Finite Scalar Quantization: VQ-VAE Made Simple

Zou Lexiao

2023/12/26

Background

OpenReview.net Search OpenReview... Login

← Go to **ICLR 2024 Conference** homepage

Language Model Beats Diffusion - Tokenizer is key to visual generation

OpenReview.net Search OpenReview... Login

← Go to **ICLR 2024 Conference** homepage

Finite Scalar Quantization: VQ-VAE Made Simple

ICLR 2024 Conference Submission1937 Authors

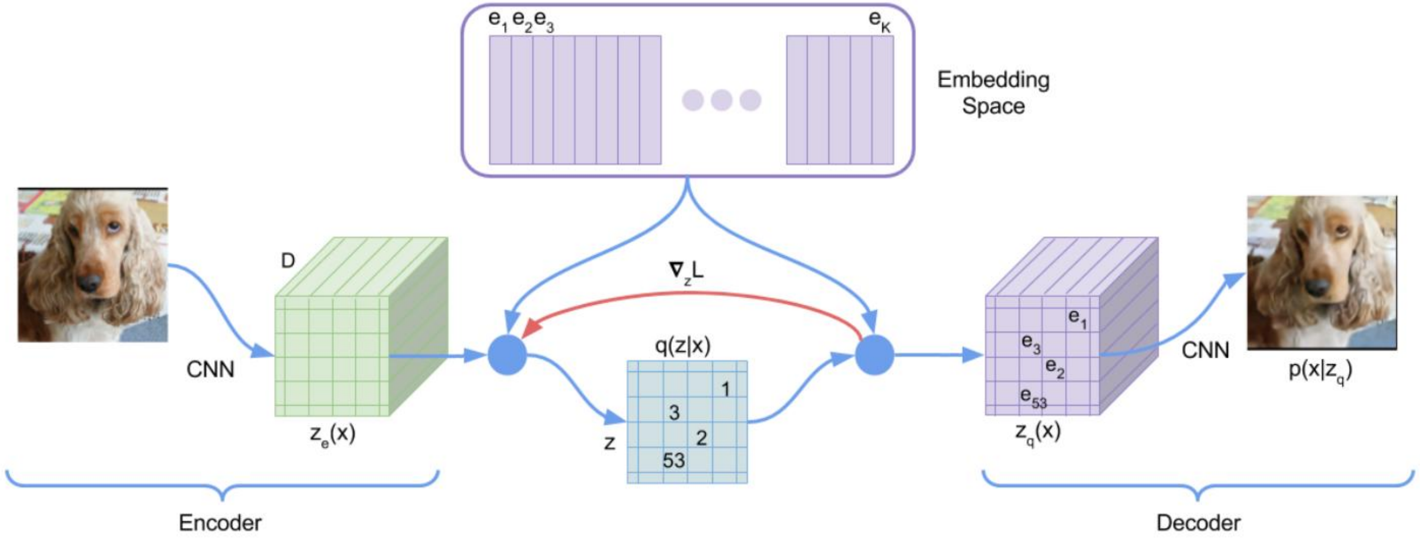
Ranked #1 on Image Generation on ImageNet 512x512 → Get a GitHub badge

Task	Dataset	Model	Metric Name	Metric Value	Global Rank	Result	Benchmark
Image Generation	ImageNet 256x256	MAGVIT-v2 (w/o guidance)	FID	3.65	# 20		Compare
Image Generation	ImageNet 256x256	MAGVIT-v2	FID	1.78	# 3		Compare
Image Generation	ImageNet 512x512	MAGVIT-v2 (w/o guidance)	FID	3.07	# 7		Compare
			Inception score	213.1	# 9		Compare
Image Generation	ImageNet 512x512	MAGVIT-v2	FID	1.91	# 1		Compare
			Inception score	324.3	# 3		Compare
Video Prediction	Kinetics-600 12 frames, 64x64	MAGVIT-v2	FVD	4.3±0.1	# 1		Compare
Video Generation	Kinetics-600 12 frames, 64x64	MAGVIT-v2	FVD	4.3±0.1	# 2		Compare
Video Generation	UCF-101	MAGVIT-v2	FVD16	58±3	# 2		Compare

Vector quantization (VQ) has recently seen a **renaissance** in the context of learning discrete representations with neural networks due to the success of large language model.

The core problem is how to tokenize continous signal into discrete token sequence. Once we obtain the token sequence, we can handle it with language model.

