwise stated, we adopt BSQ by default.

Bitwise self-correction

- Teacher-forcing training



可变宽高比

3.5 Dynamic Aspect Ratios and Position Encoding

Infinity can generate photo-realistic images with various aspect ratios, which is significantly different from VAR [61] that can only generate square images. The main obstacles of generating various aspect ratio images lie in two folds. The first is to define the height h_k and width w_k of $\boldsymbol{R}_k$ based on varying aspect ratios. In the supplementary material, we pre-define a list of scales, also called scale schedule, as $\{(h_1^r, w_1^r), ..., (h_K^r, w_K^r)\}$ for each aspect ratio. We ensure that the aspect ratio of each tuple (h_k^r, w_k^r) is approximately equal to r, especially in the latter prediction scales. Additionally, for different aspect ratios at the same scale k, we keep the area of $h_k^r \times w_k^r$ to be roughly equal, ensuring that the training sequence lengths are roughly the same.

Secondly, we need to carefully design a resolution-aware positional encoding method to handle features of various scales and aspect ratios. This issue poses a significant challenge, as the existing solutions [65, 61, 54, 26, 40] exhibit substantial limitations under such conditions. In this paper, we apply RoPE2d [26] on features of each scale to preserve the intrinsic 2D structure of images. Additionally, we exploit learnable scale embeddings to avoid confusion between features of different scales. Compared to learnable APE element-wisely applied on features, learnable embeddings applied on scales bring fewer parameters, can adapt to varying sequence lengths, and are easier to optimize.

训练数据

Data Curation. We curated a large-scale dataset from open-source academic data and high-quality internally collected data. The pre-training dataset is constructed by collecting and cleaning open-source academic datasets such as LAION [51], COYO [10], OpenImages [33]. We exploit an OCR model and a watermark detection model to filter undesired images with too many texts or watermarks. Additionally, we employ Aesthetic-V2 to filter out images with low aesthetic scores.

文本模型的替换

- 在原来的模型中文本信息在推理时输入到了两个位置：
  - 第一个token（<SOS>）
  - 每一个transformer的cross attention中

Transformer Block:
RoPE2d –> block-wise causal self-attention
-> cross-attention(+text) -> FFN