Background: VQ-VAE

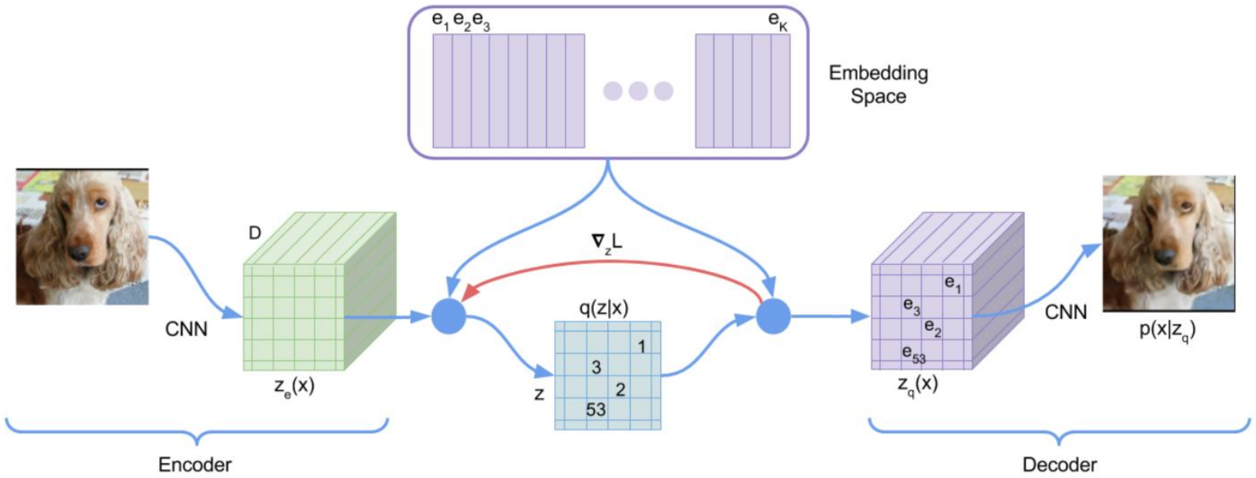


$$z = \text{encoder}(x)$$

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

$$\hat{x} = \text{decoder}\left(z_q\right)$$

Background: VQ-VAE



- Update Encoder by **Straight-Through Estimator**

$$z_q = e_k, \nabla z_q = \nabla z$$

  - Commitment Loss: $\gamma \left\| z - \mathrm{sg}\left[e_k\right] \right\|^2$

- Update Codebook by $\beta \left\| e_k - \mathrm{sg}[z] \right\|^2$

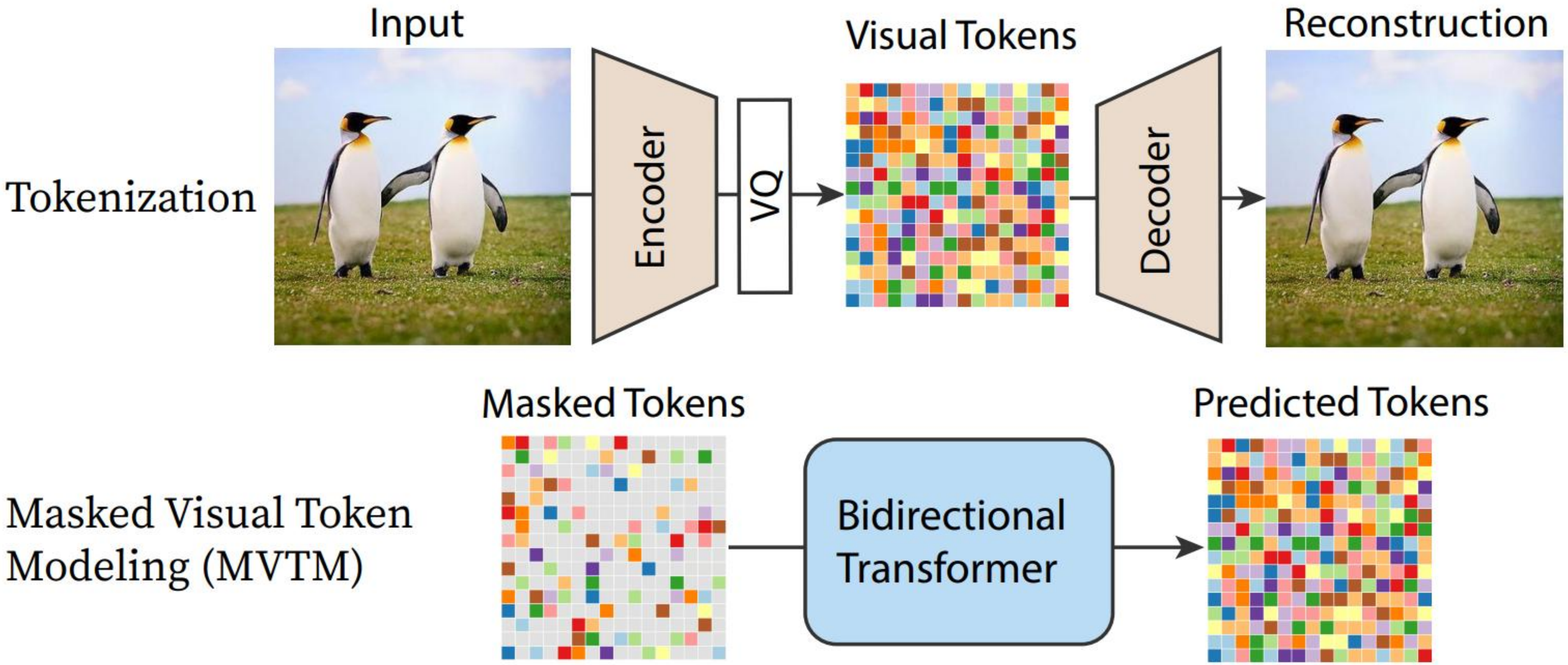
$$z = \mathrm{encoder}(x)$$

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

$$\hat{x} = \mathrm{decoder}\left(z_q\right)$$

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

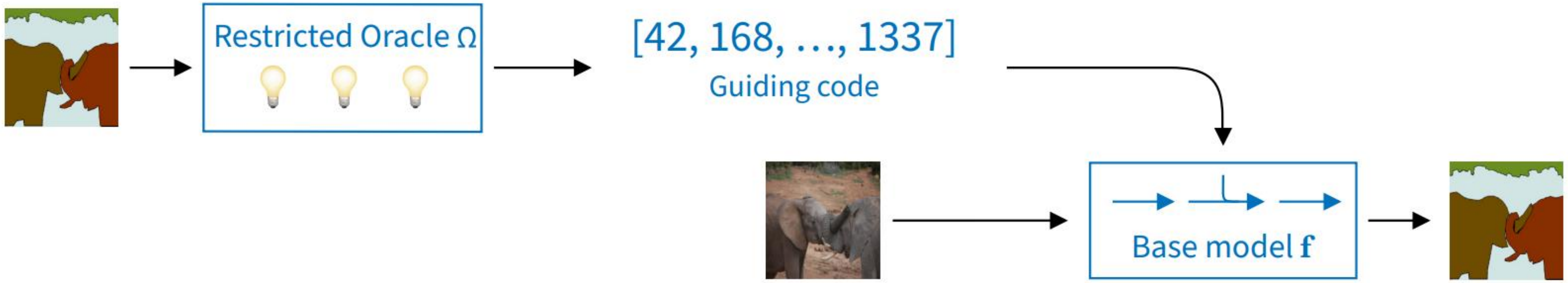
Background: Image Generation with Visual Tokens （MaskGIT）



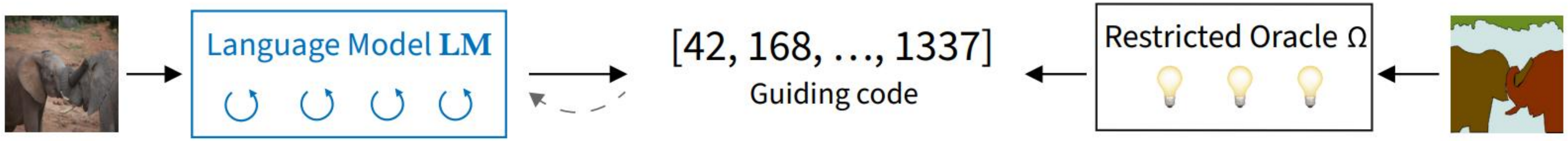
Stage 1 A tokenizer that tokenizes images into visual tokens

Stage 2 A bidirectional tranformer model that performs MVTM, i.e. learns to predict visual tokens masked at random

Background: Image Understanding with Visual Tokens （UViM）



(a) **Stage I** training: we train the base model f, which is guided by the code produced by the *restricted oracle* model Ω. The oracle has access to the ground-truth label, but is only allowed to communicate with f by passing a short discrete sequence, which we call a *guiding code*.



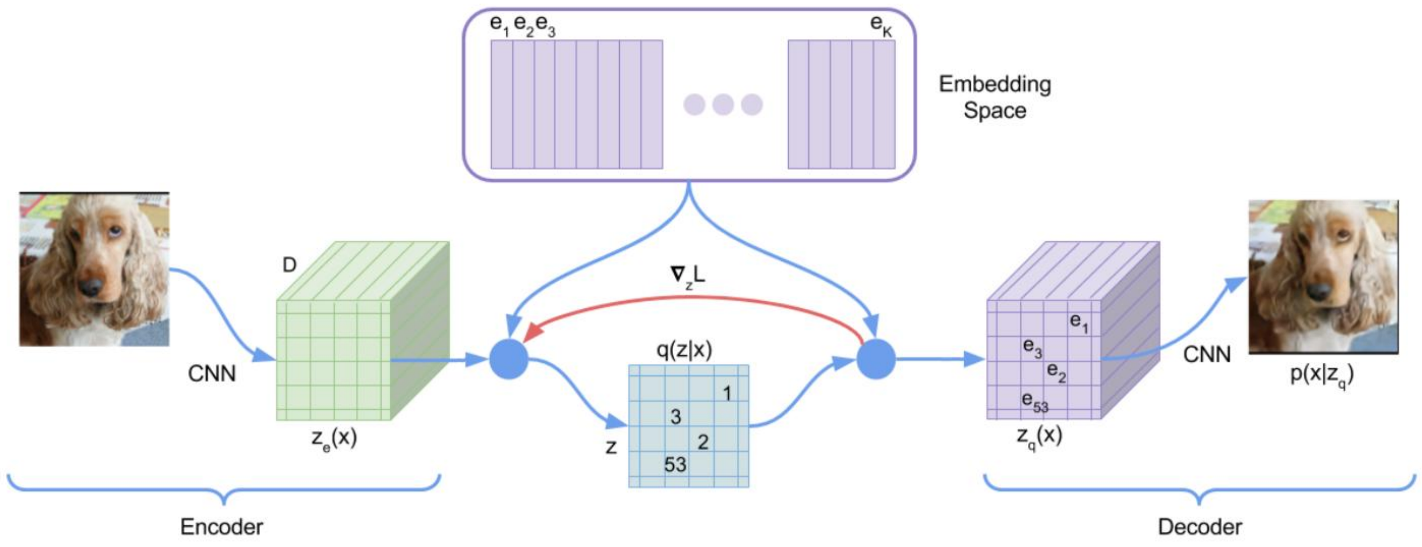
(b) **Stage II** training: we train a *language model* (LM) to output a *guiding code* by learning to mimic the oracle, but using only the image input.

(segmentation as an example)

Stage 1 A tokenizer that tokenizes target into visual tokens

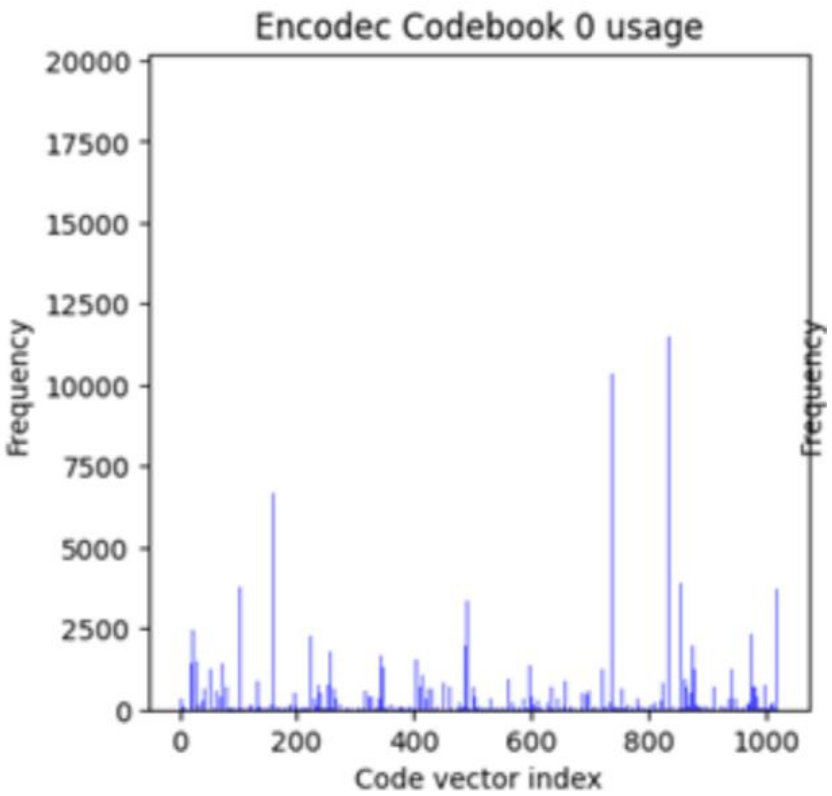
Stage 2 A vision tranformer model which takes origin image in the encoder and predict guiding code autoregressively by decoder

Background



- The upper bounds of these models are reconstruction by origin VQ decoder
  - Intuitiely, as we have more bits to store information, we should get better reconstruction metrics
  - Note that he size of natural language codebook is usually over 200k (2^18) compared to 1k in common visual setting

Motivation



- VQ-VAE usually leads to unbalanced code usage, mainly due to the optimization of the explicit codebook
  - Achieve high codebook utilization by design
- Keep the functional setup the same to the extent that we obtain a drop-in replacement for VQ

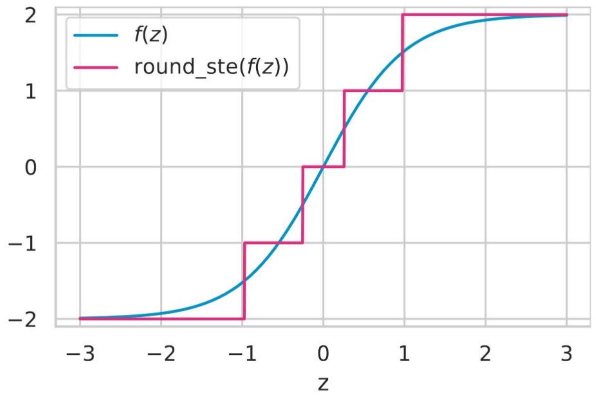
Method: Finite Scalar Quantization

- Given an output of encoder $z \in \mathbb{R}^d$, for each dimension z_i,

$$\bar{z}_i = round(f(z_i))$$

where $\bar{z} \in \mathcal{C}$, a implied codebook with L unique values for each dimension, $|\mathcal{C}| = L^d$. f is a bounding function

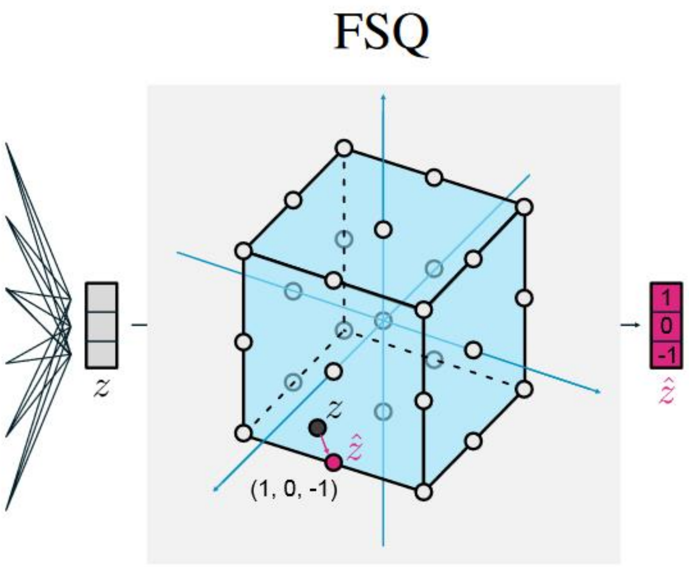
e.g. $f : z \to \lfloor L/2 \rfloor \tanh(z)$

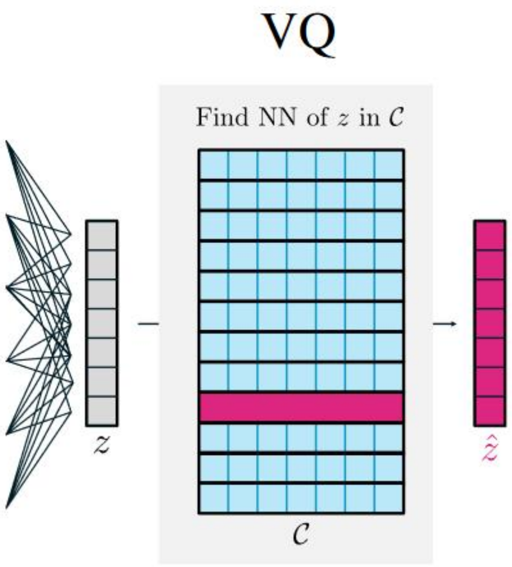


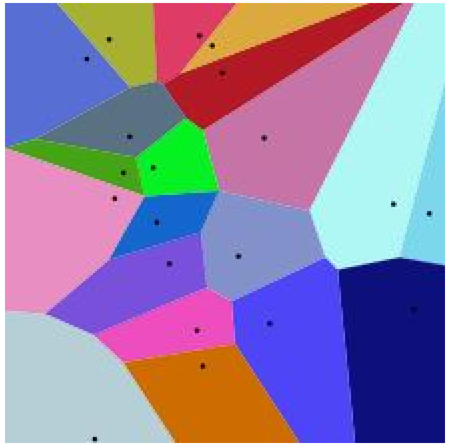
The vectors in $\mathcal{C}$ can be enumerated leading to a bijection from any $\bar{z}$ to an integer in $\{1,\ldots,L^d\}$

Method: Finite Scalar Quantization

- Intuitively







Grid partition

Voronoi parition

Method: Finite Scalar Quantization

- **Straight-Through Estimator** for propagate gradient

```
def round_ste(z):
  """Round with straight through gradients."""
  zhat = jnp.round(z)
  return z + jax.lax.stop_gradient(zhat - z)
```

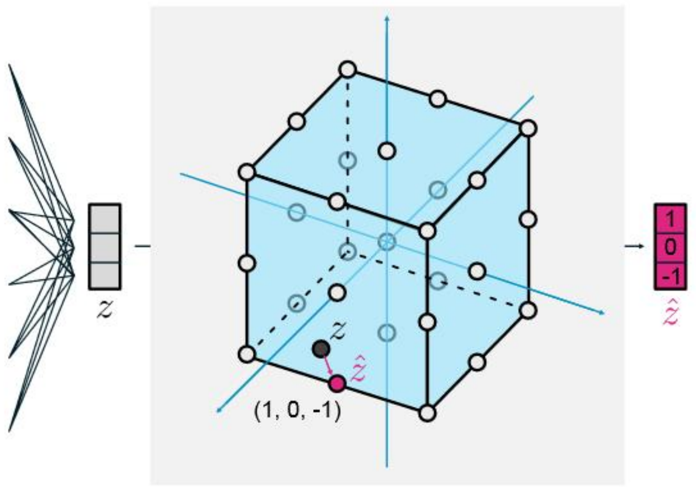
Method: Finite Scalar Quantization

- Hyperparameters: number of channels d & number of levels per channel, $L = [L_1, \ldots, L_n]$
  - e.g.

2^4	2^6	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{14}	2^{16}
$[5,3]$	$[8,8]$	$[8,6,5]$	$[8,8,8]$	$[8,5,5,5]$	$[8,8,6,5]$	$[7,5,5,5]$	$[8,8,8,6,5]$	$[8,8,8,5,5,5]$

  - a simple heuristic: $L_i \geq 5, \forall i$

FSQ

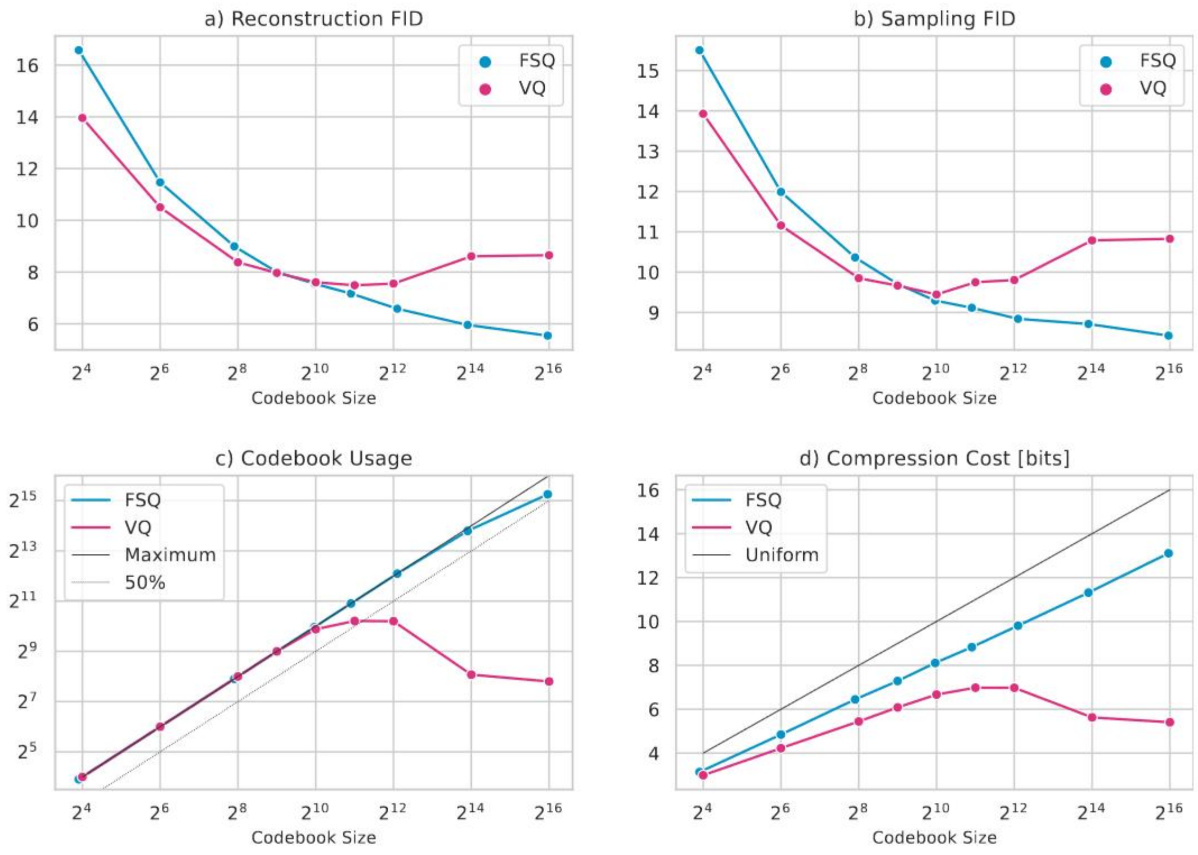


Experiment

- Codebook size correlates with FID for FSQ

  MaskGIT on ImageNet 128*128

  - We see that Reconstruction FID correlates with codebook size for FSQ, and improves as we scale the codebook size
  - FSQ gets better Sampling FID and higher codebook usage for codebook size exceeding 1024, while the metrics start deteriorating for VQ
  - For low codebook sizes, VQ marginally outperforms FSQ, likely owning to the its more expressive nature

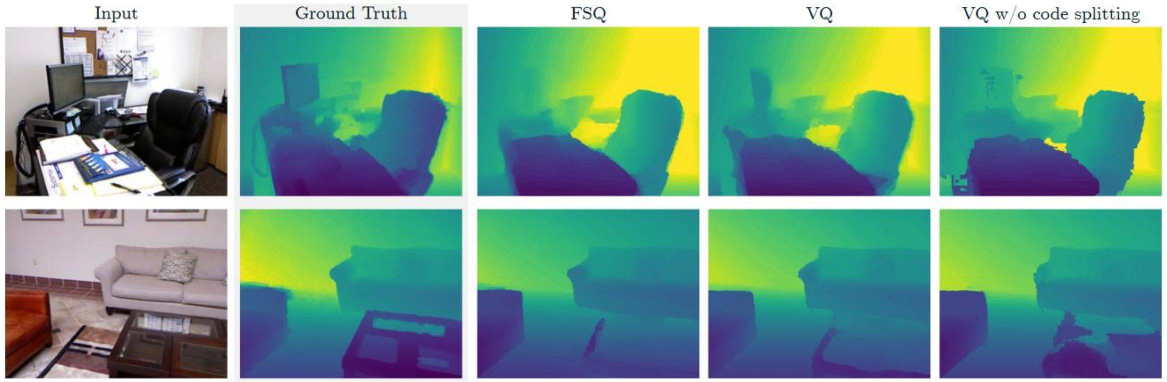


Experiment

- FSQ performs as good as VQ

  UViM for depth estimation, panoptic segmentation & colorization

  - FSQ is competitive with VQ on all tasks
  - FSQ does not rely on codebook splitting



NYU Depth v2	Source	RMSE† ↓	Codebook Usage
UViM (VQ)	Ours	0.468 ± 0.012	99%
UViM (FSQ)	Ours	0.473 ± 0.012	99%
UViM (VQ without splitting)	Ours	0.490 ± 0.0037	0.78%
UViM (VQ)	GitHub	0.463	
DenseDepth (Alhashim & Wonka, 2018)		0.465	
COCO Panoptic	Source	PQ† ↑	Codebook Usage
UViM (VQ)	Ours	43.4 ± 0.0008	100%
UViM (FSQ)	Ours	43.2 ± 0.0014	100%
UViM (VQ without context)	Ours	39.0 ± 0.0023	99%
UViM (FSQ without context)	Ours	40.2 ± 0.0019	99%
UViM (VQ)	GitHub	43.1	
DETR-R101 (Carion et al., 2020)		45.1	
ImageNet Colorization	Source	FID-5k† ↓	Codebook Usage
UViM (VQ)	Ours	16.90 ± 0.056	100%
UViM (FSQ)	Ours	17.55 ± 0.057	100%
UViM (VQ)	Github	16.99 ± 0.057	
ColTran (Kumar et al., 2021)		19.37	

Discussions

- Language Model Beats Diffusion -- Tokenizer is Key to Visual Generation
  - Look Up Free Quantization

$$q(z_i) = \operatorname{sign}(z_i) = -\mathbb{1}\{z_i \leqslant 0\} + \mathbb{1}\{z_i > 0\}$$

$$\mathcal{L}_{\text{entropy}} = \mathbb{E}[H(q(\mathbf{z}))] - H[\mathbb{E}(q(\mathbf{z}))]$$

FSQ

$$\bar{z} = round(\tanh(z))$$

  - Experiment
    - Scale up! Transformer encoder with two heads predicts a codebook with a size of 512 individually

Table 2: **Image generation results**: class-conditional generation on ImageNet 512×512. Guidance indicates the classifier-free diffusion guidance (Ho & Salimans, 2021). * indicates usage of extra training data. We adopt the evaluation protocol and implementation of ADM.

Type	Method	w/o guidance FID↓	w/o guidance IS↑	w/ guidance FID↓	w/ guidance IS↑	#Params	#Steps
GAN	StyleGAN-XL (Sauer et al., 2022)			2.41	267.8	168M	1
Diff. + VAE*	DiT-XL/2 (Peebles & Xie, 2022)	12.03	105.3	3.04	240.8	675M	250
Diffusion	ADM+Upsample (Dhariwal & Nichol, 2021)	9.96	121.8	3.85	221.7	731M	2000
Diffusion	RIN (Jabri et al., 2023)	3.95	216.0			320M	1000
Diffusion	simple diffusion (Hoogeboom et al., 2023)	3.54	205.3	3.02	248.7	2B	512
Diffusion	VDM++ (Kingma & Gao, 2023)	2.99	232.2	2.65	278.1	2B	512
MLM + VQ	MaskGIT (Chang et al., 2022)	7.32	156.0			227M	12
MLM + VQ	DPC+Upsample (Lezama et al., 2023)	3.62	249.4			619M	72
MLM + LFQ	*MAGVIT-v2 (this paper)*	4.61	192.4			307M	12
		3.07	213.1	**1.91**	**324.3**		64

Discussions

- Contributions
  - A simple yet effective method which is a drop-in replacement for VQ in various architectures
  - FSQ is able to leverage large codebooks for better reconstruction metrics, and better sample quality without relying on any auxiliary losses
    - FSQ avoid optimizing a explicit codebook
  - It is very surprising that FSQ obtains such competitive results compared to VQ -- despite not using any of its auxiliary losses and trick
    - Intuitively, FSQ works well here because the generality of a learned codebook is “absorbed” into the encoder transform
  - A promising direction to study bound function, parameterization, its combination with AR-LM, multimodality training(e.g. text to image, text to speech) etc.